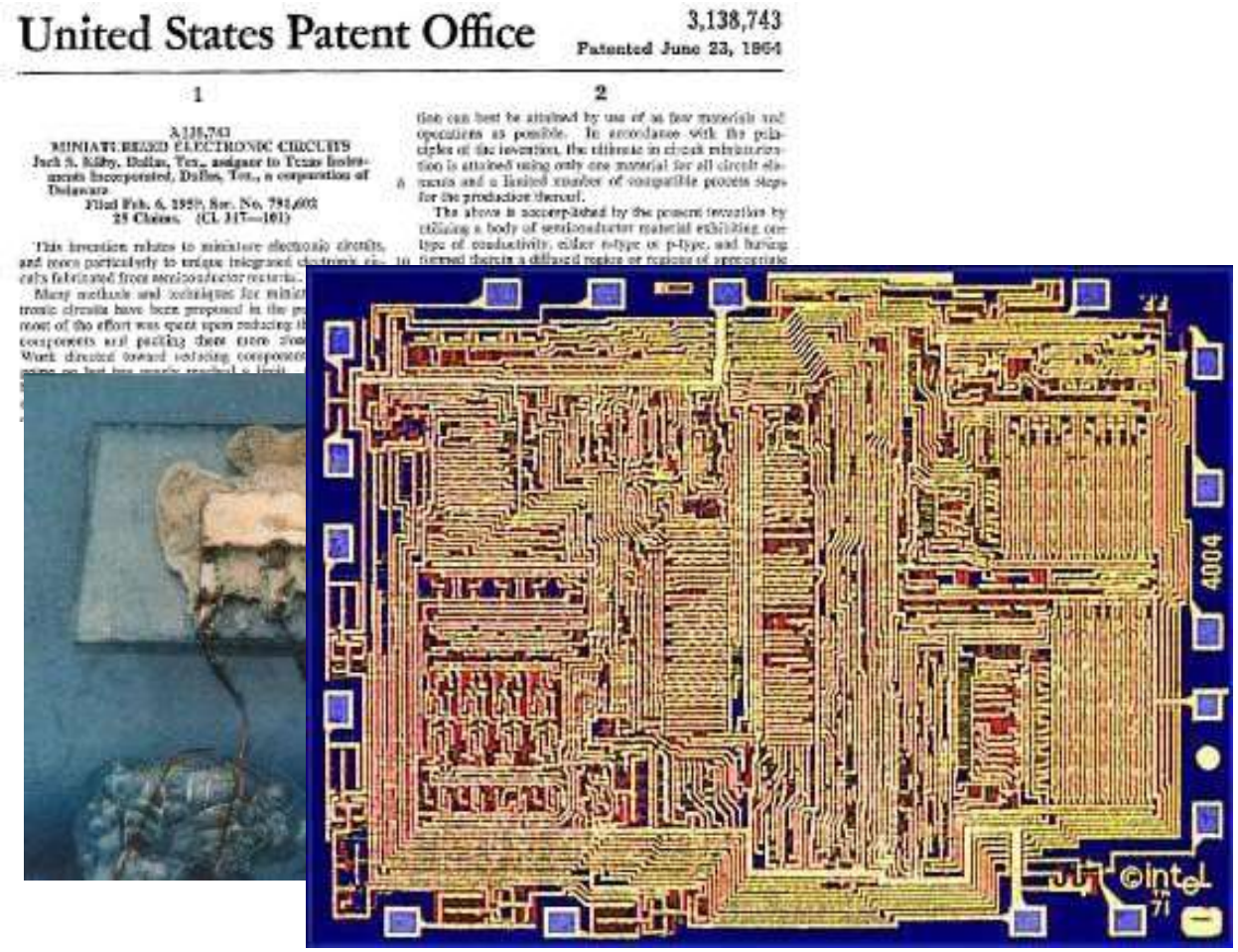
Selected Topics In VLSI

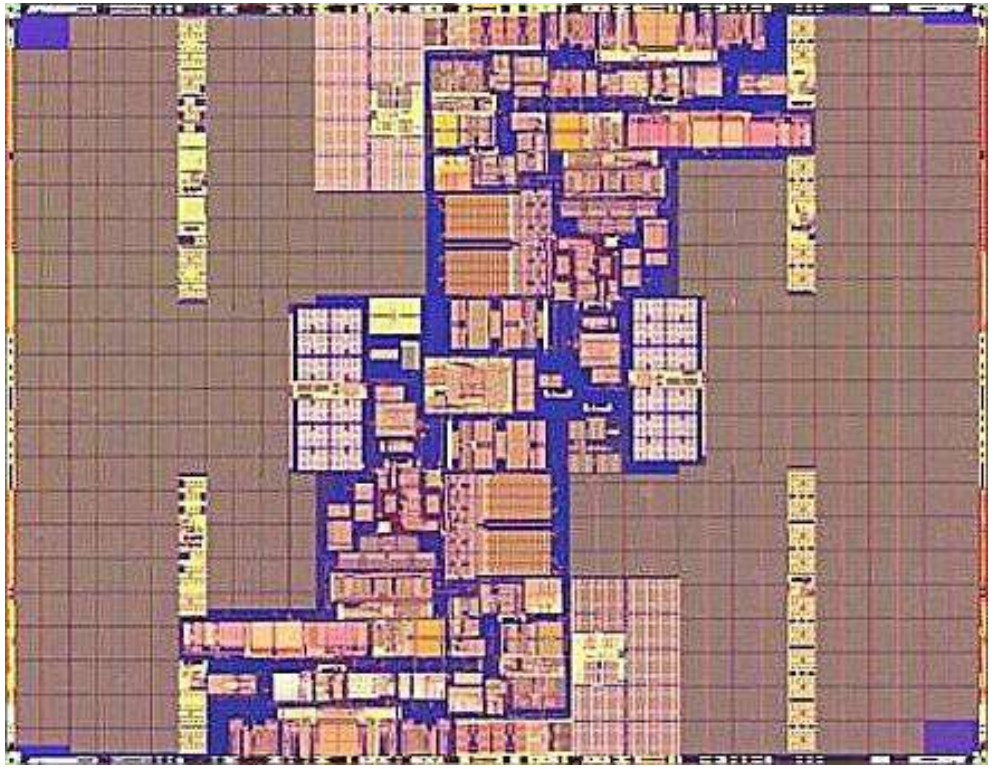
Lecture 1
Process Design Flow
Introduction

Motivation and Introduction

Motivation



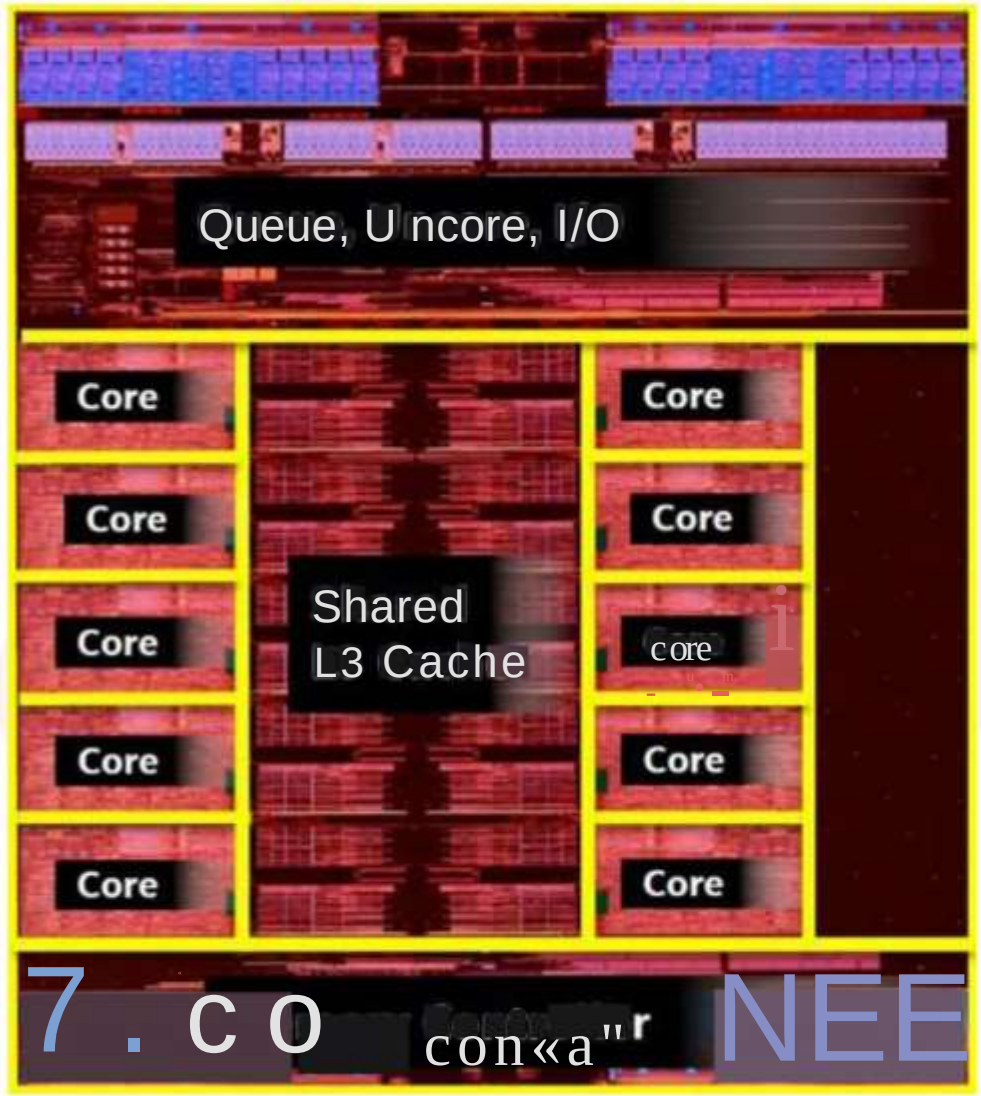
1971 -The Intel 4004
2,300 Transistors

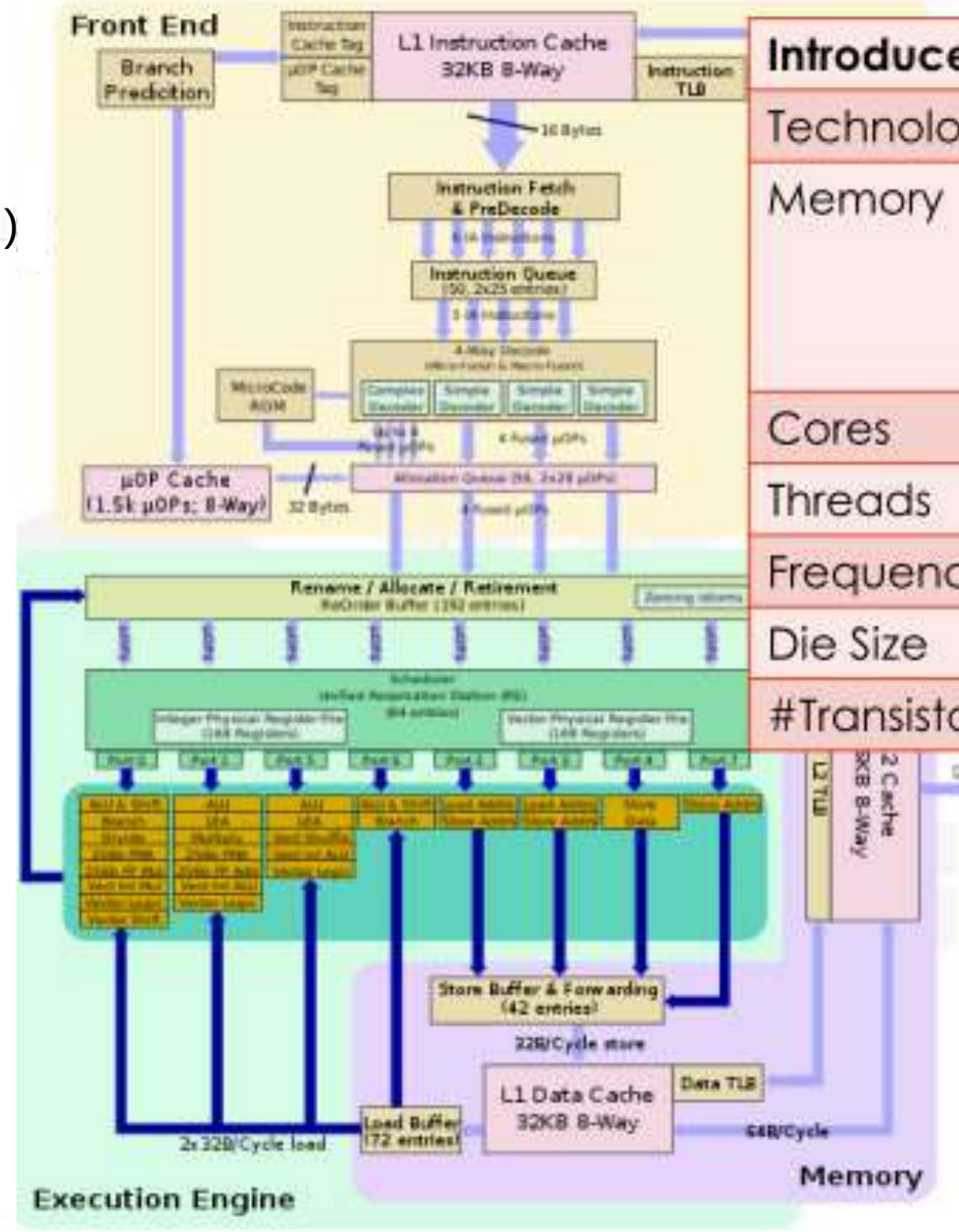


2006 - Itanium 2 "Montecito"
1.7B Transistors

Motivation

Core i7-6950X Extreme Edition (Broadwell-E)

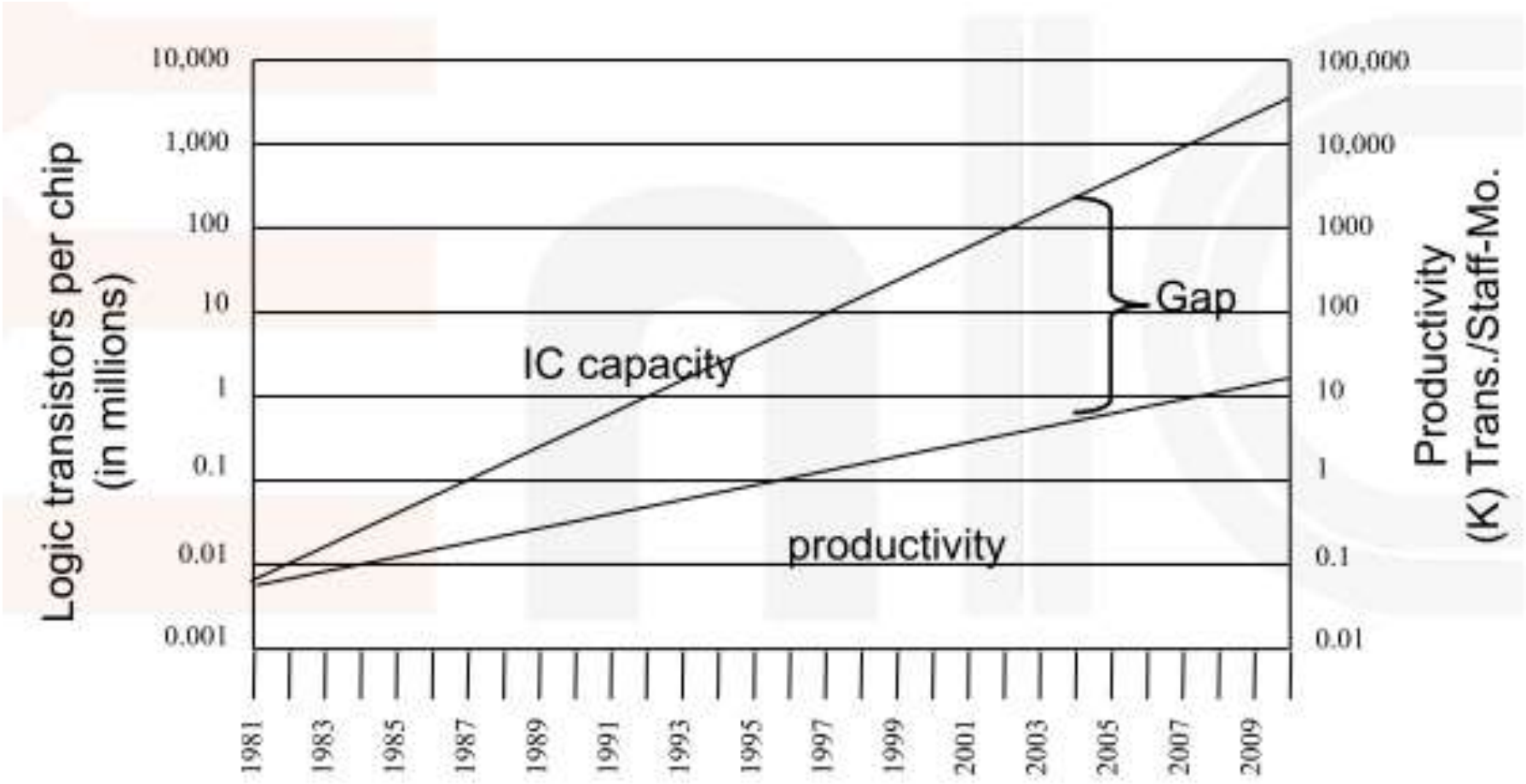




Introduced	May 2016
Technology	14nm FinFET
Memory	320KB L1-I$ 320KB L1-D$ 2.56MB L2$ 25MB L3$
Cores	10
Threads	20
Frequency	3.0-3.5 GHz
Die Size	246 mm²
#Transistors	3.2 B

http://en.wikichip.org/wiki/intel/core_i7/i7-6950x

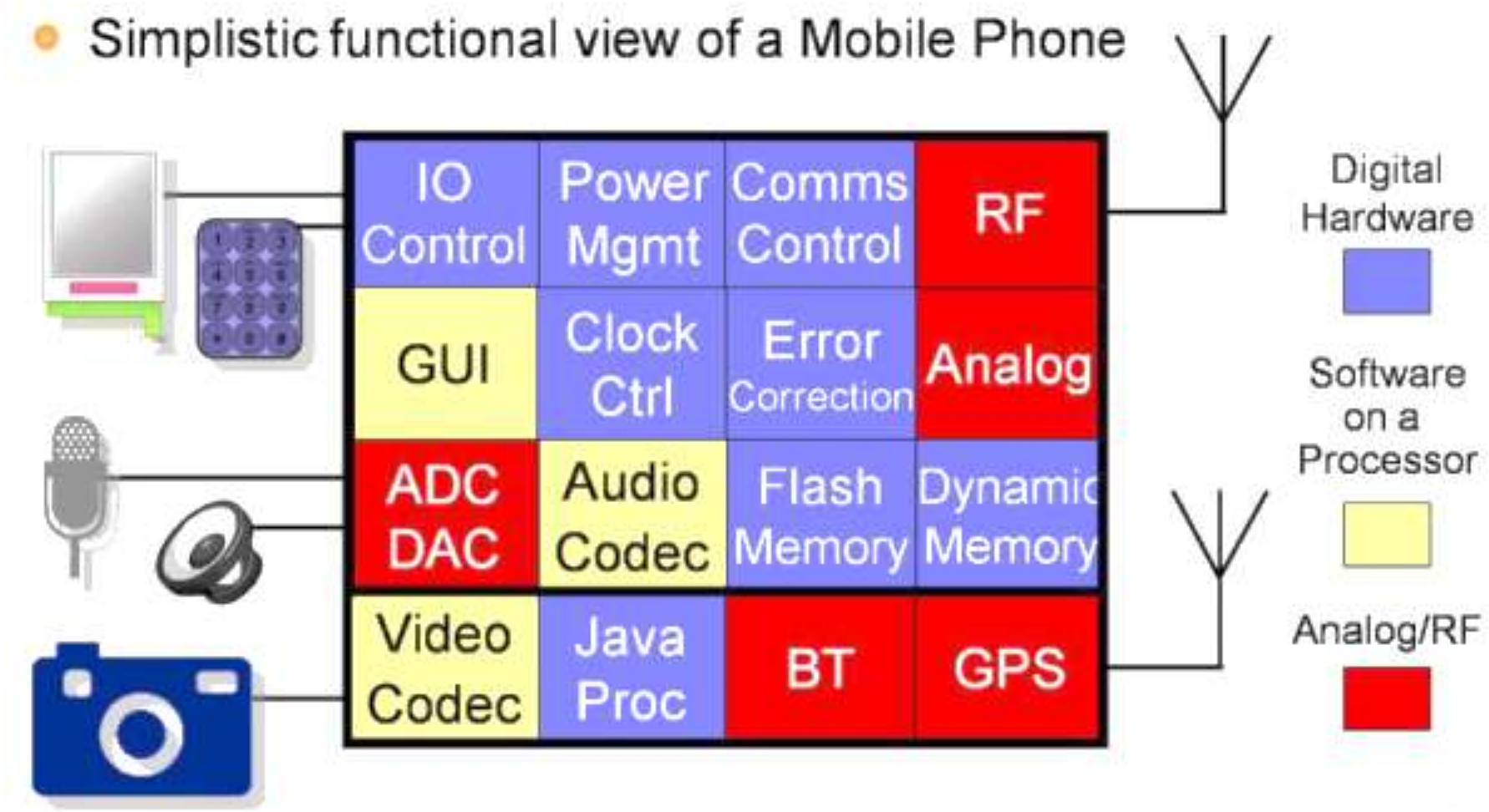
Motivation



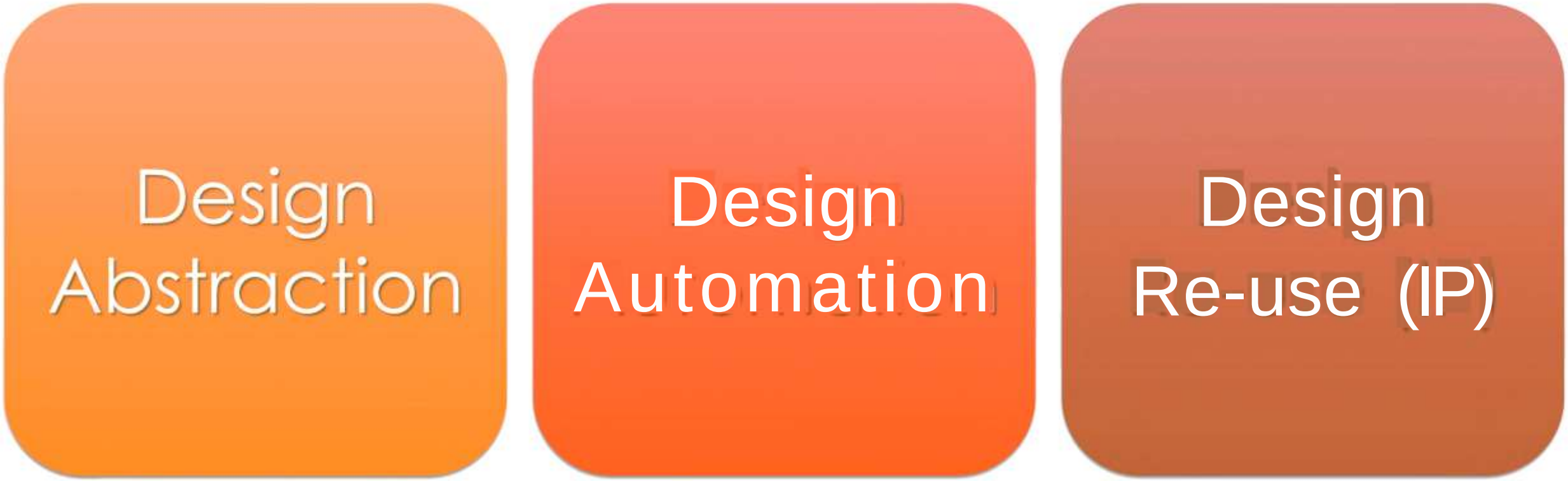
"Moore's Law of Engineers"

Motivation

- How on earth do we design such a thing???

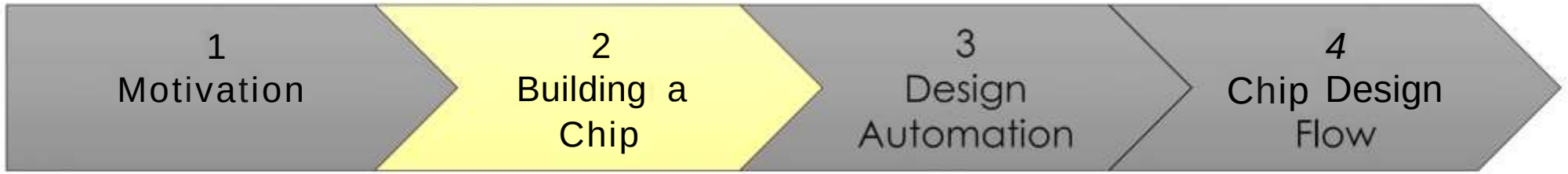


The Solution:



Syllabus

- Lecture 1: Introduction
- Lecture 2: Verilog
- Lecture 3: Logic Synthesis
- Lecture 4: Static Timing Analysis
- Lecture 5: Moving to the Physical Domain
- Lecture 6: Placement
- Lecture 7: Clock Tree Synthesis
- Lecture 8: Routing
- Lecture 9: 1/0 and Packaging
- Lecture 10: Design for Test

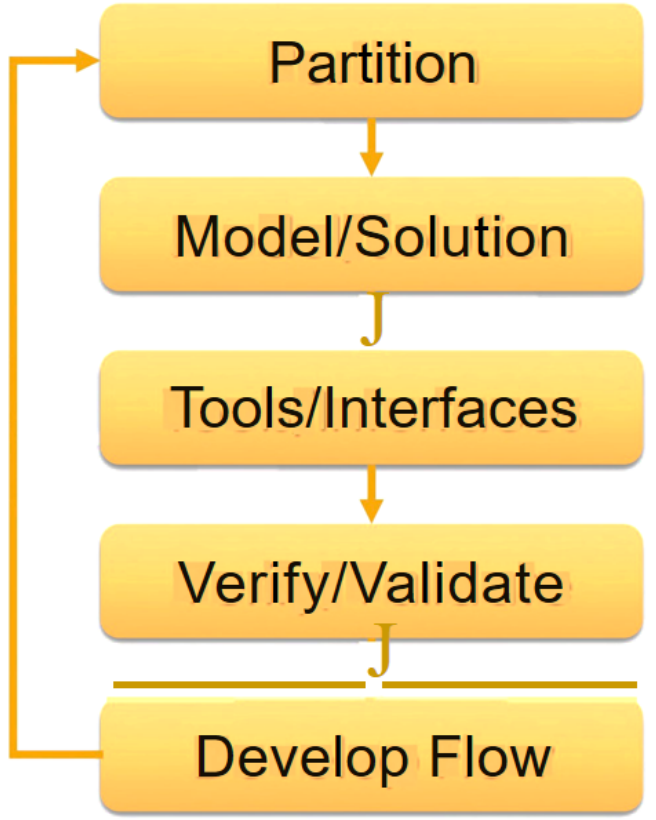


Building a Chip

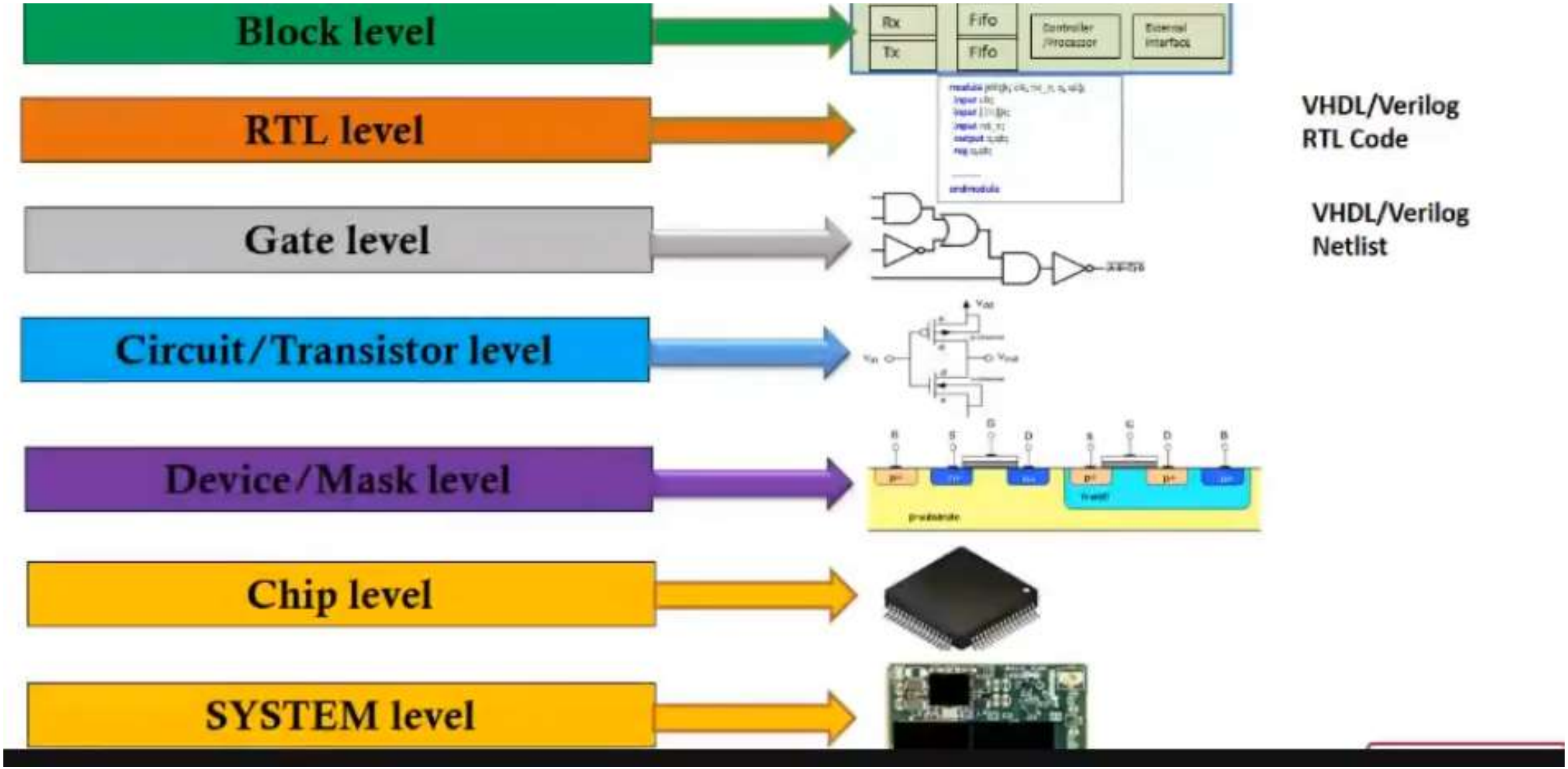
General Design Approach

- **How do engineers build a bridge?**

- **Divide and conquer!!!!**
  - **Partition** design problem into many sub-problems, which are manageable
  - Define mathematical **model** for sub-problem and find an **algorithmic** solution
    - Beware of model limitations and check them !!!!!!
  - Implement algorithm in **individual design tools,** define and implement general **interfaces** between the tools
  - Implement **checking tools** for boundary conditions
  - Concatenate design tools to general **design flows** which can be managed
  - See what doesn't work and **start over.**



VLSI System Design Flow



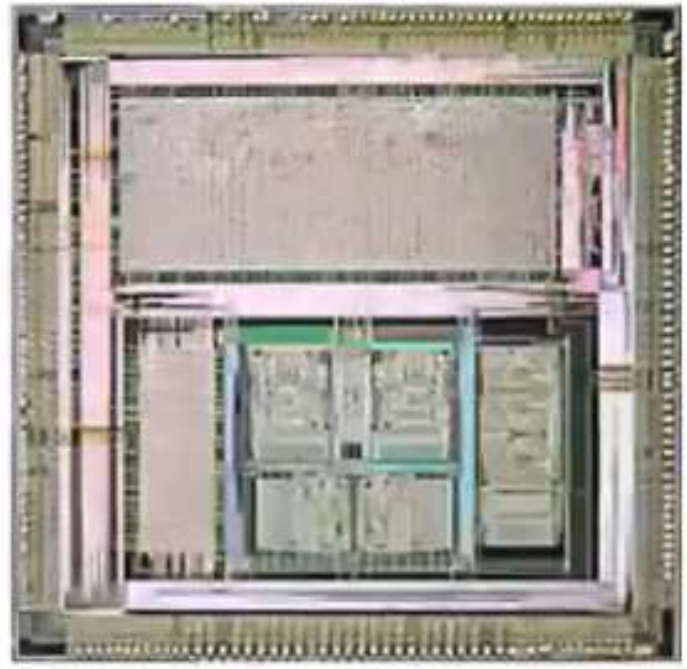
VLSI Design

FPGAs
A **field-programmable gate array** (**FPGA**) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable".
FPGAs contain an array of programmable logic blocks, and a hierarchy of "reconfigurable interconnects" that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations

FULL CUSTOM DESIGN	SEMI CUSTOM DESIGN
All mask layers are customized in full custom design	It uses pre-designed logic cell(and gates, OR gate, multiplexers) known as standard cells.
In full custom design, all logic cells, circuits or layouts are designed specifically. Design doesn't use pretested or pre-characterized cells.	Designer used pre-tested or pre-characterized cell.
This approach is considered only when there is no suitable existing	Widely used
Offers high performance lower cost as compared to semicustom	More cost. Low performance.
Design time and complexity is more.	Design time and complexity is less
Example - Microprocessor	Example - Digital logics

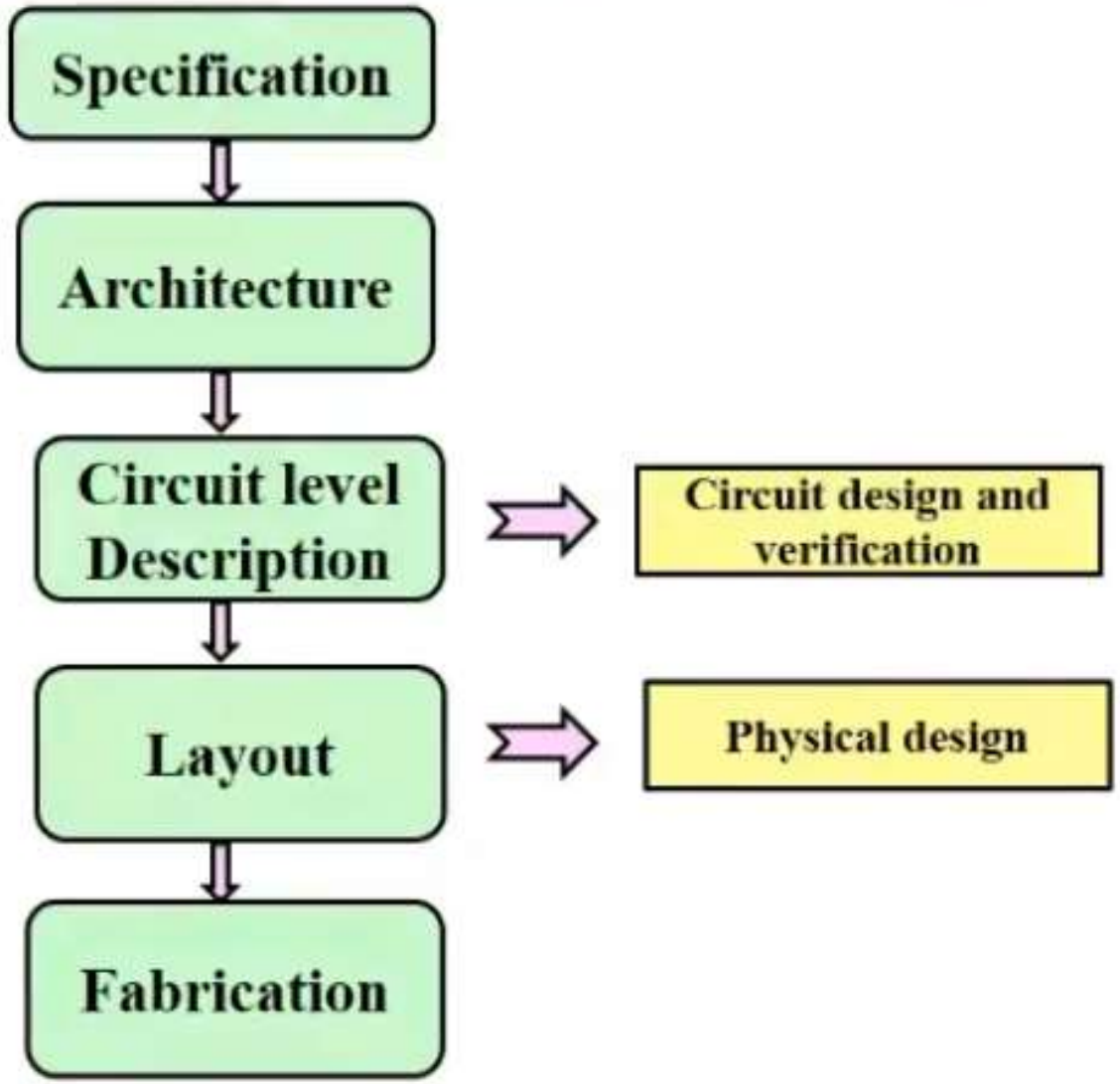
Why Full Custom?

- Full-custom ASIC design defines all the photolithographic layers of the device.
- Full-custom design is used for both ASIC design and for standard product design.
- The benefits of full-custom design include reduced area, performance improvements.
- Ability to integrate analog components and other pre-design and fully verified components, such as microprocessor cores, that form a system on a chip.
- The **disadvantages** of full-custom design can include increased manufacturing and design time, increased non-recurring engineering costs, more complexity in the computer-aided design (CAD) and electronic design automation systems, and a much higher skill requirement on the part of the design team.



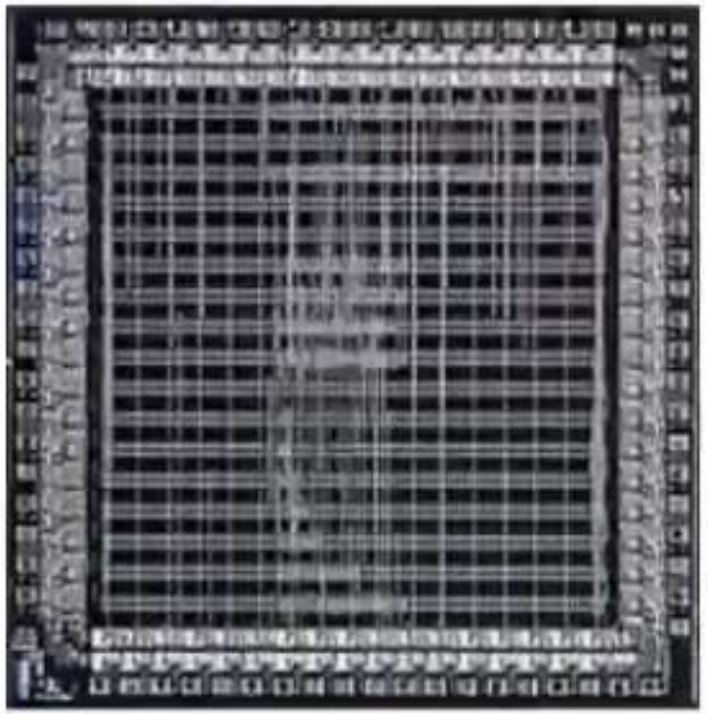
Source:Wikipedia.org

Full Custom or Analog Flow



Why FPGAs?

- Field-programmable gate arrays (FPGA) are the modern-day technology for building a prototype.
- It uses standard parts like programmable logic blocks and programmable interconnects allow the same FPGA to be used in many different applications.
- For smaller designs or lower production volumes, FPGAs may be more cost effective than an ASIC design, even in production.
- Device manufacturers typically prefer FPGAs for prototyping and devices with low production volume
- ASICs for very large production volumes where NRE costs can be amortized across many devices.

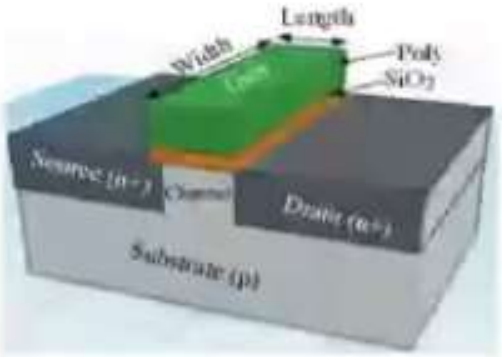


Why ASIC?

- An **application-specific integrated circuit** (**ASIC**) is an integrated circuit (IC) chip customized for a particular use, rather than intended for general-purpose use.
- For example, a chip designed to run in a digital voice recorder or a high-efficiency device is an ASIC.
- ASIC chips are typically fabricated using metal-oxide-semiconductor (MOS) technology, as MOS integrated circuit chips.
- The non-recurring engineering (NRE) cost of an ASIC can run into the millions of dollars.

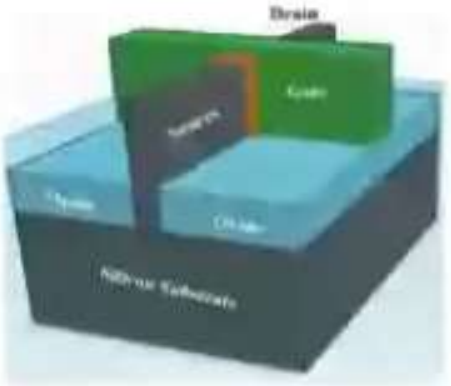
CMOS Vs FinFET Vs CNFET Transist

CMOS



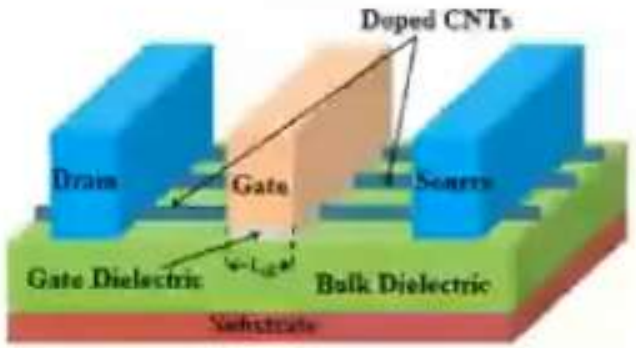
- Traditional 2-D planar implementation of MOSFET, with the conducting channel in the silicon region under the gate electrode in "ON" state
- MOSFET was the mass element in any IC at micro size, but when scaled down to nanometer range, performance degrades because of short channel effects.
- Used up to 28nm Technology

FinFET



- The fin field-effect transistor (FinFET) is a nonplanar, nanoscale three-dimensional (3D)-implementation of the MOSFET, with the conducting channel elevated so the gate can surround it on three sides
- Higher switching speeds and lower power consumption due to lower equivalent input capacitance and channel quantization effects
- Higher output current per input voltage.
- Better on/off contrast due to channel quantization effects.
- Used for 14nm, 10nm and 7nm Technology

CNFET



- CNFET is carbon nanotube (CN) field-effect transistor (FET) that utilizes a single carbon nanotube or an array of carbon nanotubes as the channel material
- The carrier velocity and trans-conductance of CNTs are an order of magnitude higher compared to FinFETs.
- Used for 7nm Technology and below

NOTE: There are other technologies like Gate All Around FET, Gallium-Arsedine, Graphene based Tetra-Hz transistors coming up for

CMOS process
- 10 µm – 1971
- 6 µm – 1974
- 3 µm – 1977
- 1.5 µm – 1981
- 1 µm – 1984
- 800 nm – 1987
- 600 nm – 1990
- 350 nm – 1993
- 250 nm – 1996
- 180 nm – 1999
- 130 nm – 2001
- 90 nm – 2003
- 65 nm – 2005
- 45 nm – 2007
- 32 nm – 2009
- 22 nm – 2012

FINFET process
- 14 nm – 2014
- 10 nm – 2016
- 7 nm – 2018

CNFET Process
- 5 nm – ~2020

Future
- 3 nm – ~2021
- 2 nm – ~2024

ASIC Design Flow

Design

Verification

Front-End

Back-End

Synth/STA

DFT

PD

Verification

PD/PV

Architecture

Design, RTL Coding

Functional Verification

Synthesis and STA

DFT

Physical Design

Gate Level Simulation

Physical Verification

GDS-II

Architecture, Micro Design

RTL Development, Design for low power

Test bench, Test cases, Simulation

Synthesis

Static Timing Analysis

Gate-Level power estimation

SCAN, ATPG, MBIST, LBIST

Floor plan, power plan

Placement, Clock Tree Synthesis

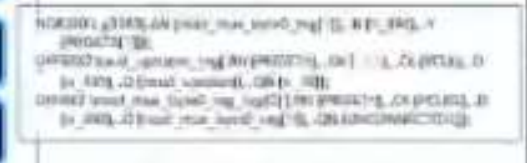
Routing

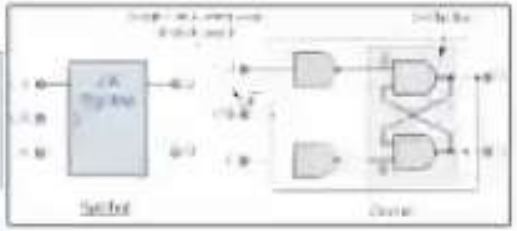
Test bench, Test cases, Simulation

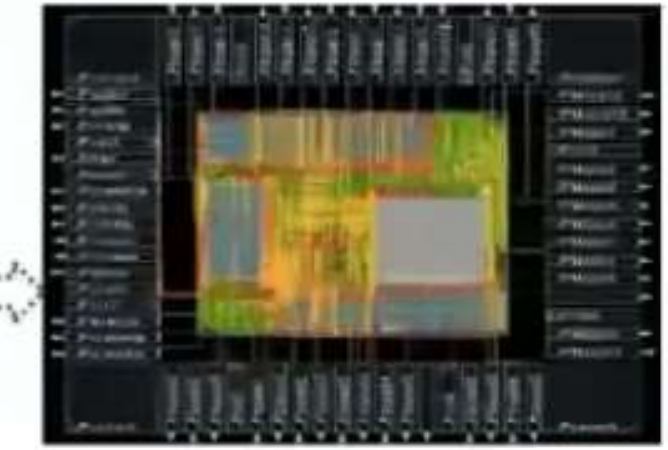
Physical Verification

Signoff Power Analysis

Tape out

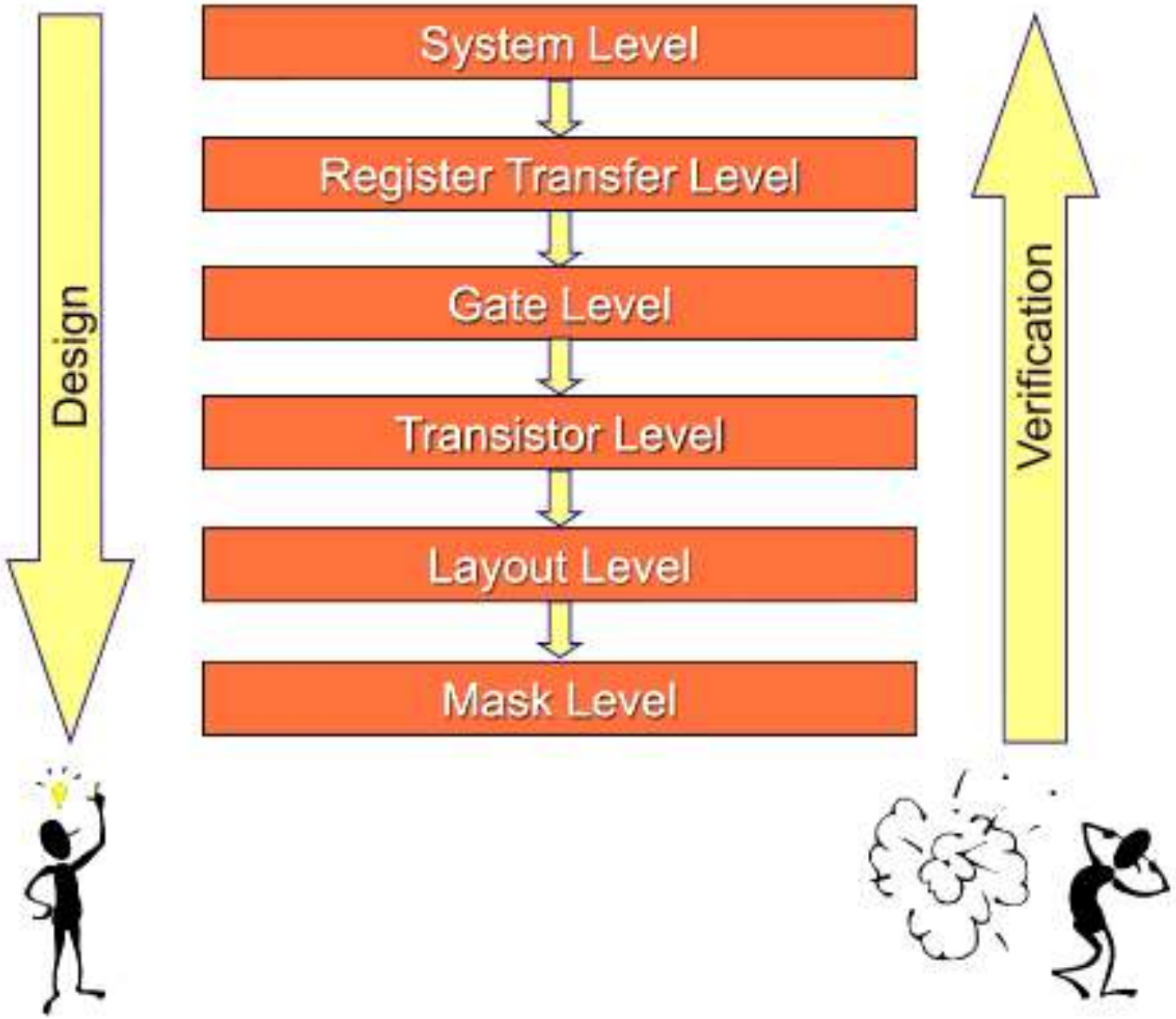






GDS (Graphic Data Stream) is a file that was developed by calma company in the year 1971 and the **GDS** II in the year 1978. It is a binary file format that represents layout data in a hierarchical format. Data such as labels, shapes, layer information and other 2D and 3D layout geometric data.

Basic Design Abstraction



Another view:

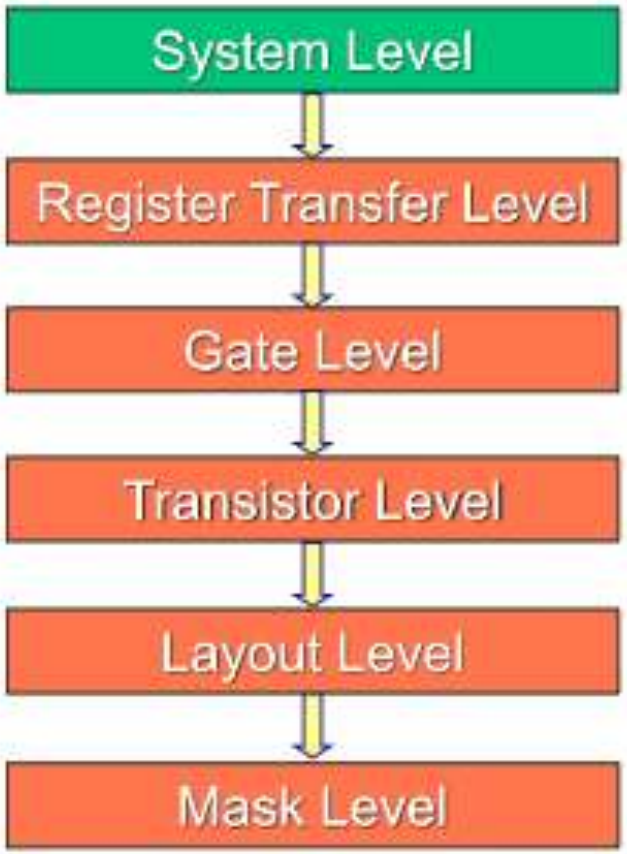
- Application
- Algorithm
- Programming Language
- OS / Virtual Machine
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Circuits
- Devices
- Physics

System Level Abstraction

- **Abstract algorithmic description of high-level behavior**
  - e.g., C-Programming language

```
Port* compute_optimal_route_for_packet
          (Packet_t *packet, Channel_t *channel)
   {
      static Queue_t *packet_queue;
      packet_queue = add_packet(packet_queue, packet);
      ...
   }
```

  - Abstract because it does not contain any implementation details for timing or data
  - Efficient to get a compact execution model as a first design draft
  - Difficult to maintain throughout project because no link to implementation



Register-Transfer Level (RTL)

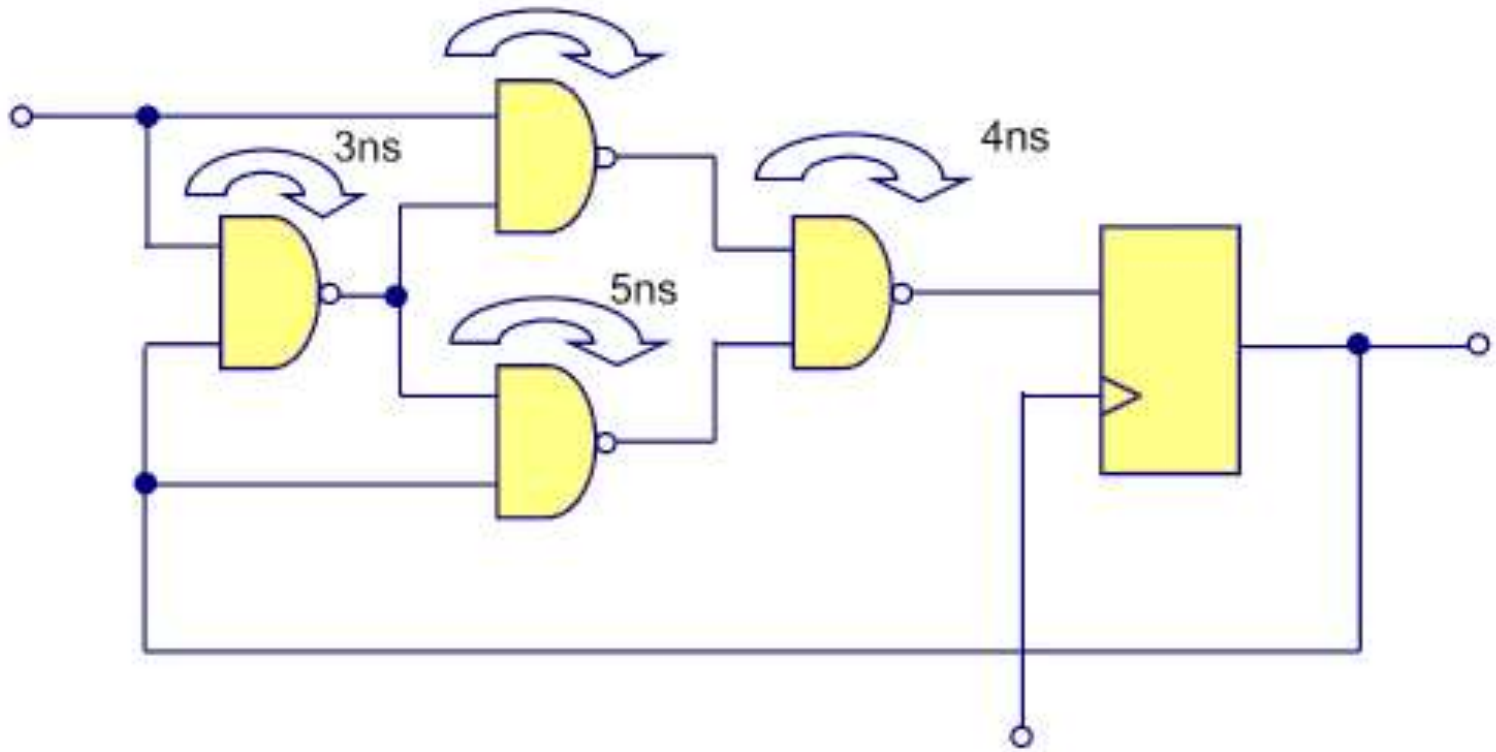
- **Cycle accurate model "close" to the hardware implementation**
  - bit-vector data types and operations as abstraction from bit-level implementation
  - sequential constructs (e.g., `if-then-else`, `while` loops) to support modeling of complex control flow

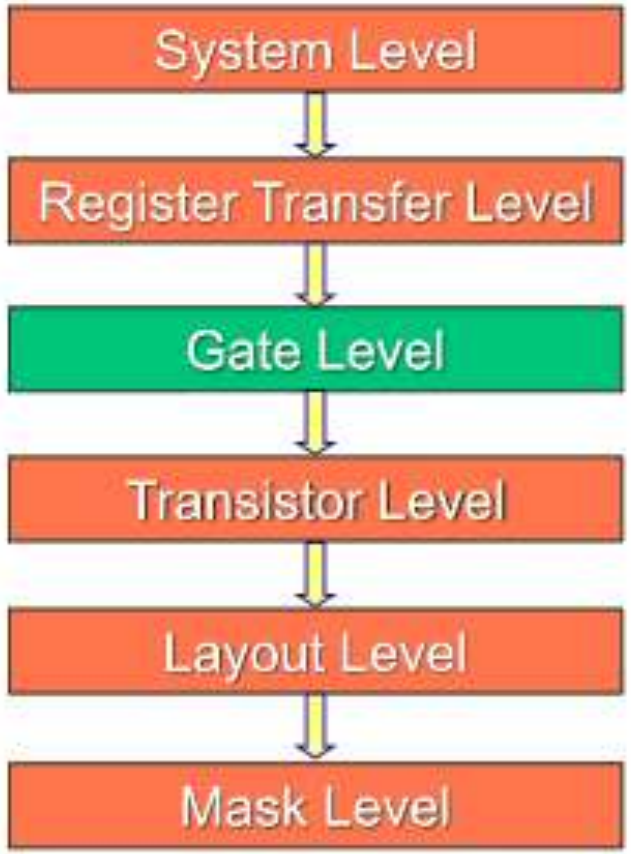
```
module mark1;
  reg [31:0] m[0:8192];
  reg [12:0] pc;
  reg [31:0] acc;
  reg[15:0] ir;

  always
   begin
     ir = m[pc];
     if(ir[15:13] == 3b'000)
        pc = m[ir[12:0]];
     else if (ir[15:13] == 3'b010)
        acc = -m[ir[12:0]];
     ...
   end
endmodule
```

Gate Level Abstraction (GTL)

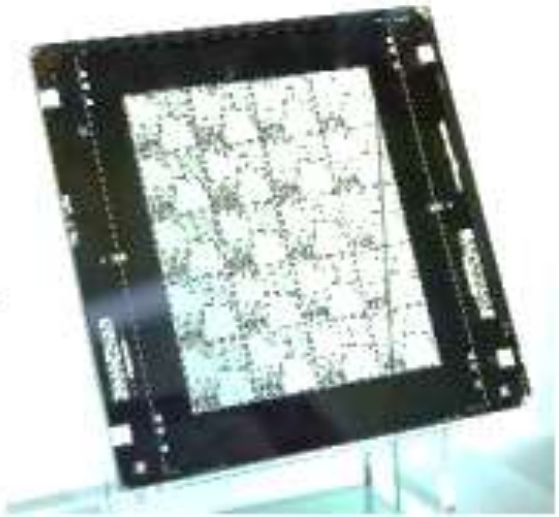
- **Model on finite-state machine level**
  - Models function in Boolean logic using registers and gates
  - Various delay models for gates and wires

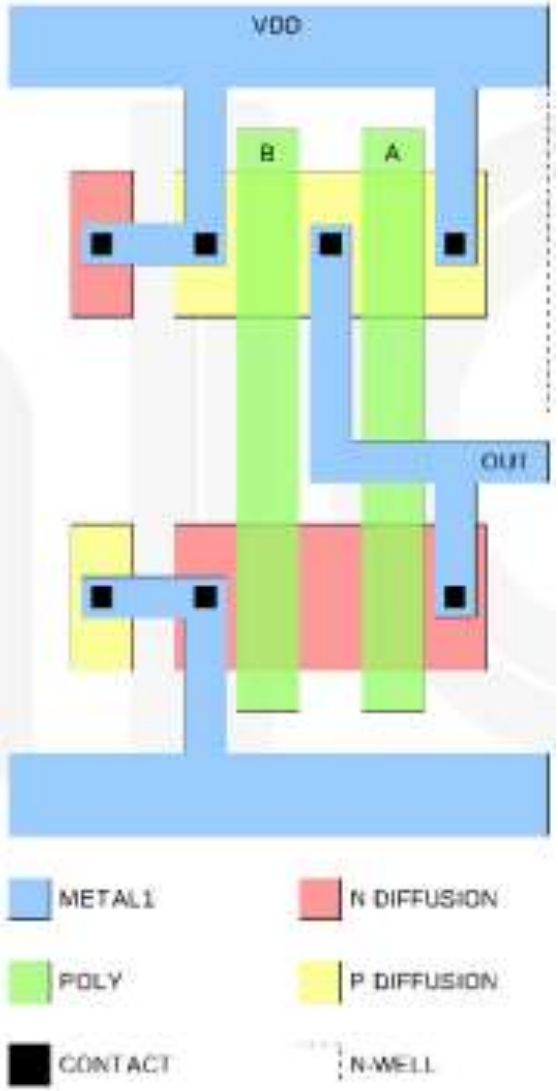


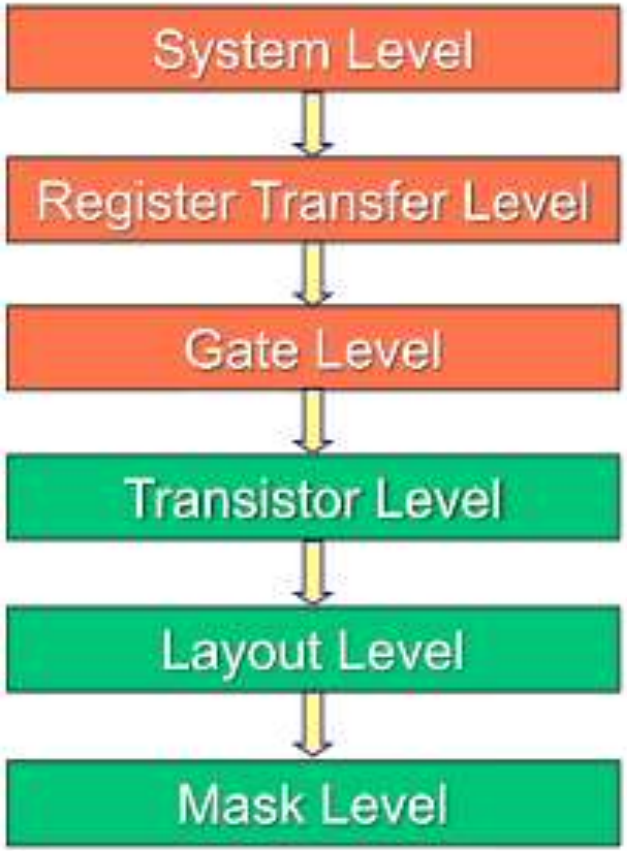


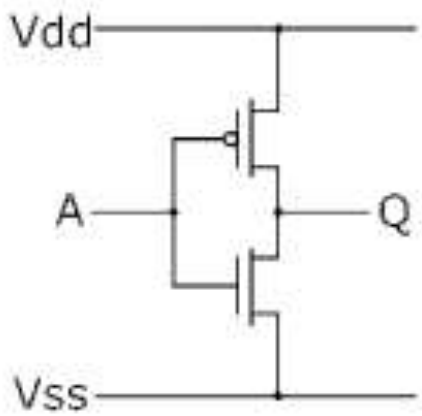
Transistor to Mask Level

- **As we've seen in previous courses:**
  - Transistor Level:
    - Use *compact models* to enable accurate circuit simulation.
  - Layout Level:
    - Draw polygons to implement the devices and interconnect.
  - Mask Level:
    - Create actual photo-masks for performing lithography during fabrication process.









The Chip Hall of Fame

- **To get started, let's remember the CPU that started it all**

The Intel 4004 Microprocessor

- The first commercially available monotlithic CPU.
- Release date: March 1971
- Transistor Count: 2,300 Process: 10 um pMOS
- Frequency: 740 KHz 4-bit data bus
- Designed as a side project to drum up some cash, while Intel developed its *real* product line, memory chips.
- Developed as part of a 4-chip product line (MCS-4 chipset) for the Busicom calculator.

The Busicom Calculator

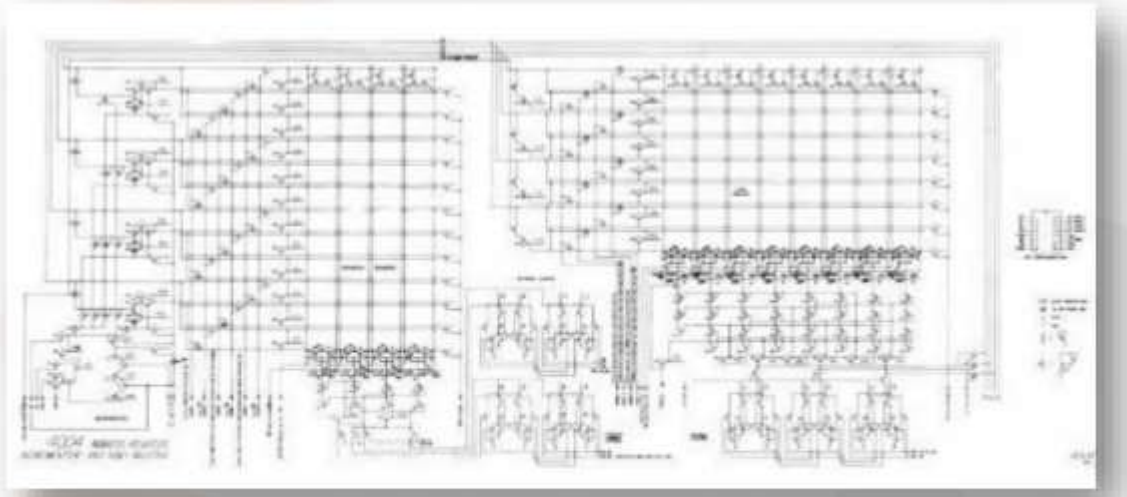
Federico Faggin and the 4004 layout

2018 Inductee to the IEEE Chip Hall of Fame

Design Automation

The (really) Olden Days

Early chips were prepared entirely by hand:



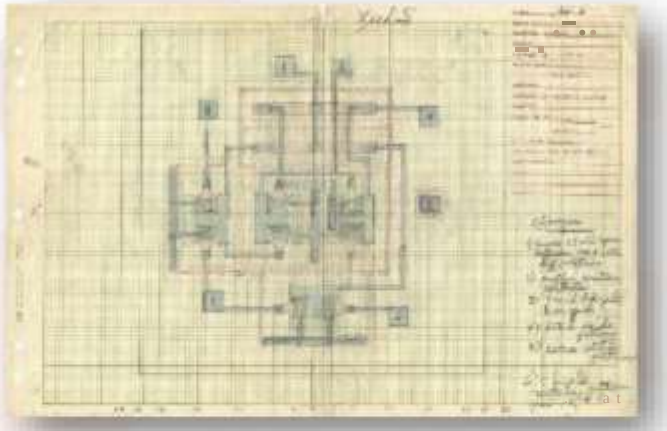
Schematic of Intel 4004 (1971)

Mainframe CAD System

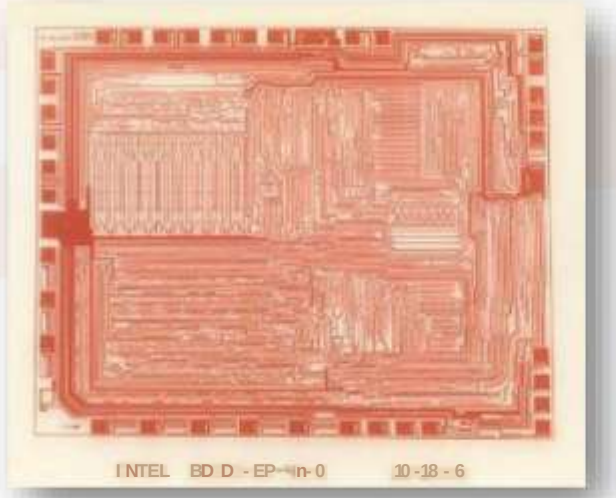
http://www.computerhistory.org/revolution/digital-logic

The (really) Olden Days

- **Early chips were prepared entirely by hand:**

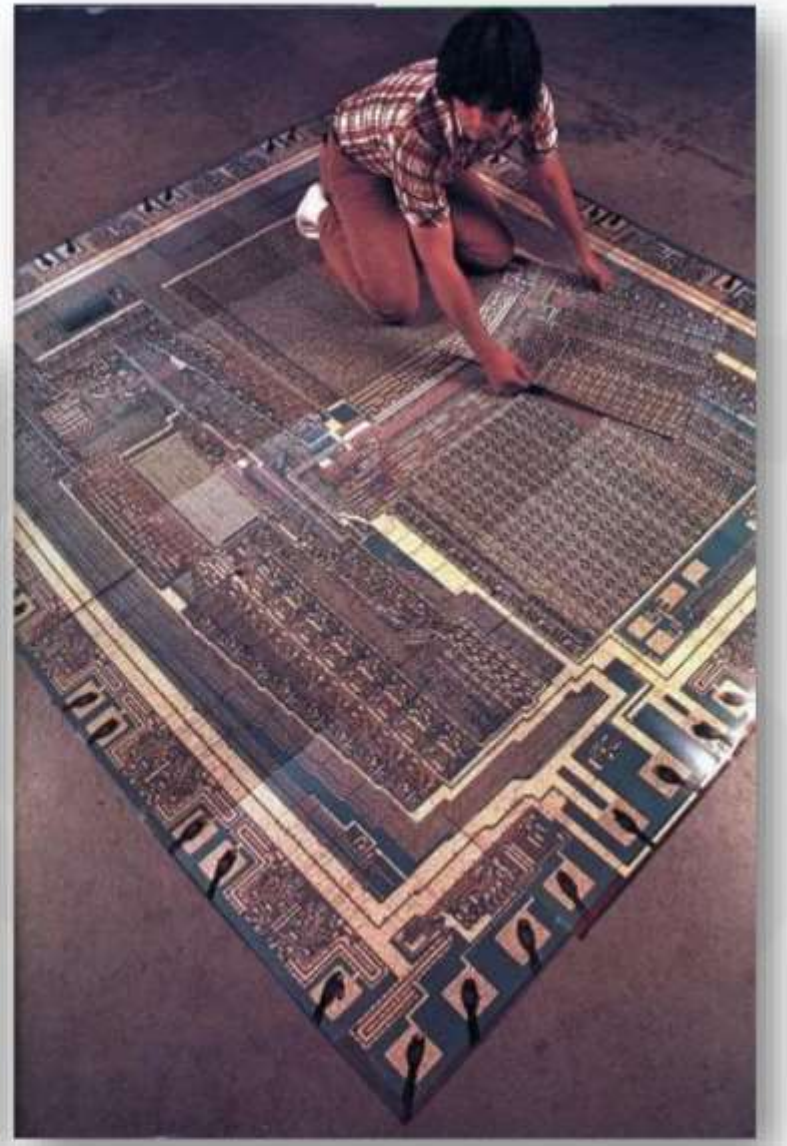


Hand drawn gate layout (Fairchild)



8088A Mask Transparent Overlays (1976)

Rubylith Operators (1970)



The original Tape-Out?

http://www.computerhistory.org/revolution/digital-logic

Design Automation Today

Design:
- High-Level Synthesis
- Logic Synthesis
- Schematic Capture
- Layout
- PCB Design

Simulation:
- Transistor Simulation
- Logic Simulation
- Hardware Emulation
- Technology CAD
- Field Solvers

Validation:
- ATPG
- BIST

Analysis and Verification:
- Functional Verification
- Clock Domain Crossing
- Formal Verification
- Equivalence Checking
- Static Timing Analysis
- Physical Verification

Mask Preparation:
- Optical Proximity Correction (OPC)
- Resolution Enhancement Techniques
- Mask Generation

EDA tools

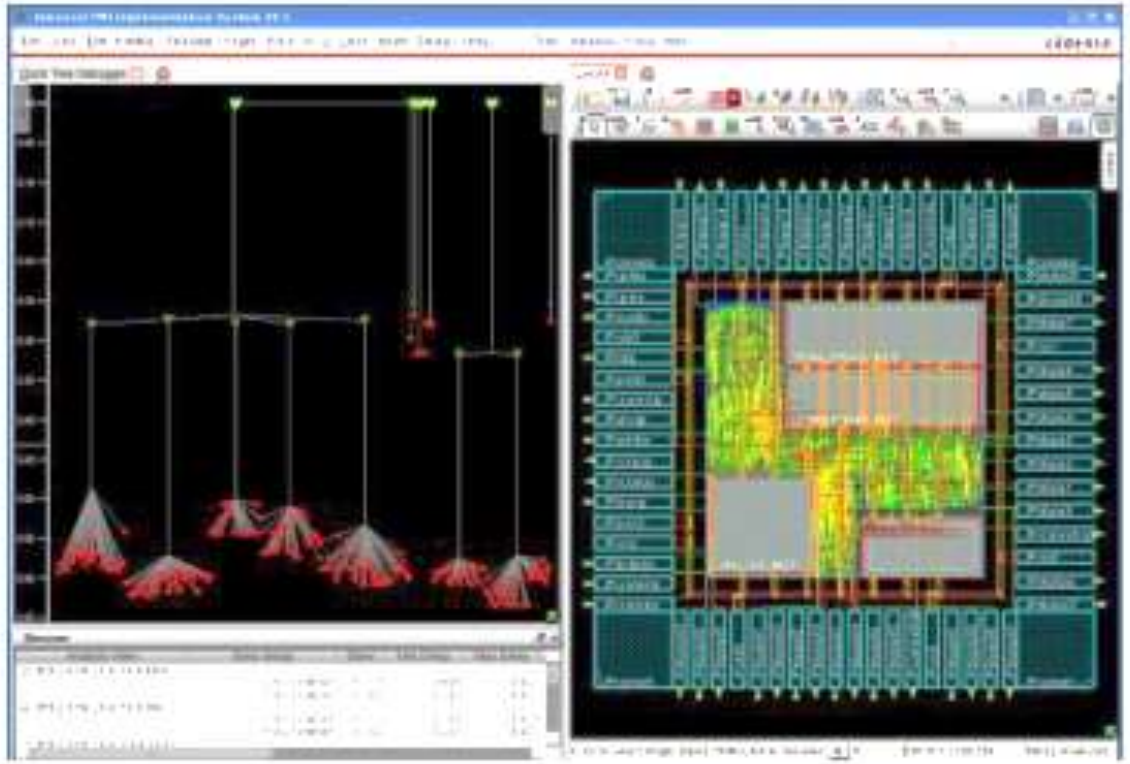
- **RTL**
  - Verilog

Questasim from Mentor Graphics, VCS from synopsis and IES (ncsim) from Cadence

- **Synthesis**
  - Cadence Genus
- **Place and Route**
  - Cadence Innovus
  - Static Timing Analysis – Tempus
  - Power Estimation – Voltus
  - Parasitic Extraction – QRC
  - Clock Tree Synthesis - CCOpt
- **Logic Simulation**
  - Cadence Incisive

cadence®

ACADEMIC NETWORK

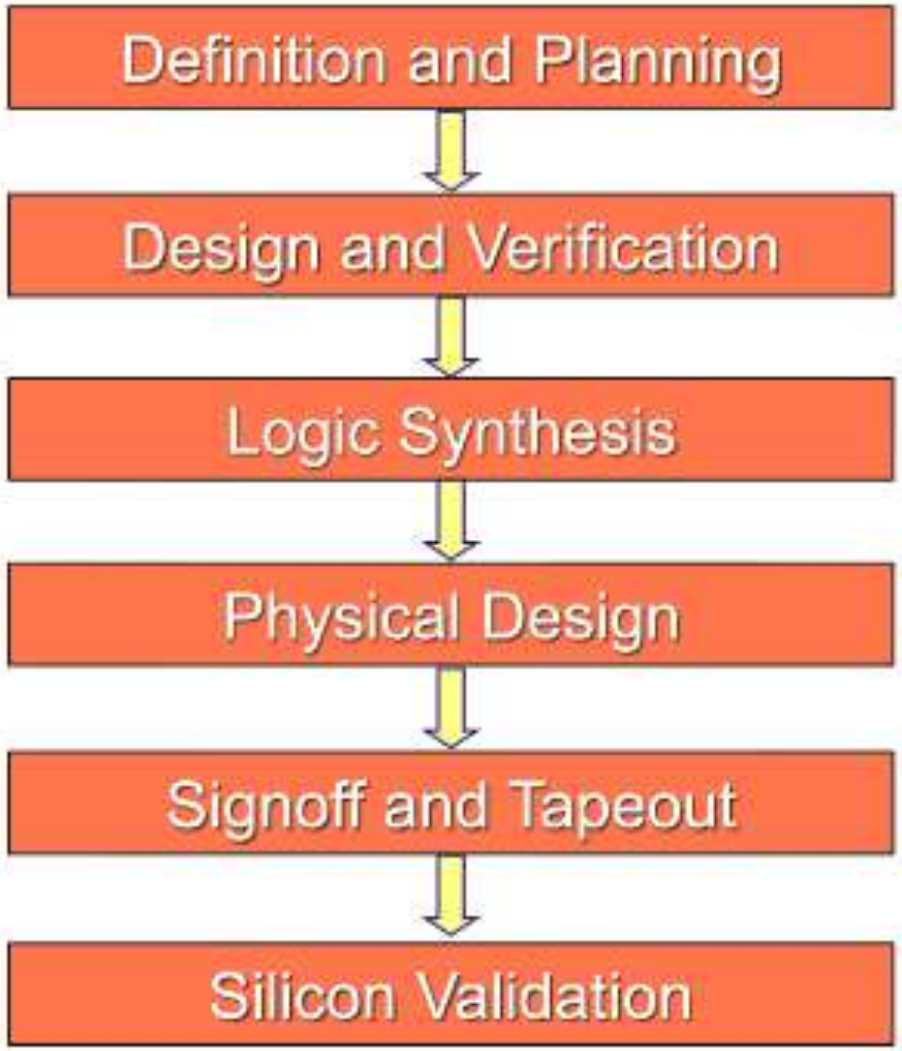


Chip Design Flow

How a chip is built

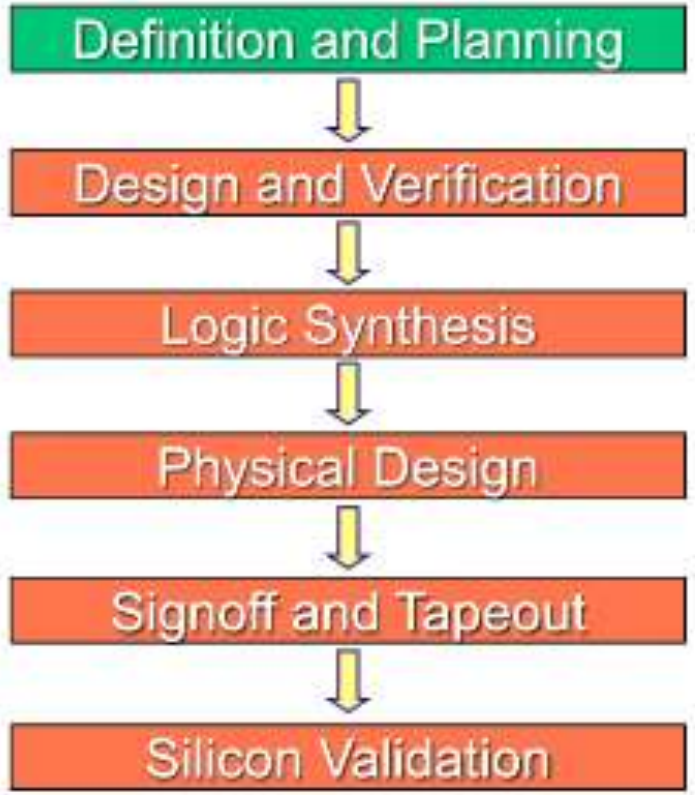
- Definition and Planning
- Design and Verification (Frontend)
- Logic Synthesis (Frontend and Backend)
- Physical Design (Backend)
- Signoff and Tapeout
- Silicon Validation

- Don’t forget package & board design, software design, test plan, etc., etc., etc.



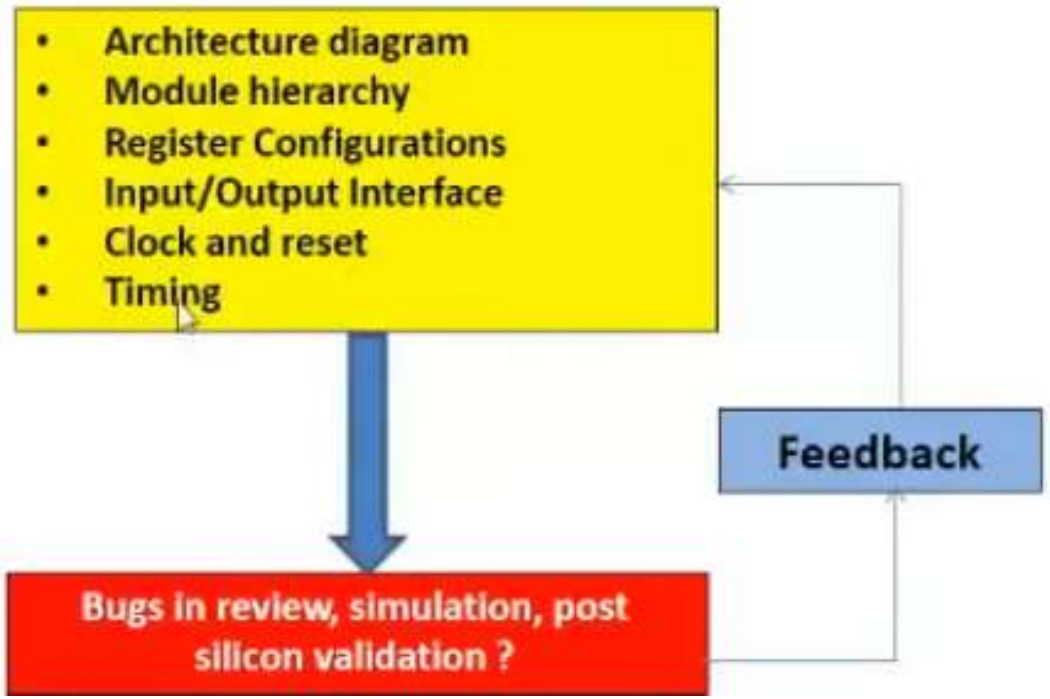
Definition & Planning

- **Marketing Requirements Document (MRD)**
- **Chip Architecture**
  - Define bus structures, connectivity
  - Partition Functionality
  - High-Level System Model (Bandwidths, Power, Freq.)
  - System partitioning (HW vs SW, #Cores, Memories…)
- **Design Documents**
- **Floorplan/Board Requirements**
- **Process and fab**
- **Project kick-off – transfer to implementation**

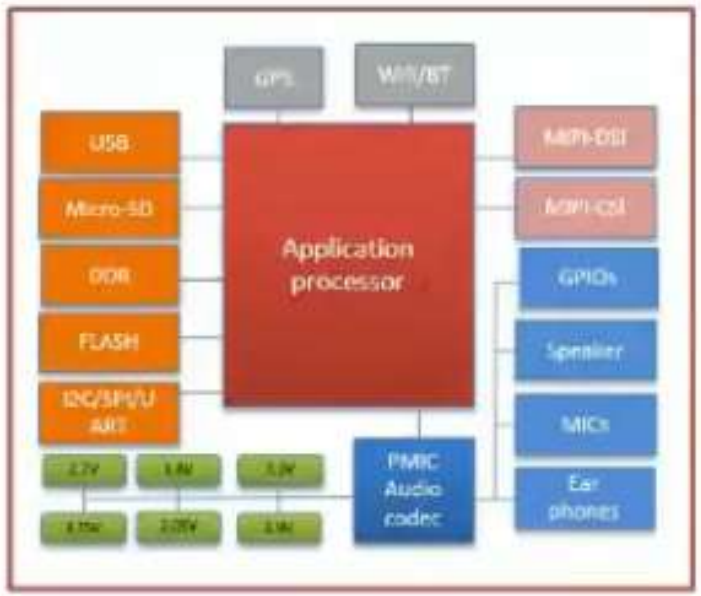


Chip Architecture

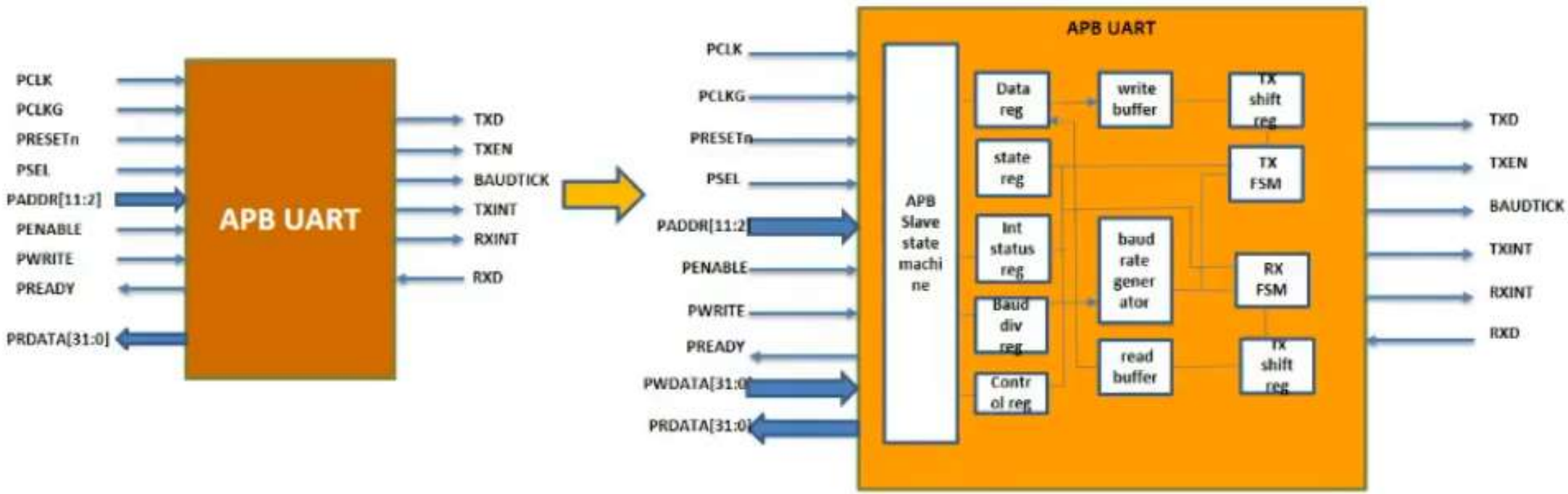
- High Level Chip Architecture Specification includes
  - Functionality/Application
  - Performance
  - Power
  - Area
  - Schedule



- Chip Architecture consists of various blocks/IPs instantiated in the system

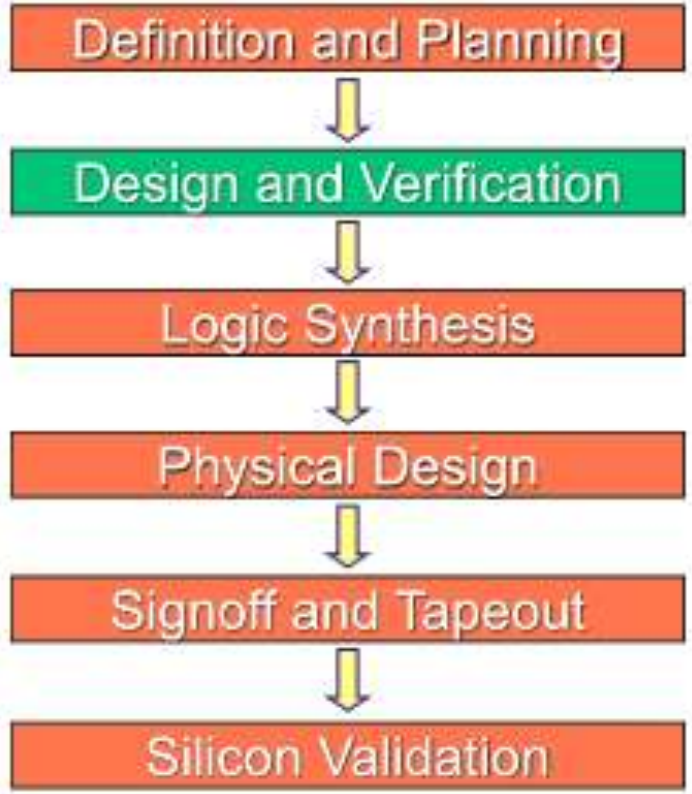


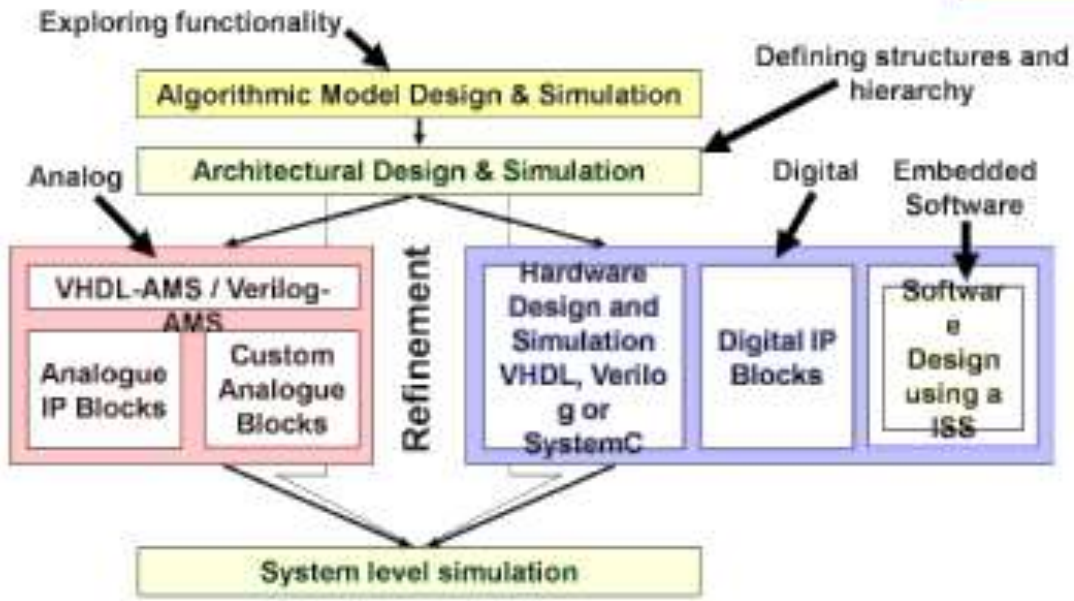
Design


PCLK
PCLKG
PRESETn
PSEL
PADDR[11:2]
PENABLE
PWRITE
PREADY
PRDATA[31:0]
APB UART
TXD
TXEN
BAUDTICK
TXINT
RXINT
RXD
APB UART
APB Slave state machine
Data reg
write buffer
TX shift reg
state reg
TX FSM
Int status reg
baud rate generator
RX FSM
Baud div reg
read buffer
TX shift reg
Control reg
PWDATA[31:0]

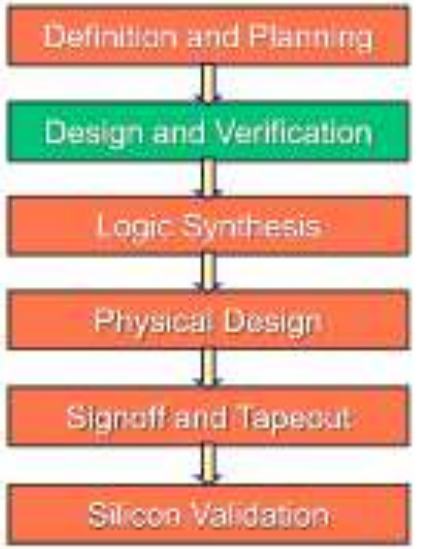
Design and Verification

- RTL (Register Transfer Level) Design
- Integration/Development of IPs
- RTL Lint/Synthesability checks
- Formal Verification
- Functional verification all the IPs:
  - Unit level
  - Sub-system level
  - Chip (SOC) level





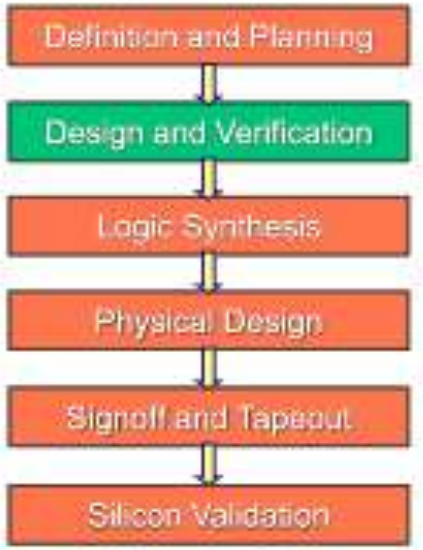
Design and Verification - IP Integration

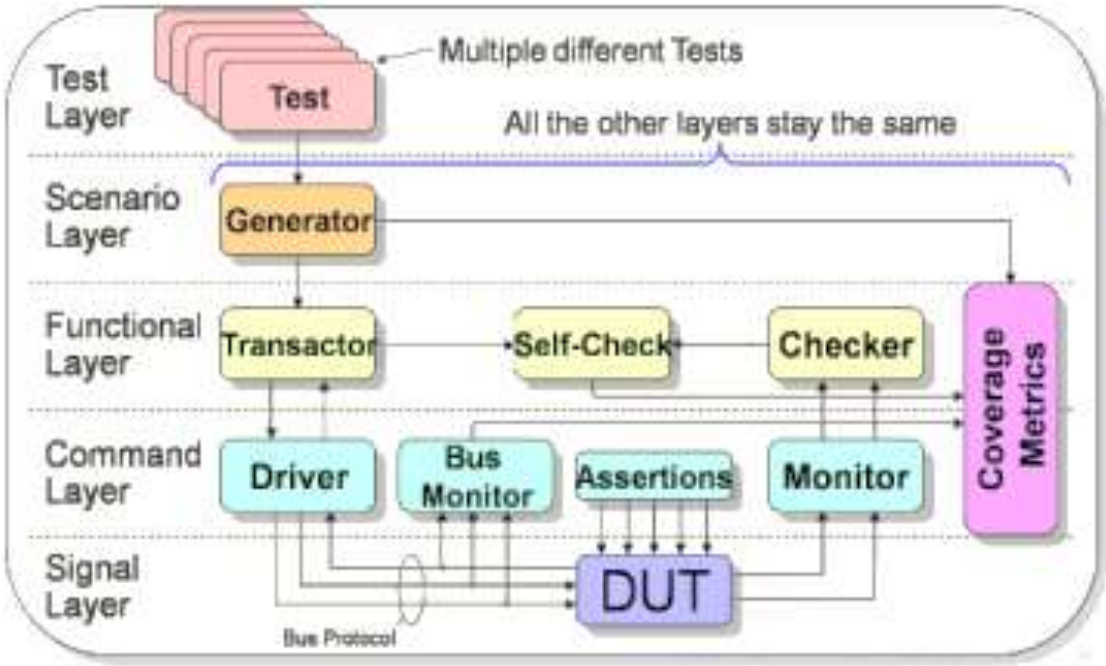


- **Hard IP**
  - IP provided as pre-existing layouts with:
    - Timing models
    - Layout abstracts
    - Behavioral models (Verilog/VHDL)
    - Sometimes with Spice models, full-layouts
  - This is the standard delivery format for custom digital blocks
    - RAMs, ROMs, PLLs, Processors
- **Soft IP**
  - RTL Code
    - Can be encrypted
    - Instantiated just like any other RTL block
  - Sometimes with behavioral models

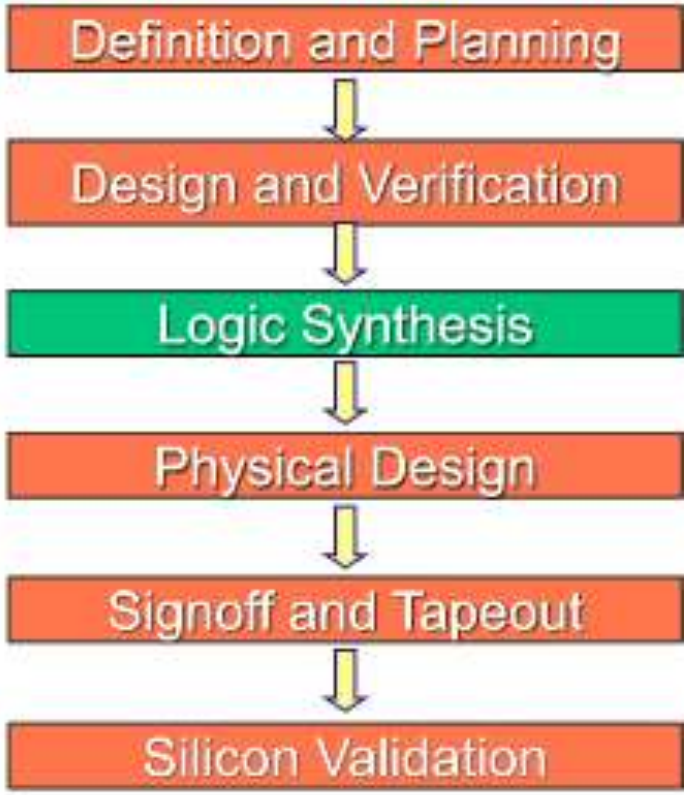
Design and Verification - Prototyping

- **Different levels of verification:**
  - Specification driven testing
  - Bug driven testing
  - Coverage driven testing
  - Regression
- **FPGA Prototyping:**
  - Synthesize to FPGA
  - Speeds up testing where possible.
- **Hardware Emulation:**
  - Big servers that can emulate the entire design.





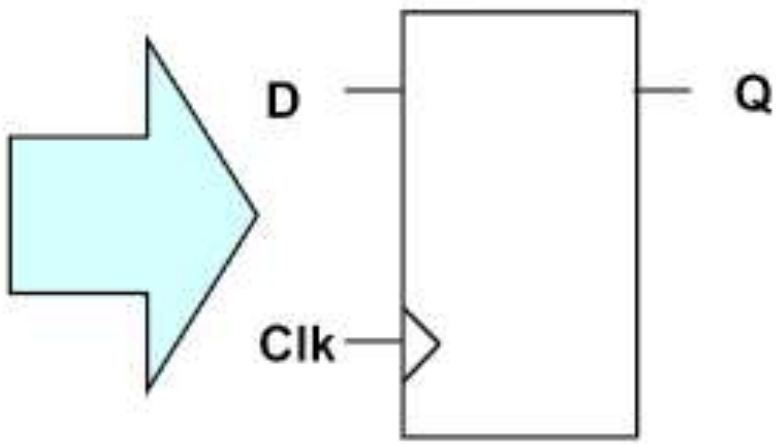
Logic Synthesis



- **Inputs:**
  - Technology library file
  - RTL files
  - Constraint files (SDC)
  - DFT definitions
- **Output:**
  - Gate-level netlist

- **Synthesis**
  - Converting RTL code into a generic logic netlist
- **Mapping**
  - Mapping generic netlist into standard cells from the core library
- **Optimization**
  - To meet Timing / Area / Power constraints

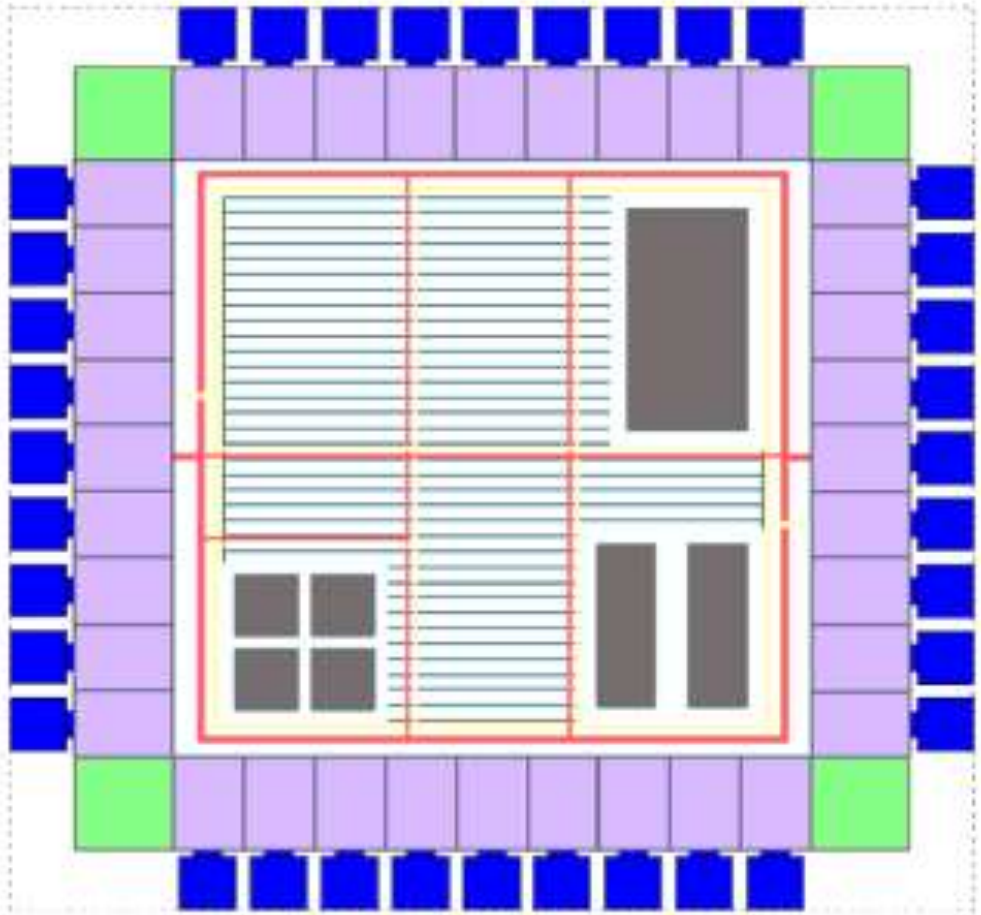
```
module DFF(Clk, D, Q);
  input Clk;
  input D;
  output Q;
  always @(posedge Clk)
    Q <= D;
endmodule
```

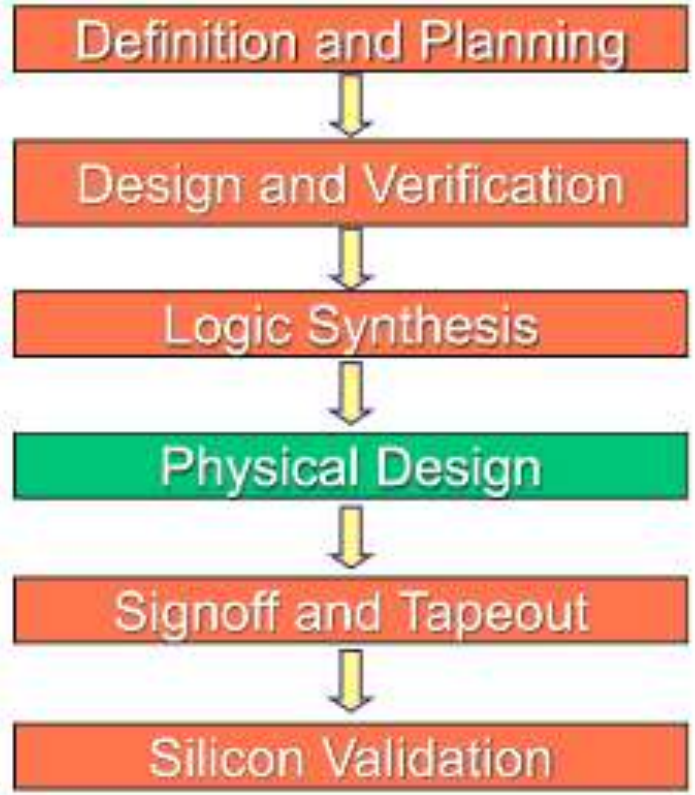


- **Post Synthesis checks**
  - Gate-level simulation
  - Formal verification (Logic Equivalence)
  - Static Timing Analysis (STA)
  - Power/Area estimation

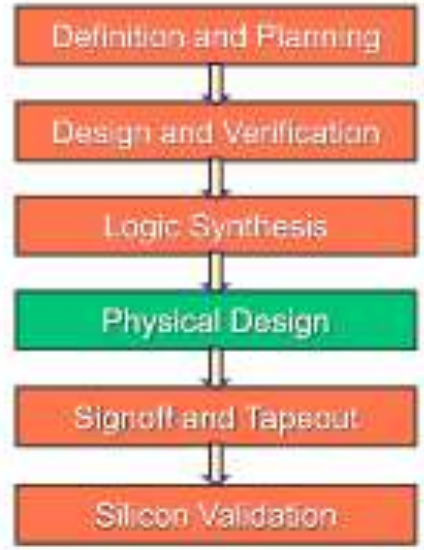
Physical Design (Backend)

- Floorplan
- I/O Ring
- Power Plan
- Placement
- Clock Tree Synthesis
- Route
- DRC, LVS, Antennas, EM
- LEC, Post-layout





Physical Design – Backend Flow



- **Physical Implementation Inputs**

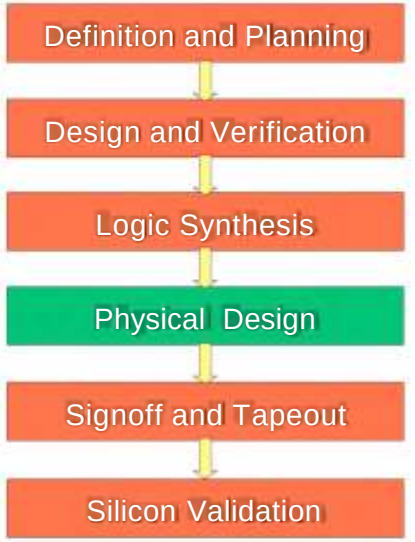
Front-End
- Spec
- Architecture
- RTL
- Verification

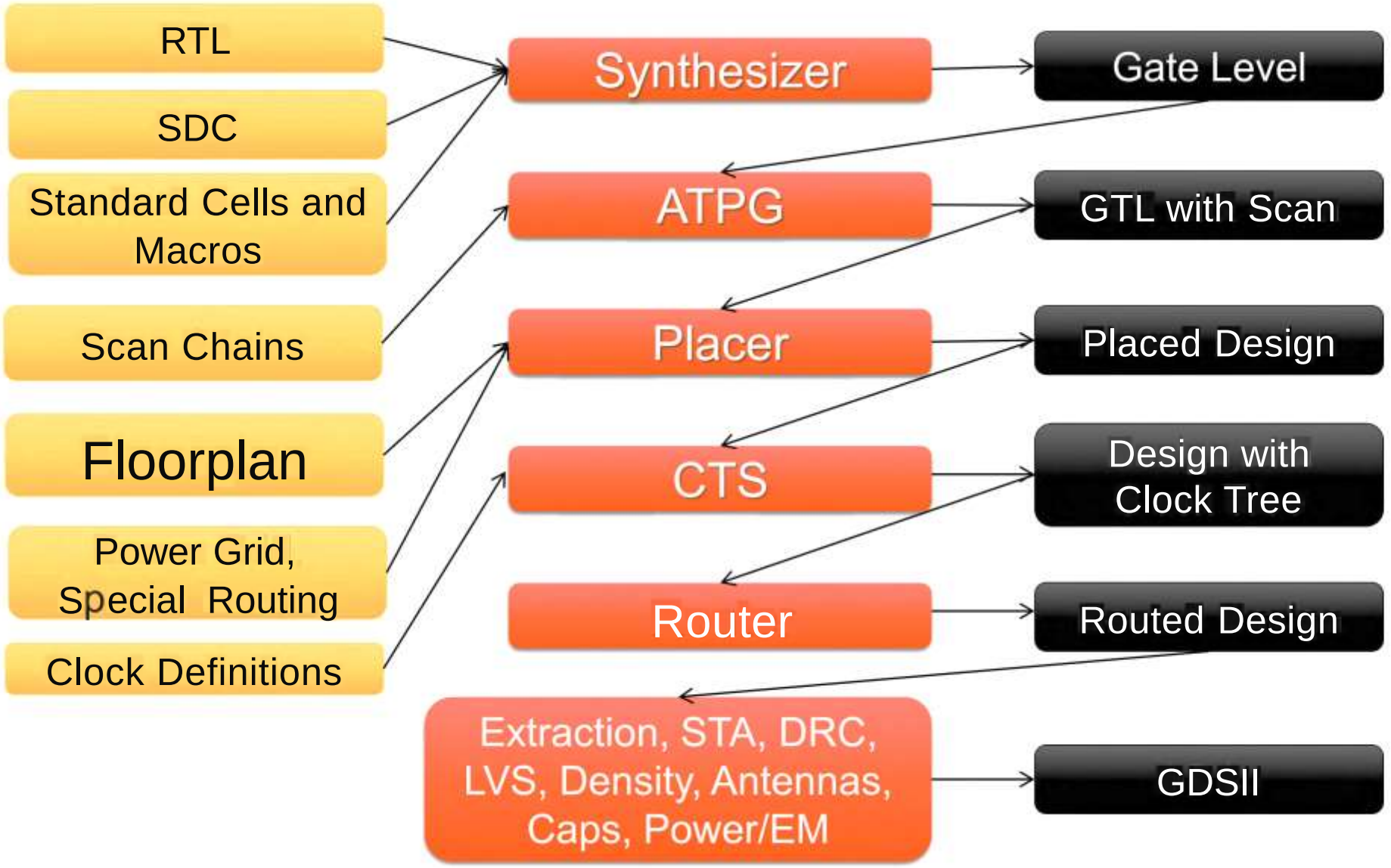
Vendors
- Standard Cells
- Memory Compiler
- I/Os
- Hard IPs

Foundry
- Device Models
- Techfile
- Design Rules

Physical Design (Backend)

Physical Design - Backend Flow





Signoff and Tapeout

- Parasitic Extraction
- STA with SI
- DRC/LVS/ERC/DFM
- Post-layout Gate-level Simulation
- Power Analysis
- DFT
- Logic Equivalence