EEC 116 Lecture #11: Implementation Strategies

Rajeevan Amirtharajah Bevan Baas
Zhiyi Yu
University of California, Davis

Announcements

- **Lab 5 due Wednesday, Nov. 23**
- **Homework 5 due Monday, Nov. 28**
- **Lab 6 due Friday, Dec. 2**

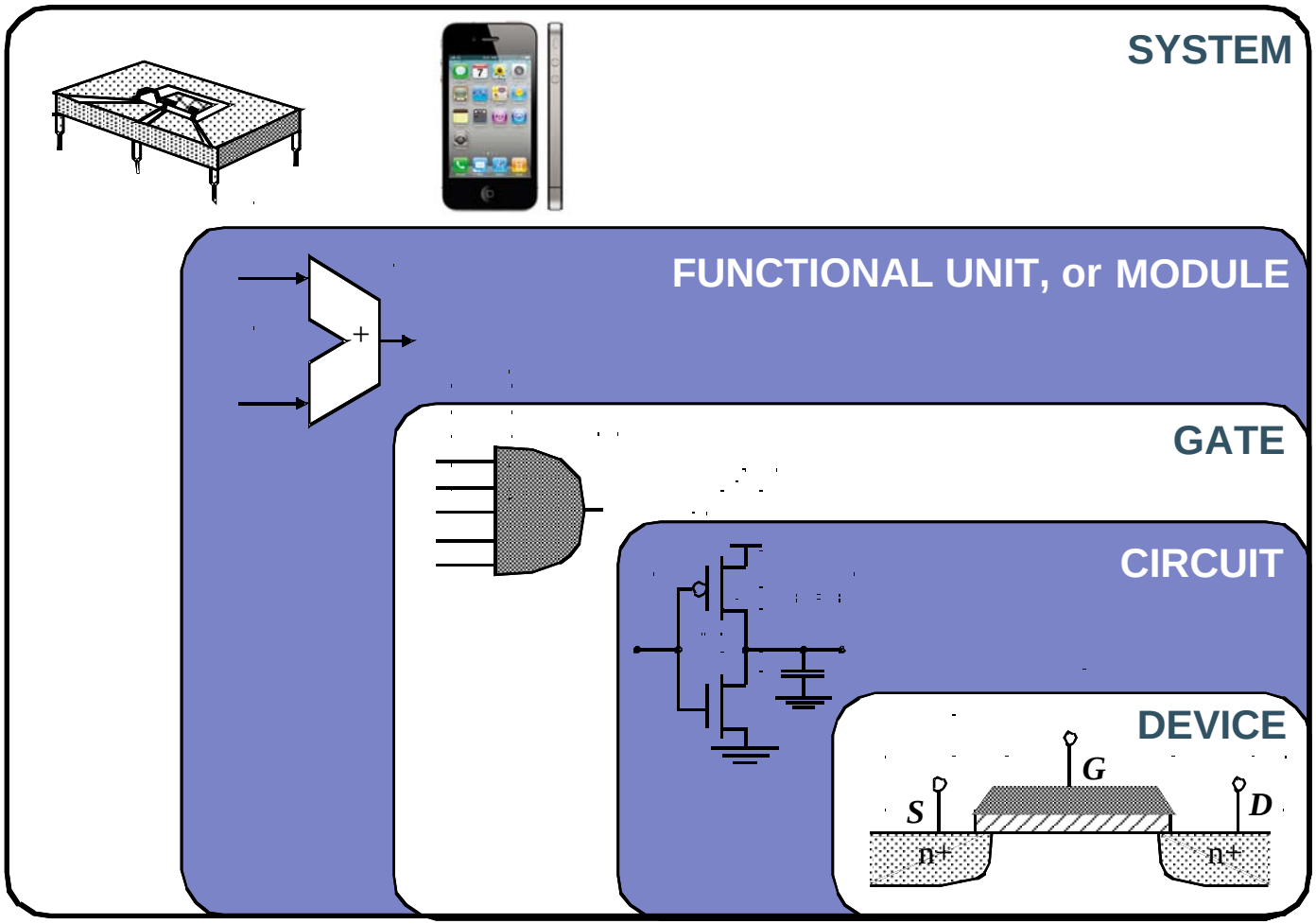
Outline

- **Review and Finish: Memories**
- **Implementation Strategies: Rabaey Ch. 1, 8 (Kang & Leblebici, Ch. 1)**

Abstraction of Design Complexity

- **Design complexity**
  - Typically tens of transistors in analog circuits
    - Each is normally hand crafted along with placement and wiring
  - Hundreds of transistors
    - Each can be hand crafted
  - Thousands to 100s of thousands of transistors
    - Must find regularity in structure and exploit it (re-use cells)
    - Ex: memory
  - Millions to billions of transistors
    - Must find high-level regularity in structure and exploit it (re-use modules and subsystems)
    - Ex: System on Chip (SOC)

Design Abstraction Levels

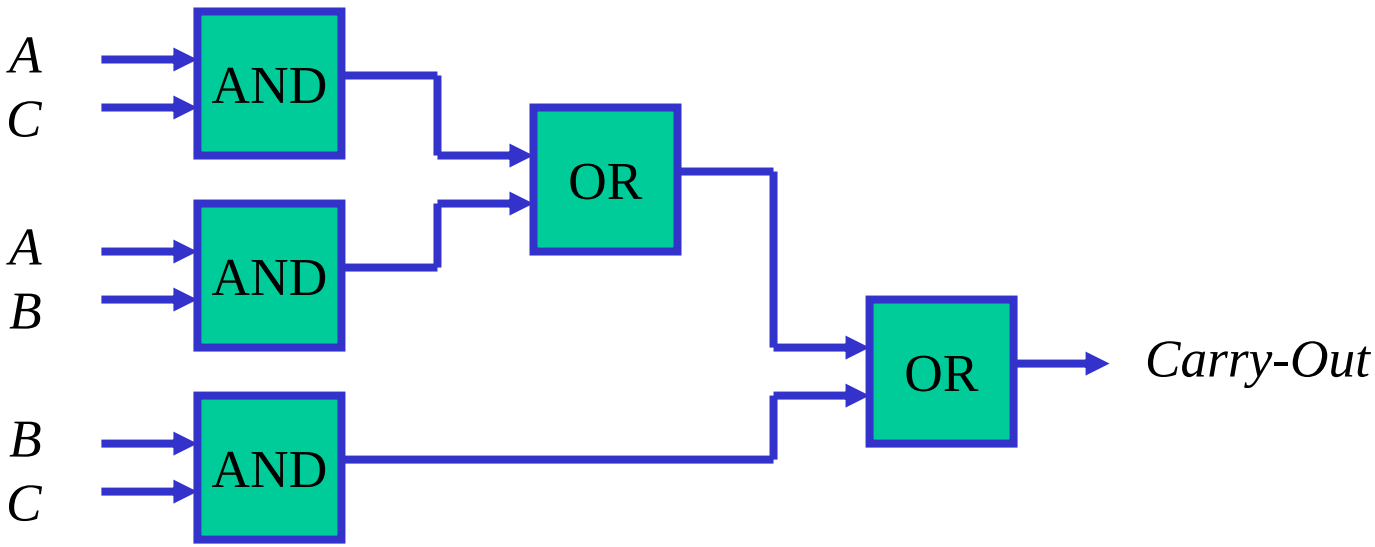


Abstraction of Design Complexity

- **Levels**
  - Device
  - Circuit
  - Gate
  - Module or functional unit (e.g., adder, memory, etc.)
  - Sub-system (e.g., processor, display driver, network interface, etc.)
- **Methods to *abstract* complexity**
  - Sophisticated Computer-Aided-Design (CAD) tools
  - Standard cell libraries

Hierarchical Abstraction

- **Example: While designing at the gate level, we do not consider what is inside each gate**



Why Learn About Circuits and Layout Then?

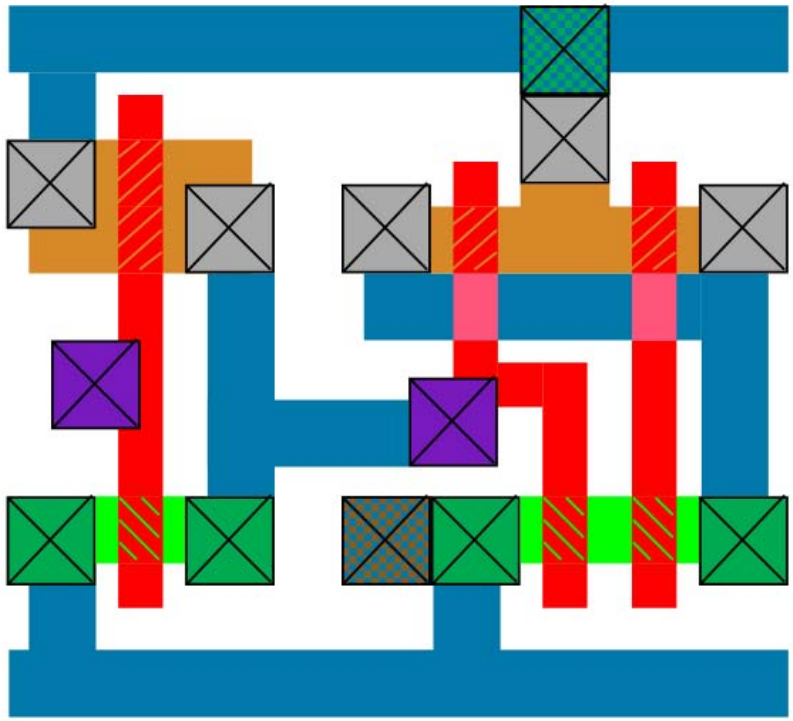
- **Best designers can:**
  - Build model abstractions
  - Understand limitations of models
    - Wire or interconnect performance
    - Changes with technology scaling
- **Abstractions limit maximum attainable performance and energy-efficiency**
  - Multi-disciplinary view needed
- **Troubleshooting and debugging**

Design Aspects that “Defy Hierarchy”

- **Clock distribution**
  - Timing skew
- **Power distribution**
  - Sufficient current handling
  - Adequate noise suppression

Full Custom

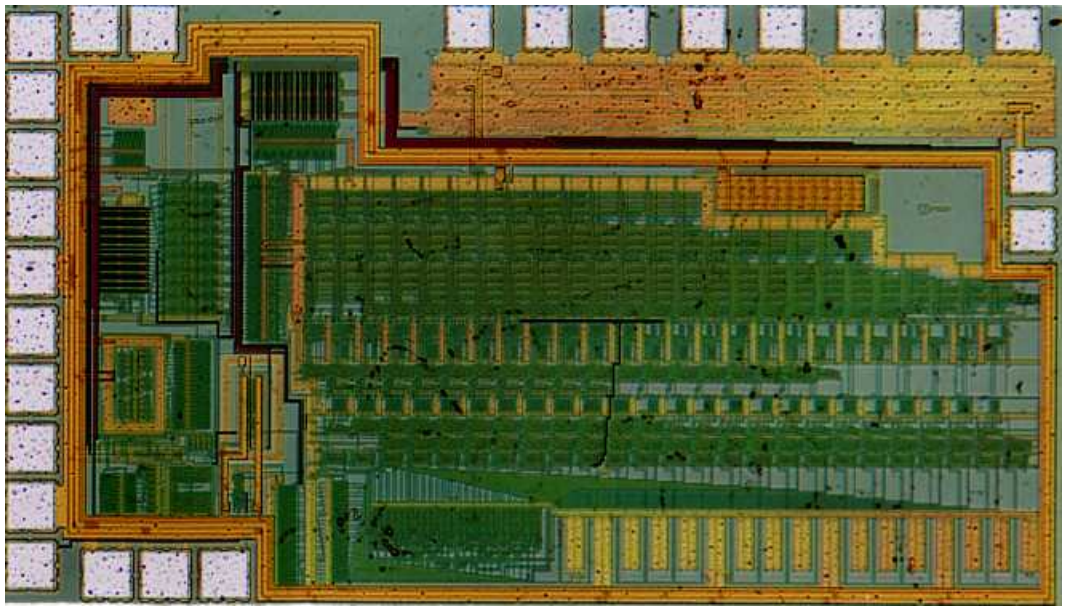
- **All transistors and interconnect drawn by hand**
- **Full control over sizing and layout**
- **Highest area density and higher performance**
- **Longest time to design “maturity”**



[figure from S. Hauck]

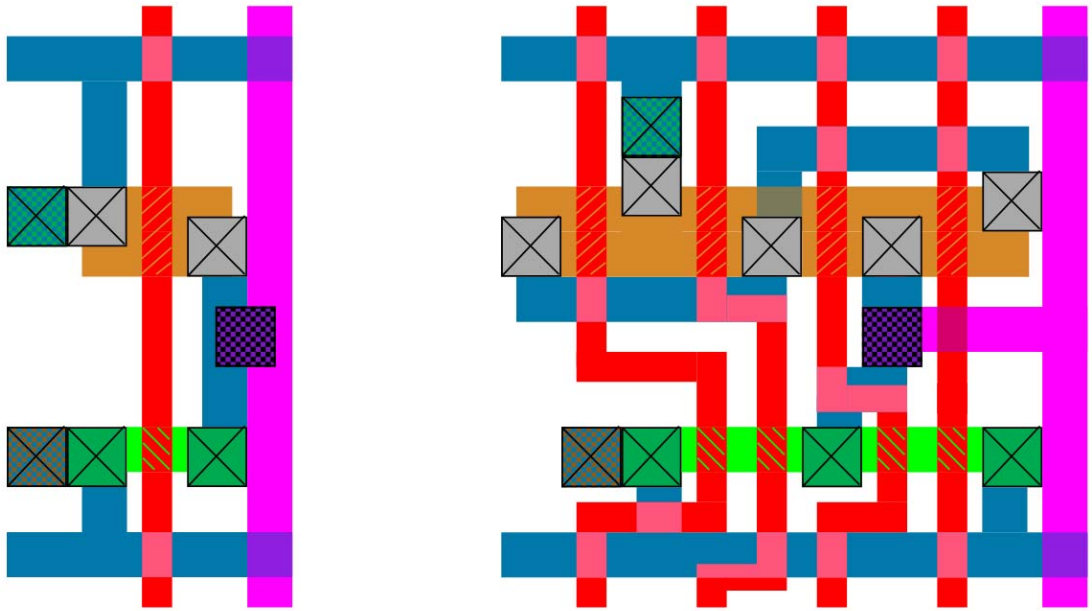
Full Custom Design Example

- **Multiplier Chip**



Standard Cell

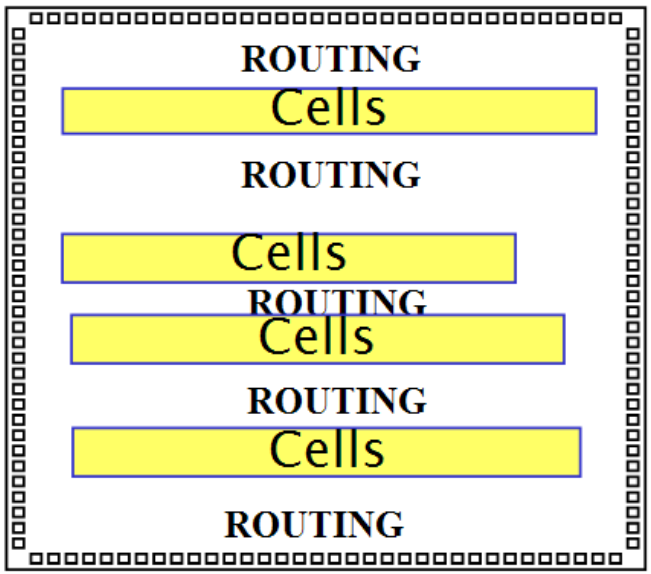
- **Constant-height cells**
- **Regular “pin” locations**
- **Cells represent gates, latches, flip-flops**
- **Placed and routed by software**

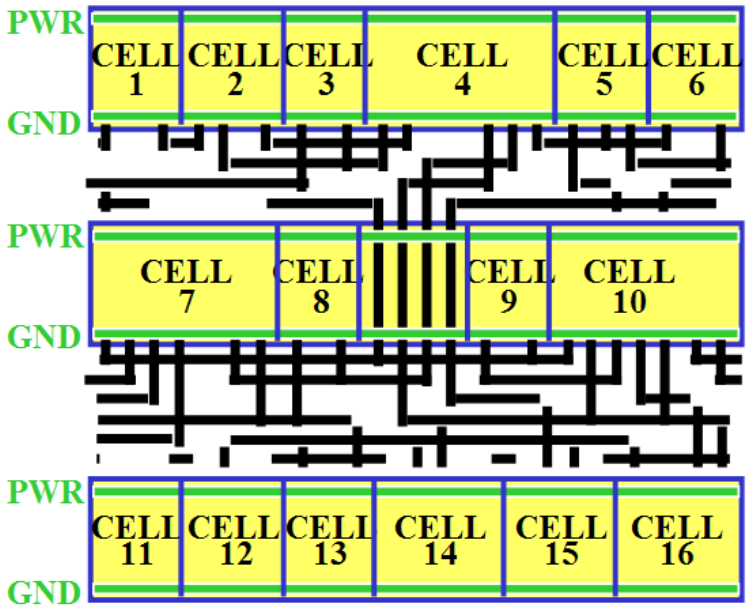


[figure from S. Hauck]

Standard Cell

- **Channels for routing only in older technologies (not necessary with modern processes with many levels of interconnect)**

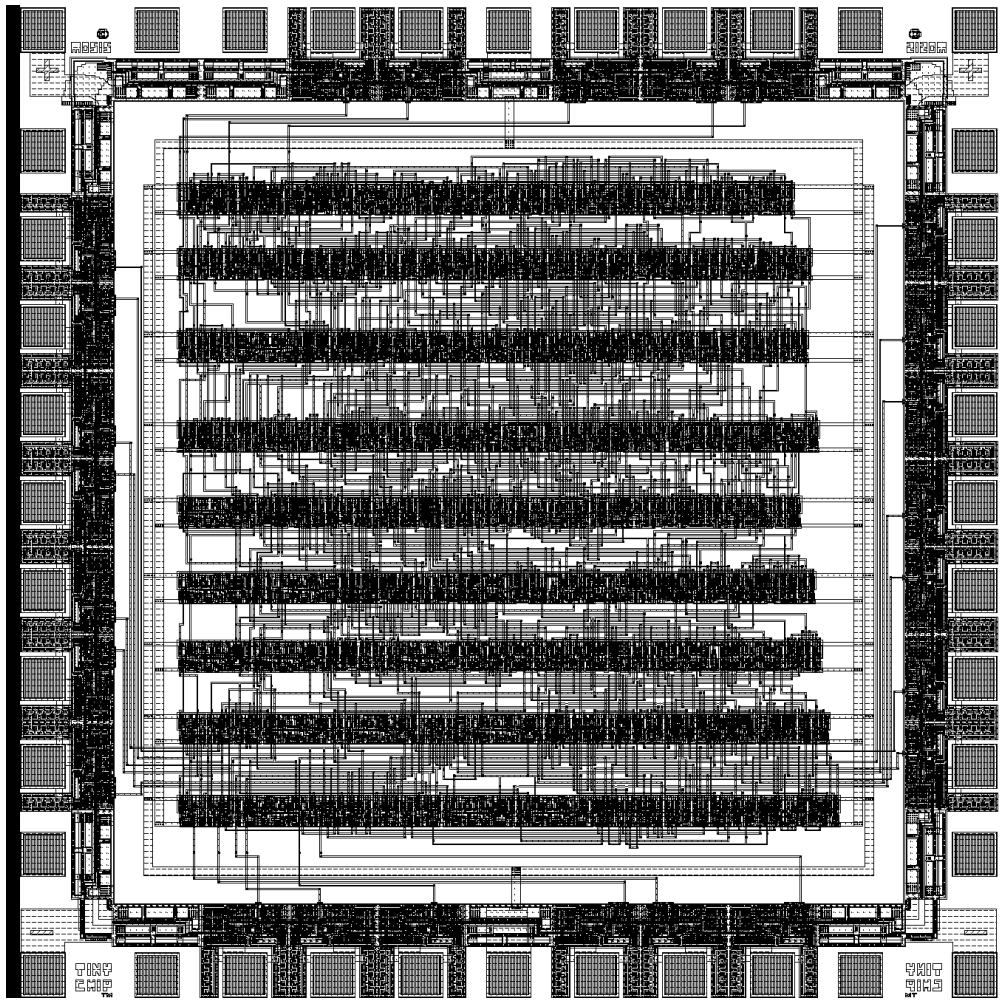




[figure from S. Hauck]

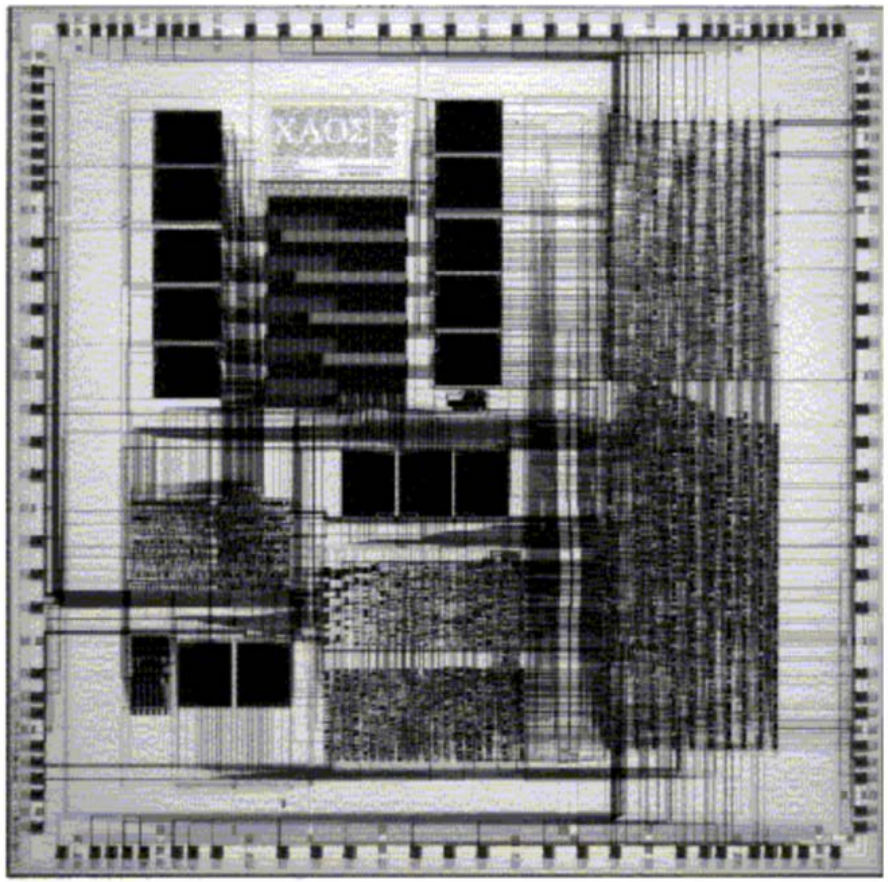
Standard Cell — Example

- **Application-Specific Integrated Circuit (ASIC)**
- **Hardwired combination of standard cells implements a fixed logic function or FSM (e.g., video codec)**



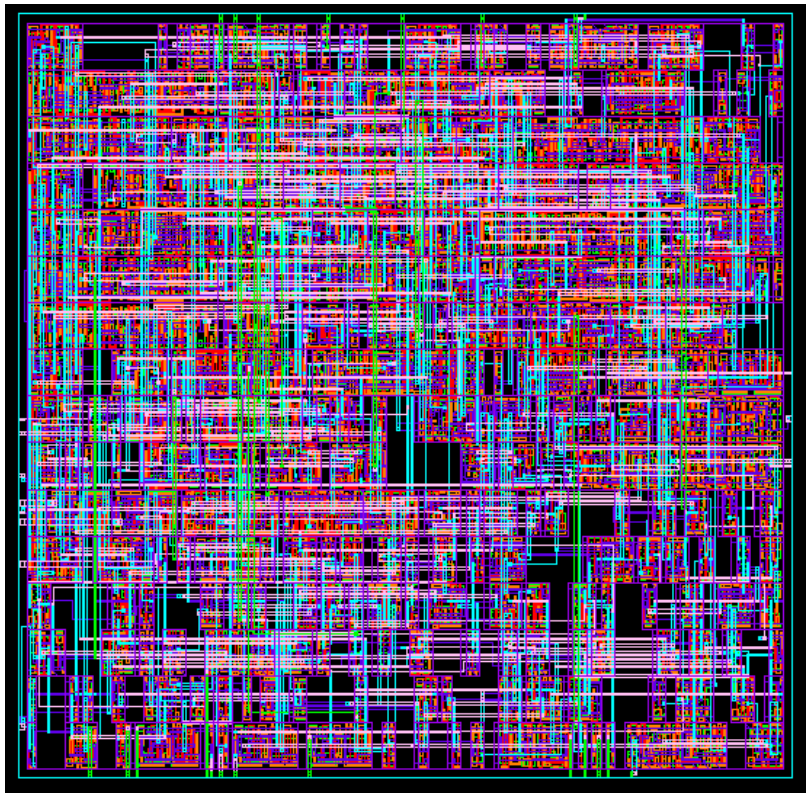
[Brodersen92]

Combination Standard Cell and Full Custom

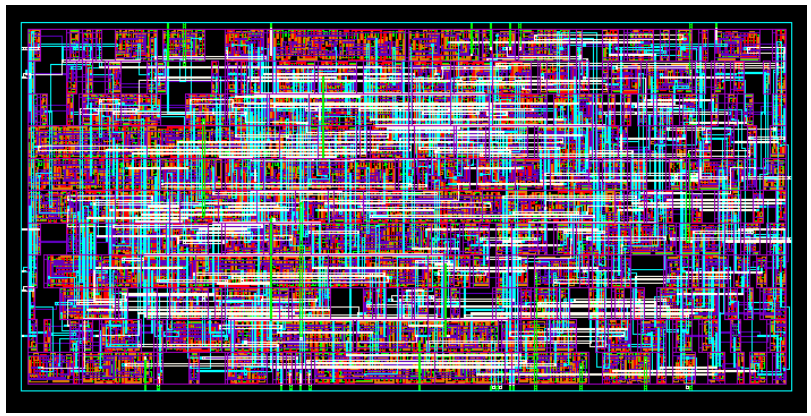


[figure from S. Hauck]

“Soft” MacroModules



```
string mat = “booth”;
directive (multtype = mat);
output signed [16] Z = A * B;
```

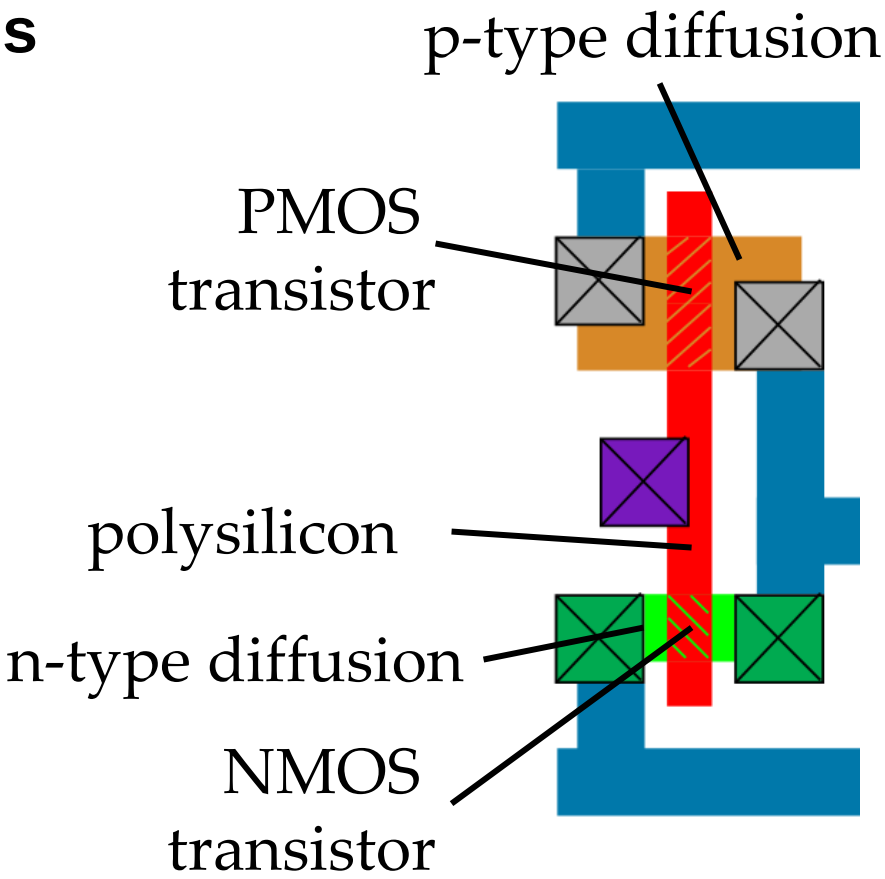


Synopsys DesignCompiler

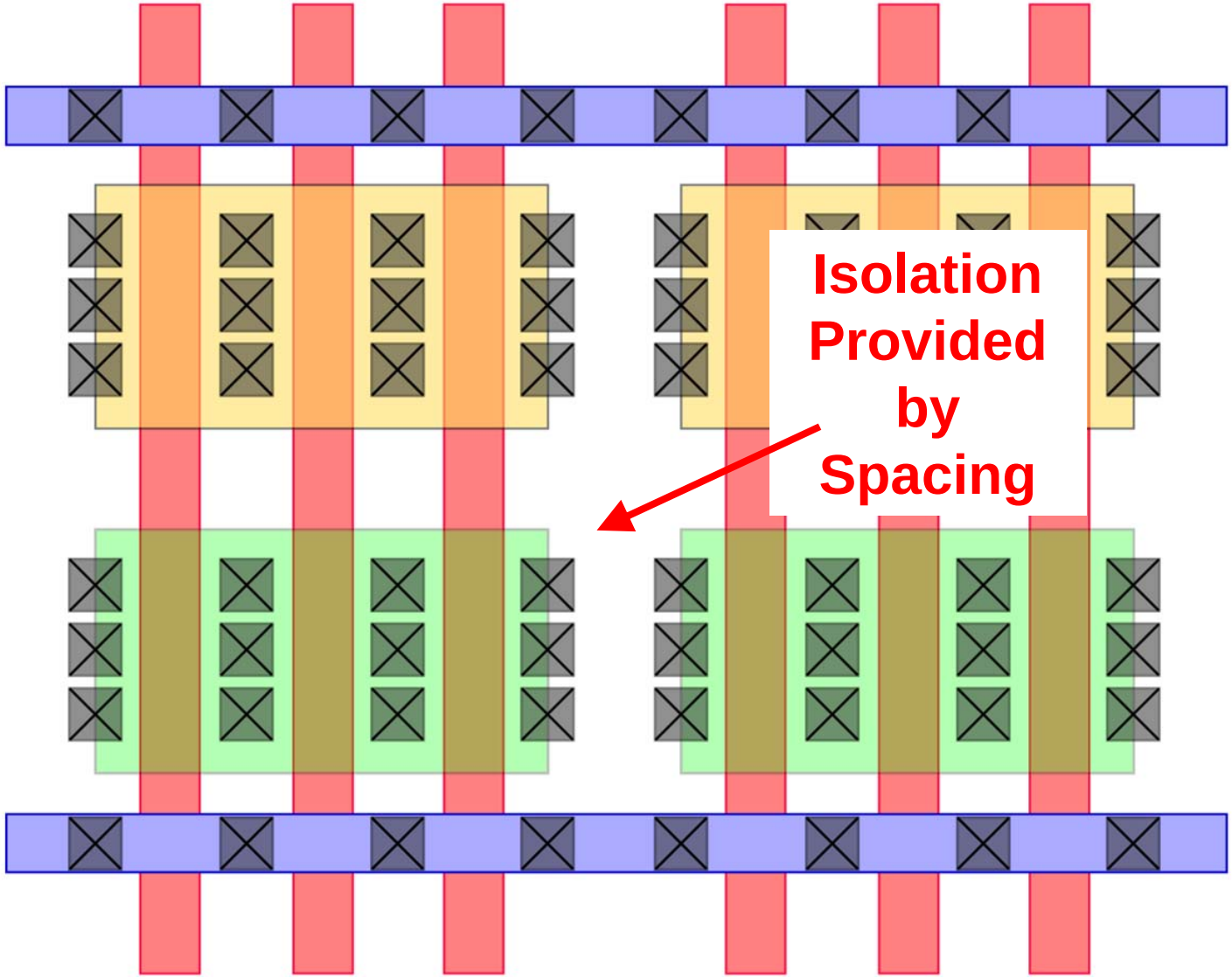
Source: Digital Integrated Circuits, 2nd ©

Gate Array

- **Polysilicon and diffusion are the same for all designs**
- **Metal layers customized for particular chips**

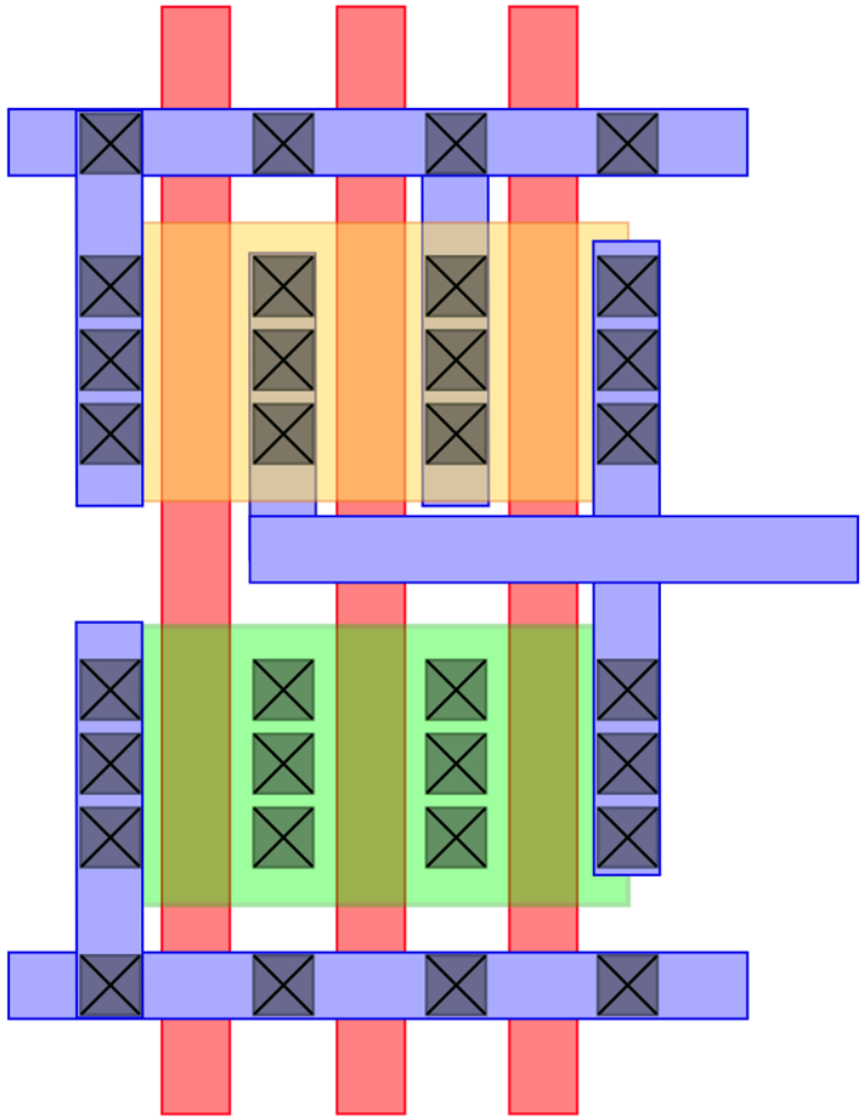


Unprogrammed Gate Array

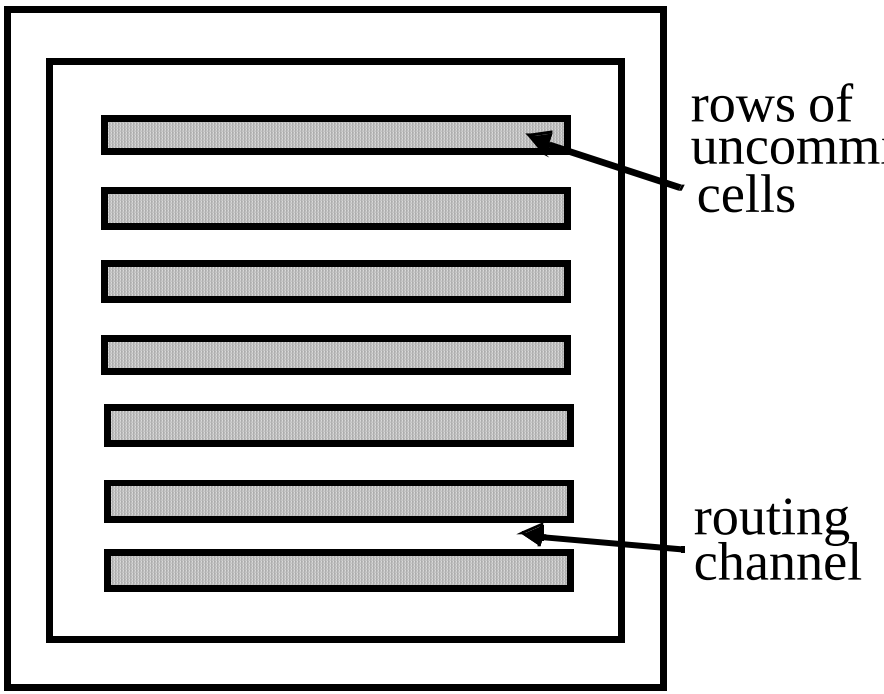


Programmed Gate Array

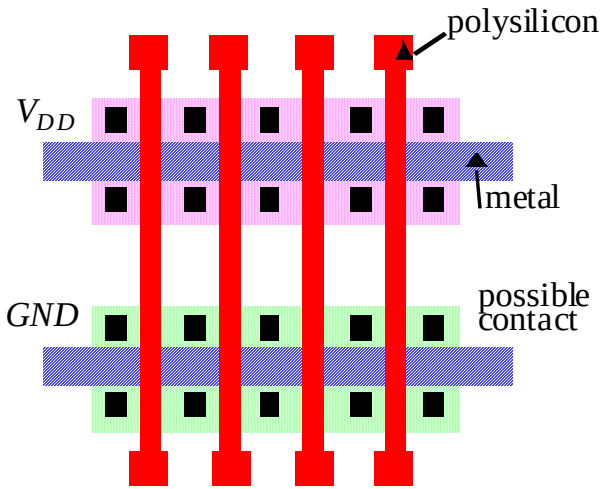
- **Metal connections made to create particular function**
- **What logic gate is this?**



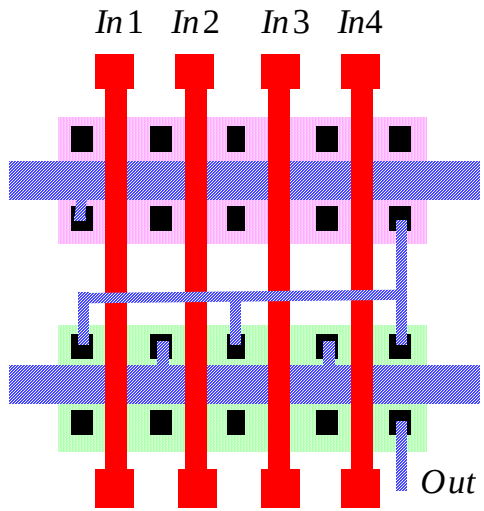
Gate Array — Sea-of-Gates



Uncommited Cell

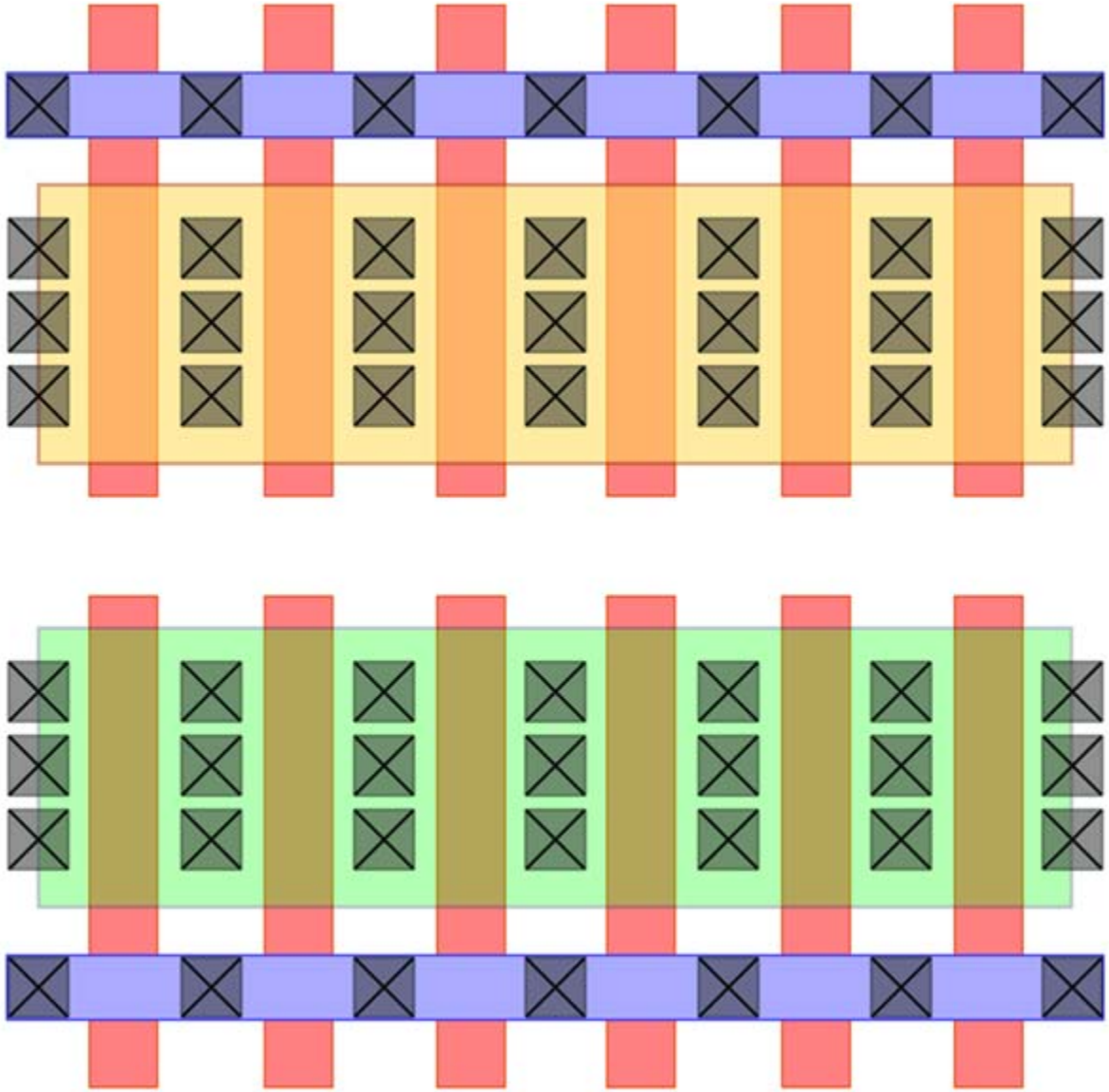


Committed Cell (4-input NOR)

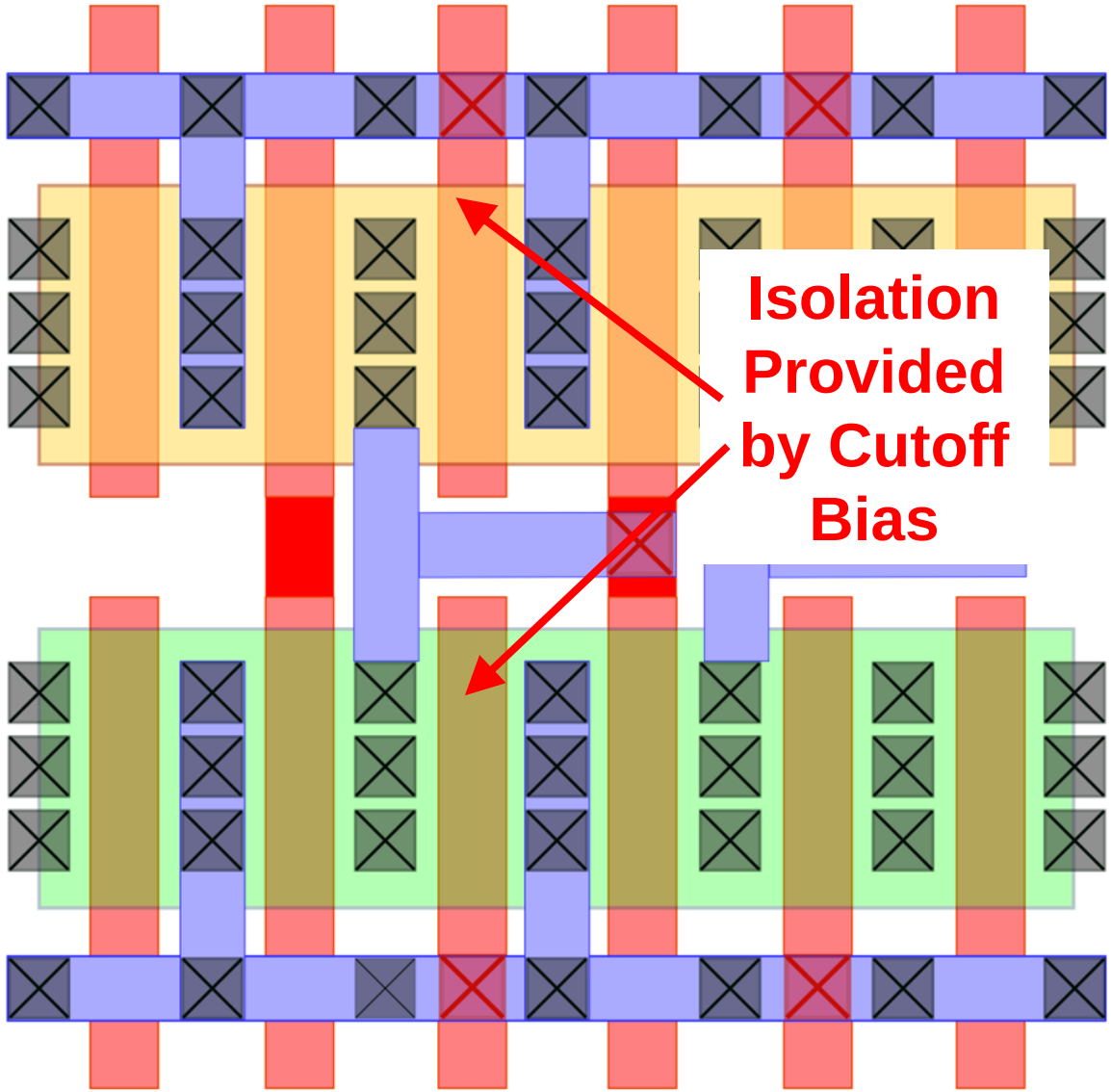


Source: Digital Integrated Circuits, 2nd ©

Unprogrammed Sea-of-Gates Array

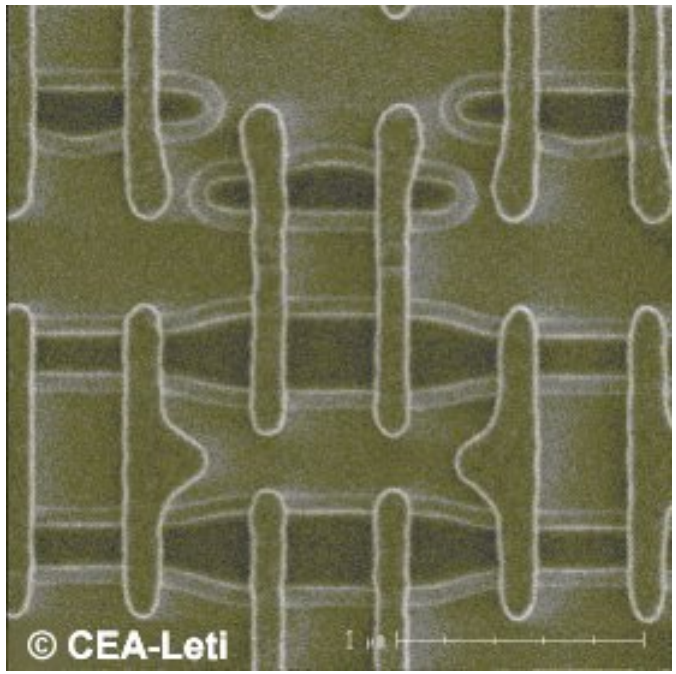


Programmed Sea-of-Gates Array



Gate Array

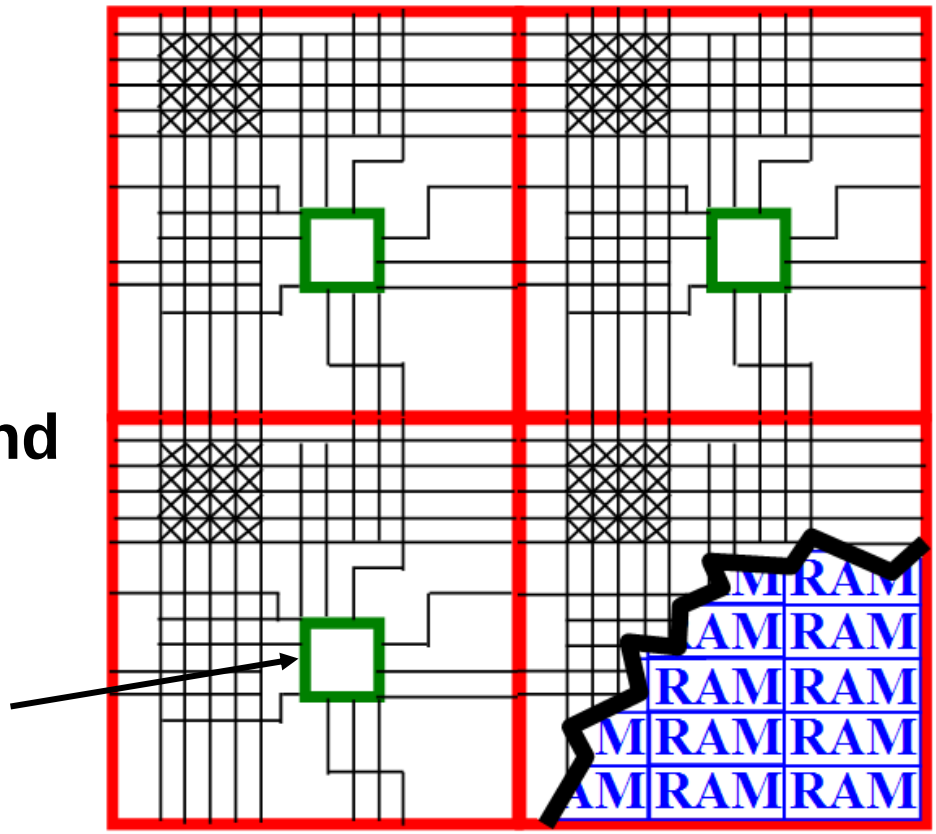
- **Polysilicon and diffusion the same for all designs**
- **0.125 um example**



[figure from LETI]

Field Programmable Gate Array (FPGA)

- **Metal layers now programmable with SRAM instead of hardwired during manufacture as with a gate array**
- **Cells contain general programmable logic and registers**

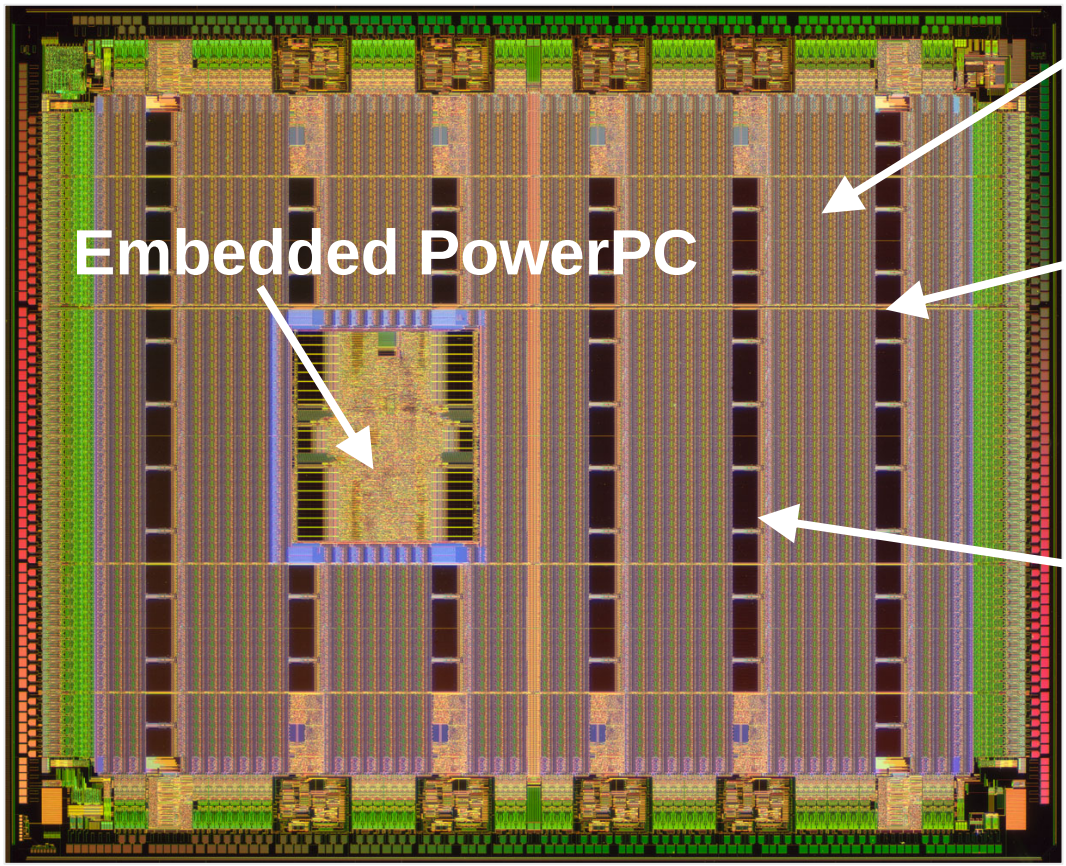


[figure from S. Hauck]

Field Programmable Gate Array (FPGA)

- **Chips can now be "designed" with software**
- **User pays for up-front chip design costs**
  - All costs: full-custom, standard cell
  - Half: gate array
  - Shared: FPGA
- **User writes code (e.g., Verilog), compiles it, and downloads into the chip**
  - Can be used to prototype standard cell (ASIC) design

Heterogeneous Programmable Platforms



FPGA Fabric

Embedded memories

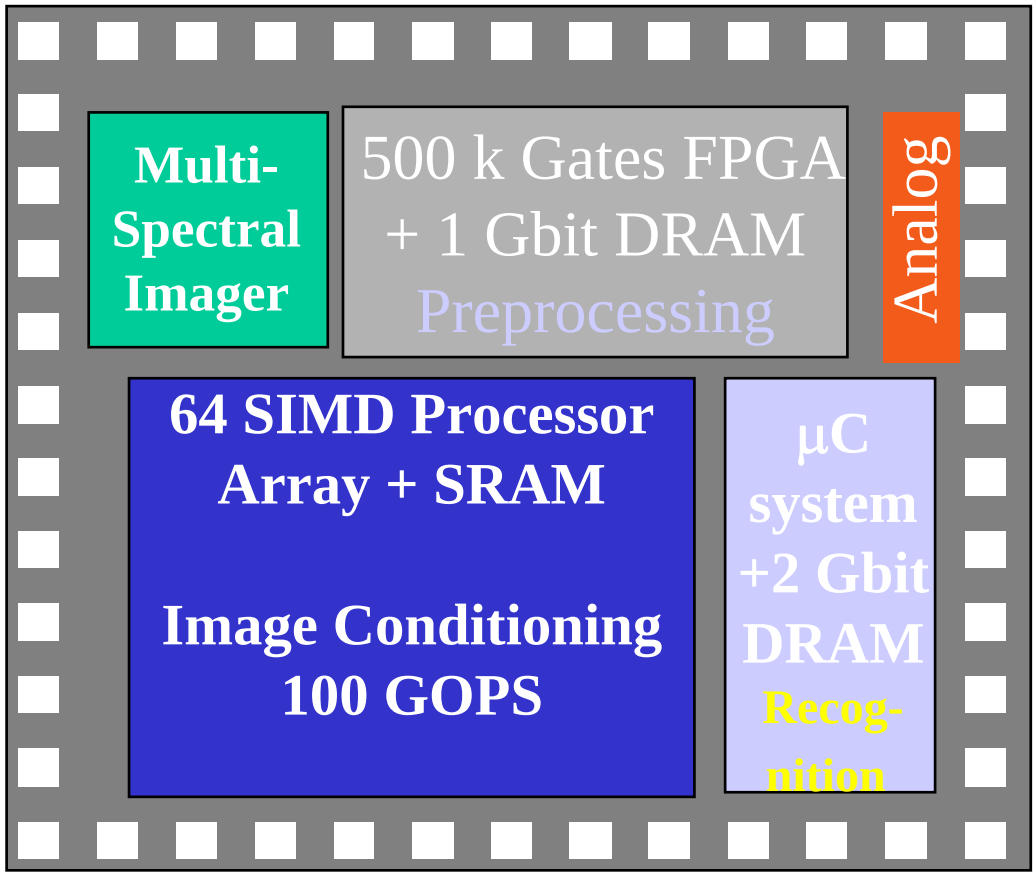
Hardwired multipliers

Xilinx Vertex-II Pro

High-speed I/O

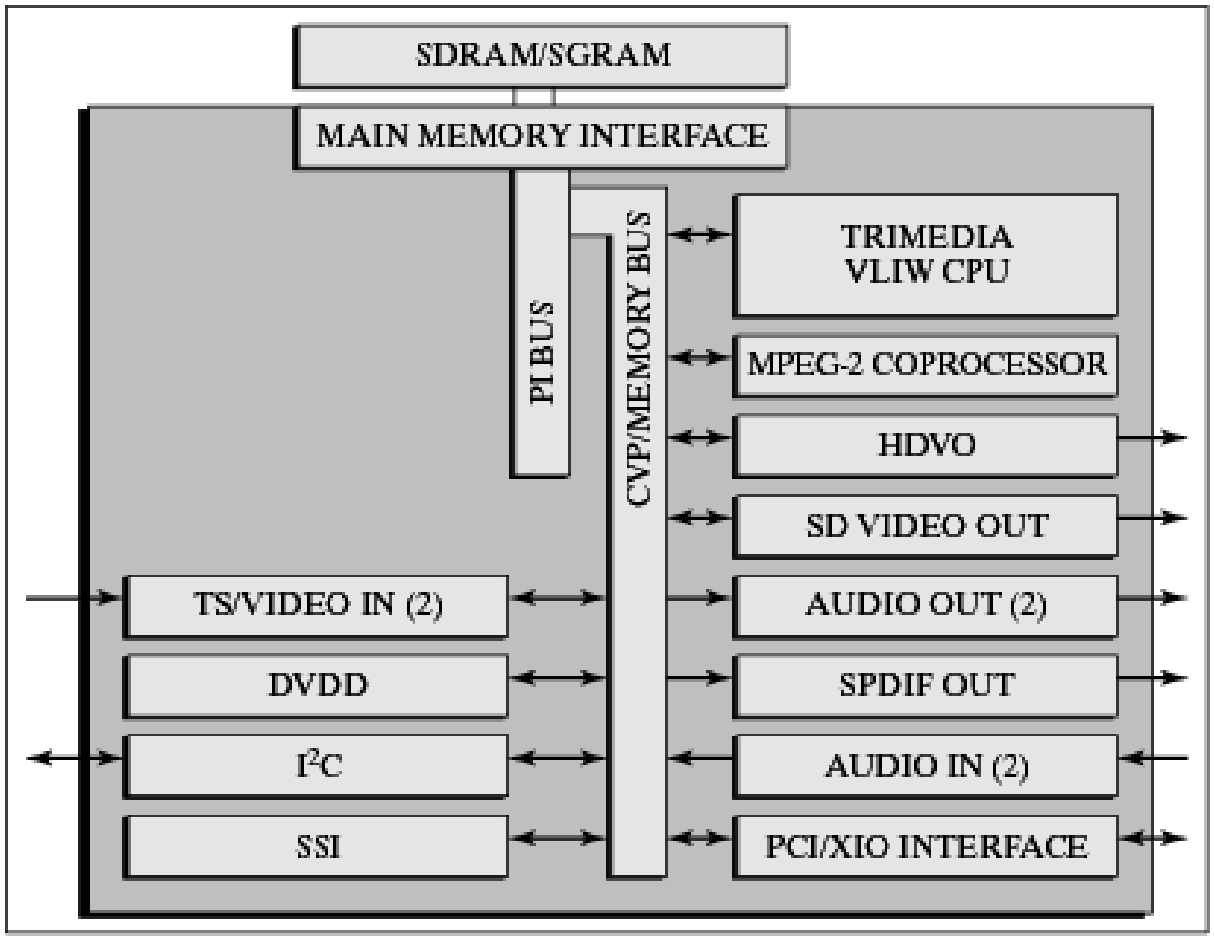
Courtesy Xilinx

Design at a Crossroad: System-on-a-Chip



- **Often used in embedded applications where cost, performance, and energy are big issues!**
- **DSP and control**
- **Mixed-mode**
- **Combines programmable and application-specific modules**
- **Software plays crucial role**

A SoC Example: High Definition TV Chip



Courtesy: Philips

Standard Cell Based IC vs. Custom Design IC

- **Standard cell based IC:**
  - Design using standard cells
  - Standard cells come from library provider
  - Many EDA tools to automate this flow
  - Shorter design time
- **Custom designed IC:**
  - Designed by individual engineers using manual process
  - Higher performance

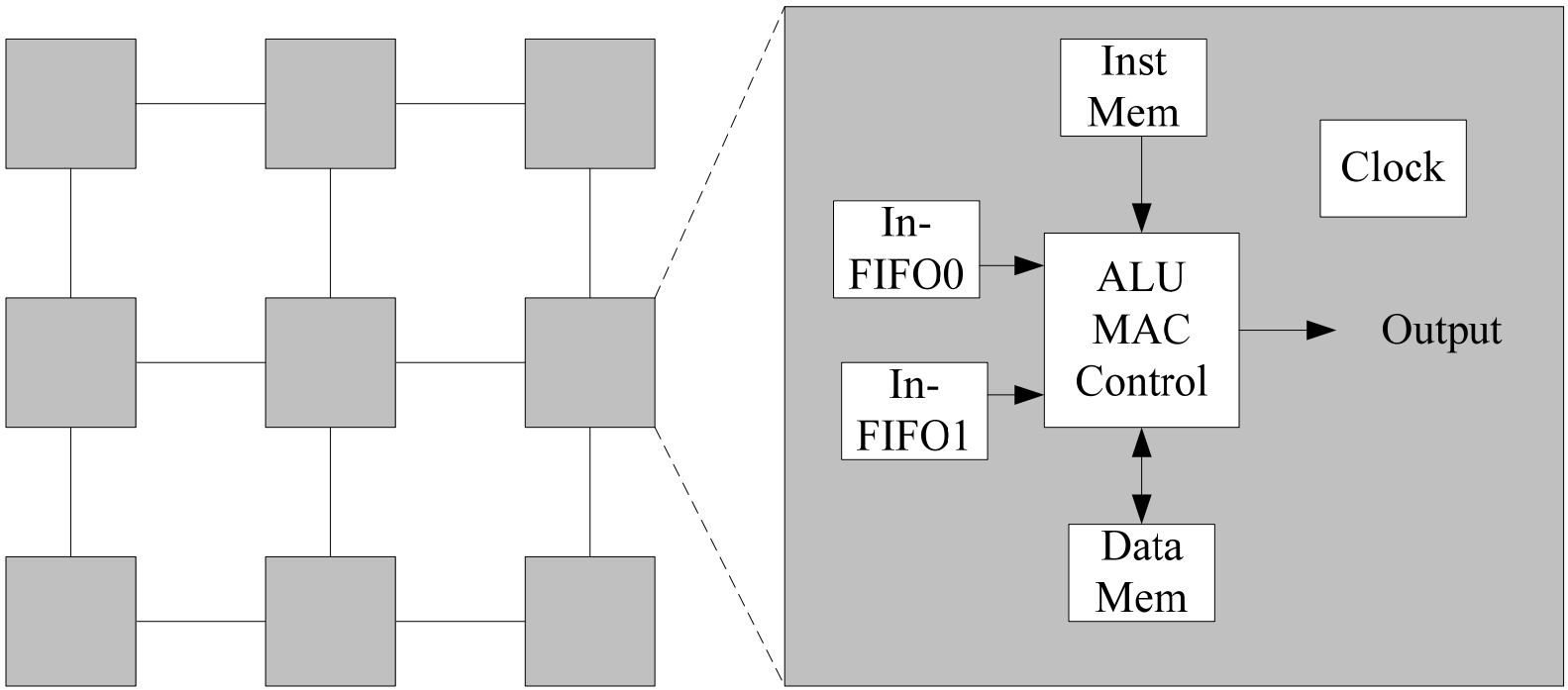
Standard Cell Based VLSI Design Flow

- **Front end**
  - System specification and architecture
  - HDL coding & behavioral simulation
  - Synthesis & gate level simulation
- **Back end**
  - Placement and routing
  - DRC, LVS
  - Dynamic simulation and static analysis

Asynchronous Array of Simple Processors

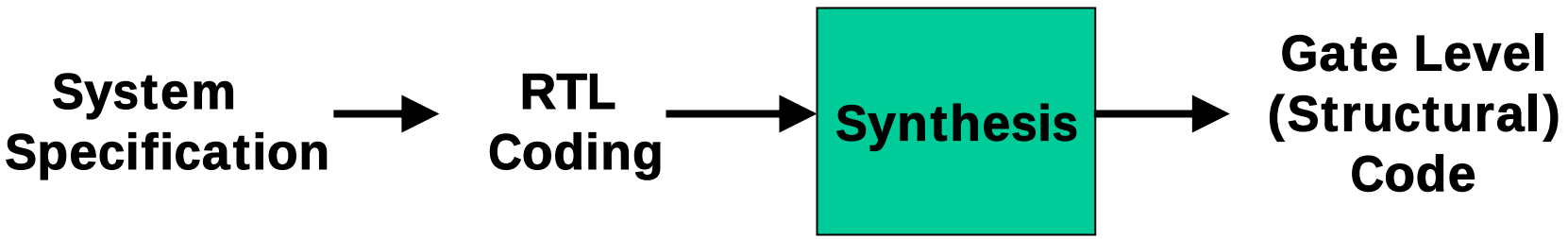
- **AsAP Project by Prof. Baas' VLSI Computation Lab**
- **A processing chip containing multiple uniform simple processor elements**
- **Each processor has its local clock generator**
- **Each processor can communicate with its neighbor processors using dual-clock first-in first-out buffers (FIFOs)**

Diagram of a 3x3 AsAP



More information: http://www.ece.ucdavis.edu/vcl/asap/

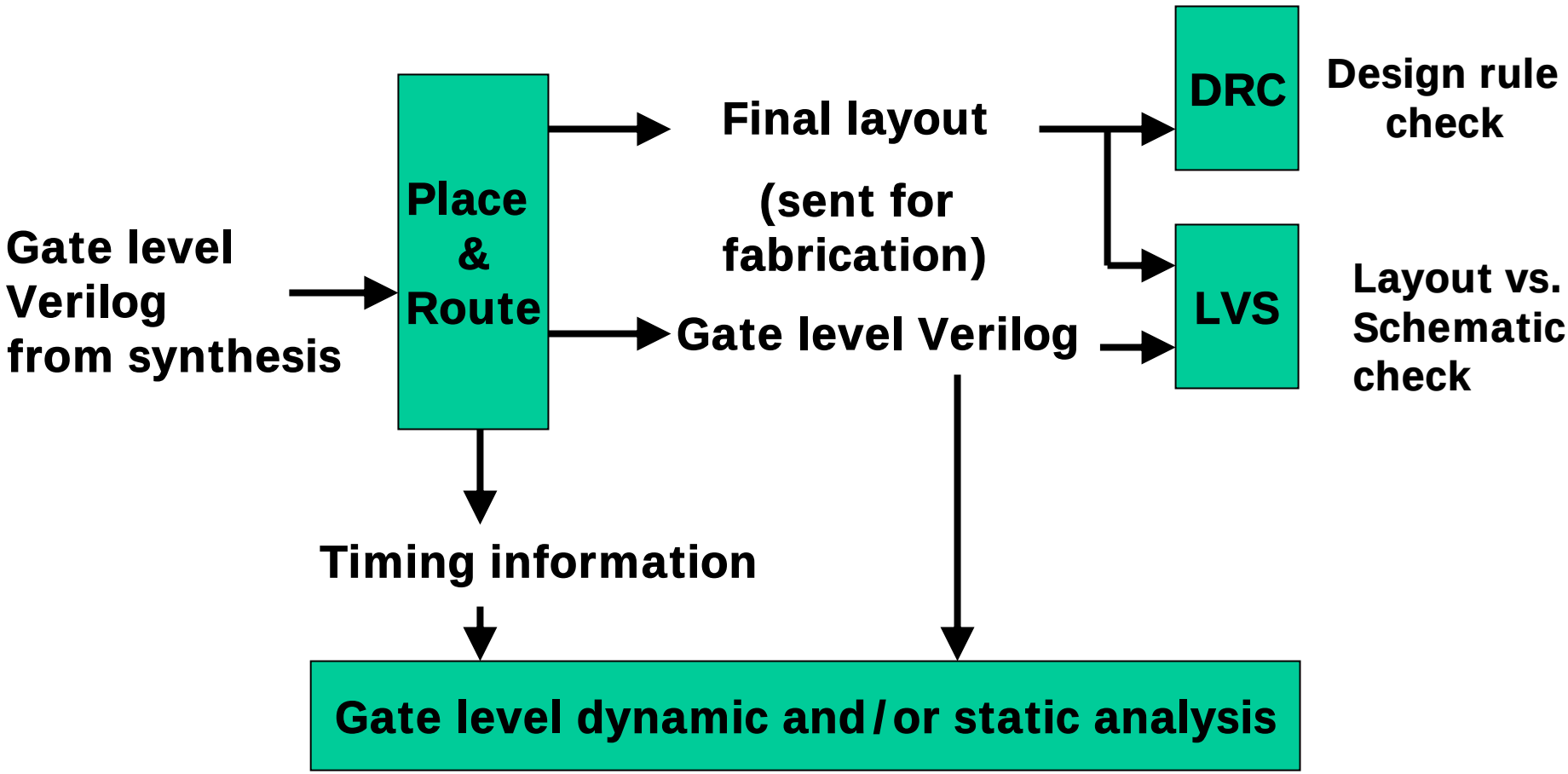
Simple Diagram of Front-End Design Flow



Ex: c = !a & b →

```
INV (.in (a),     .out (a_inv));
AND (.in1 (a_inv), .in2 (b),. out (c));
```

Simple Diagram of Back-End Design Flow



Back-end Design of AsAP

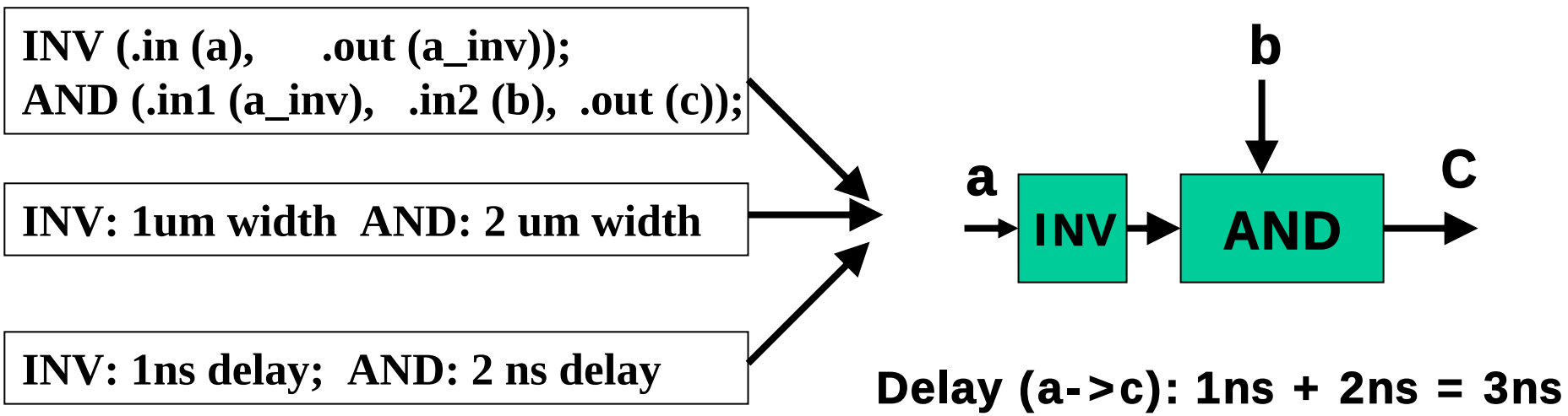
- **Technology: CMOS 0.18 um**
- **Standard cell library: Artisan**
- **Tools**
  - Placement & Route: Cadence Encounter
  - Final layout edit: icfb
  - DRC & LVS: Calibre
  - Static timing analysis: Primetime

Flow of Placement and Routing

- **Import needed files**
- **Floorplan**
- **Placement & in-place optimization**
- **Clock tree generation**
- **Routing**

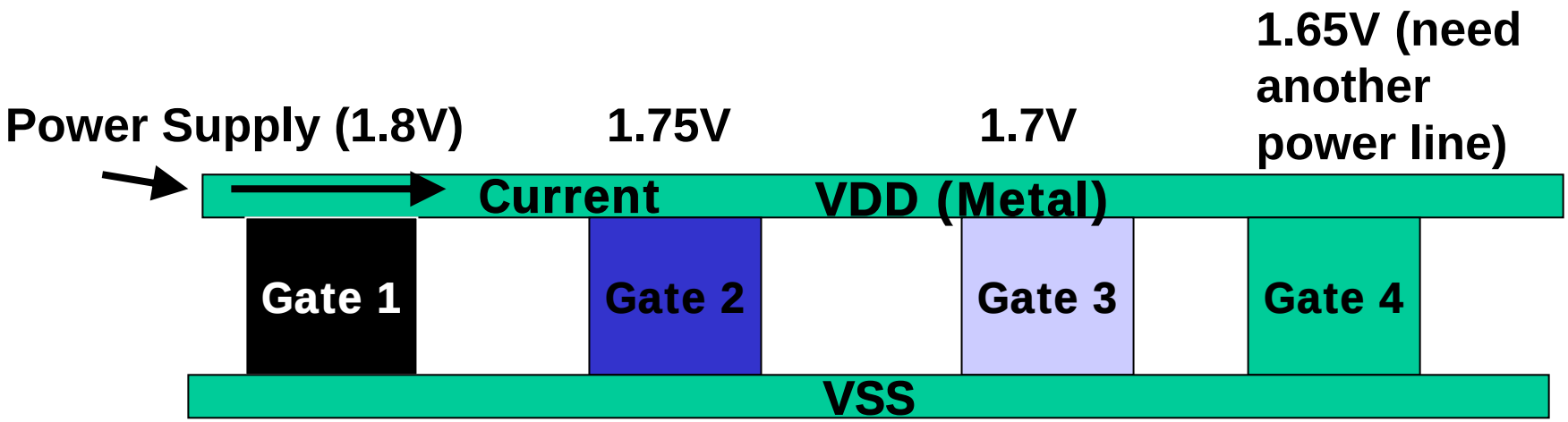
Import Needed Files

- Gate level verilog (.v)
- Geometry information (.lef)
- Timing information (.lib)



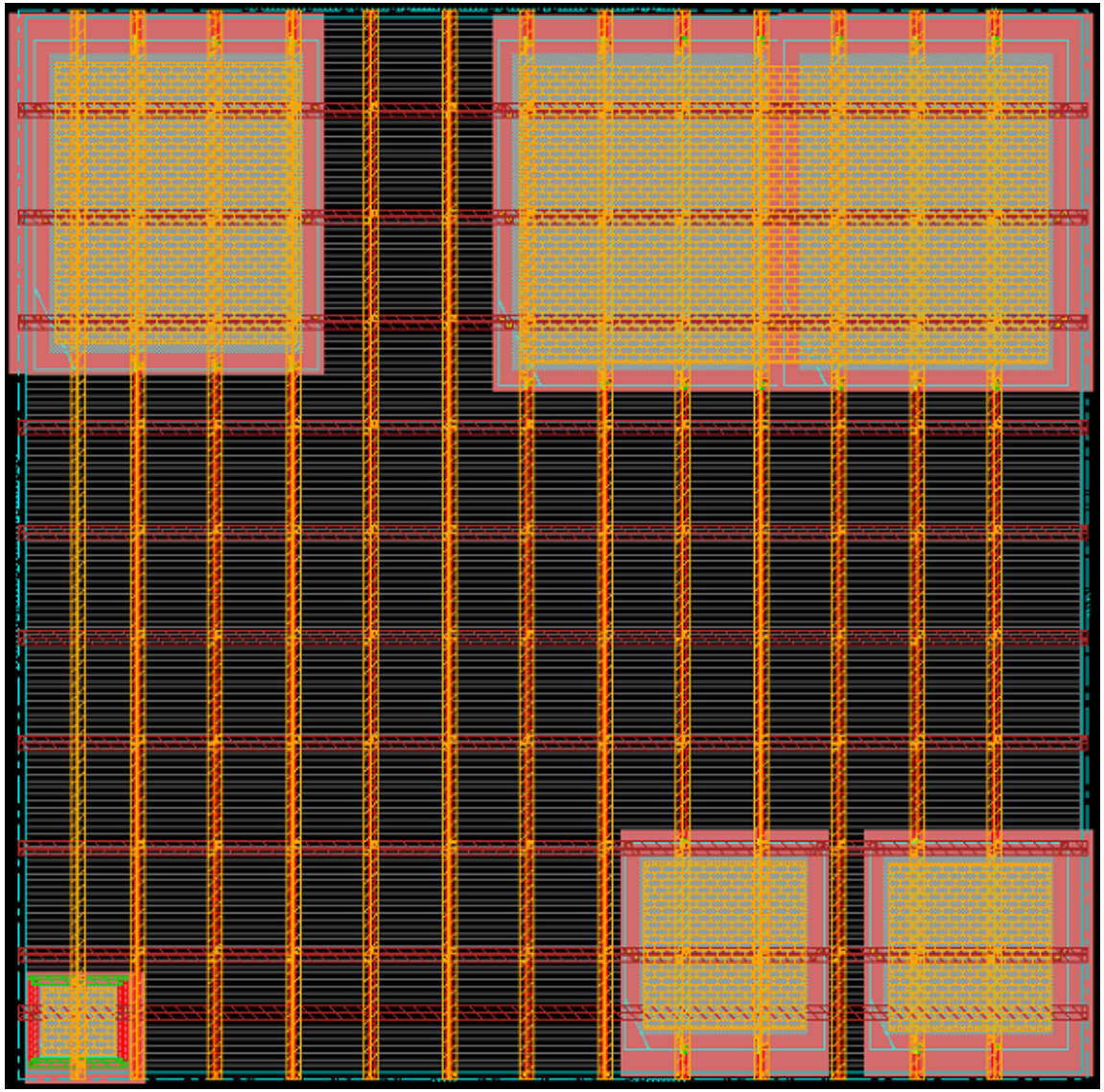
Floorplan

- Size of chip
- Location of pins
- Location of main blocks
- Power supply: deliver enough power for each gate



Voltage drop equation: V2 = V1 – I * R

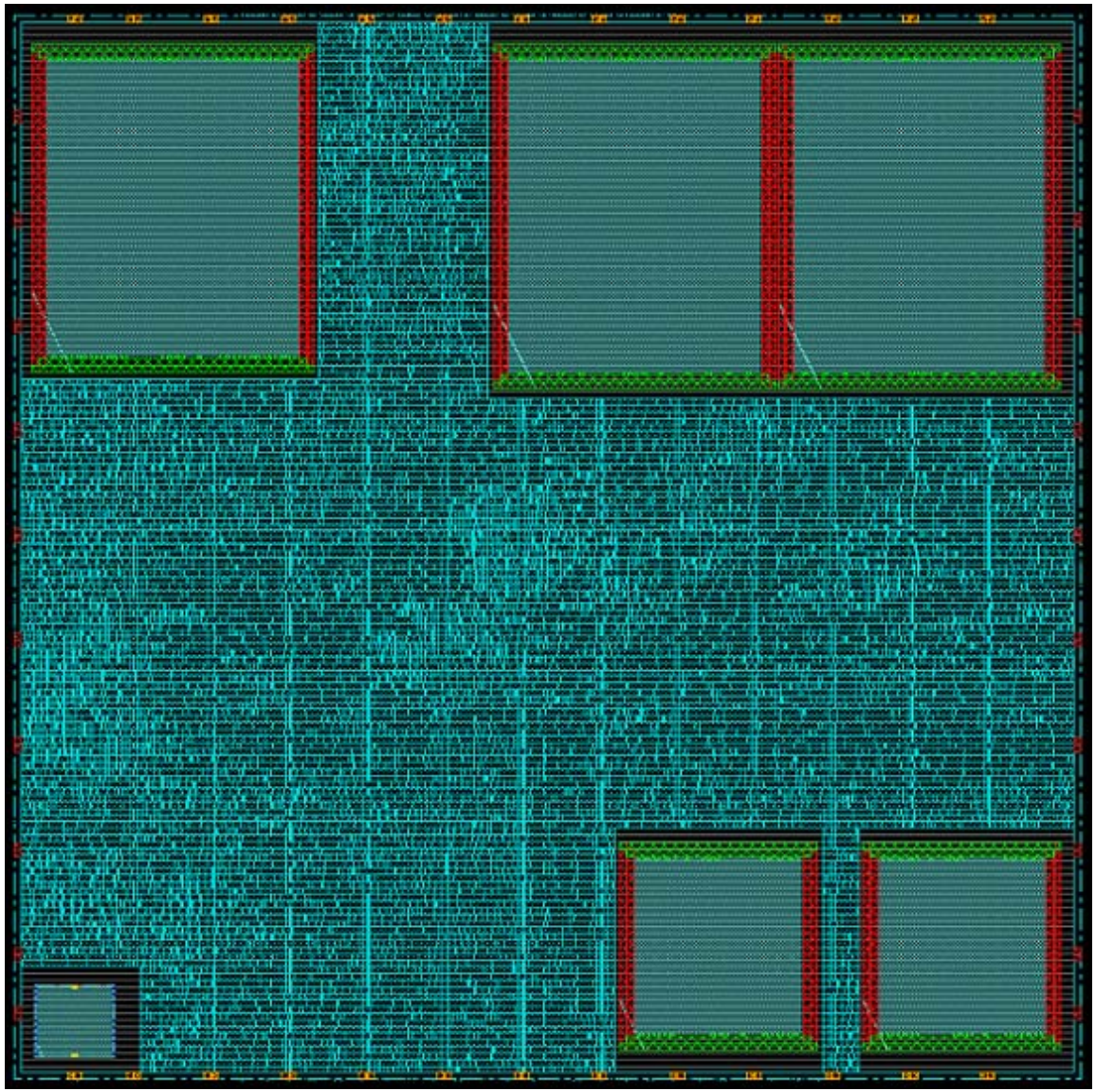
Single Processor Floorplan



Placement and In-Placement Optimization

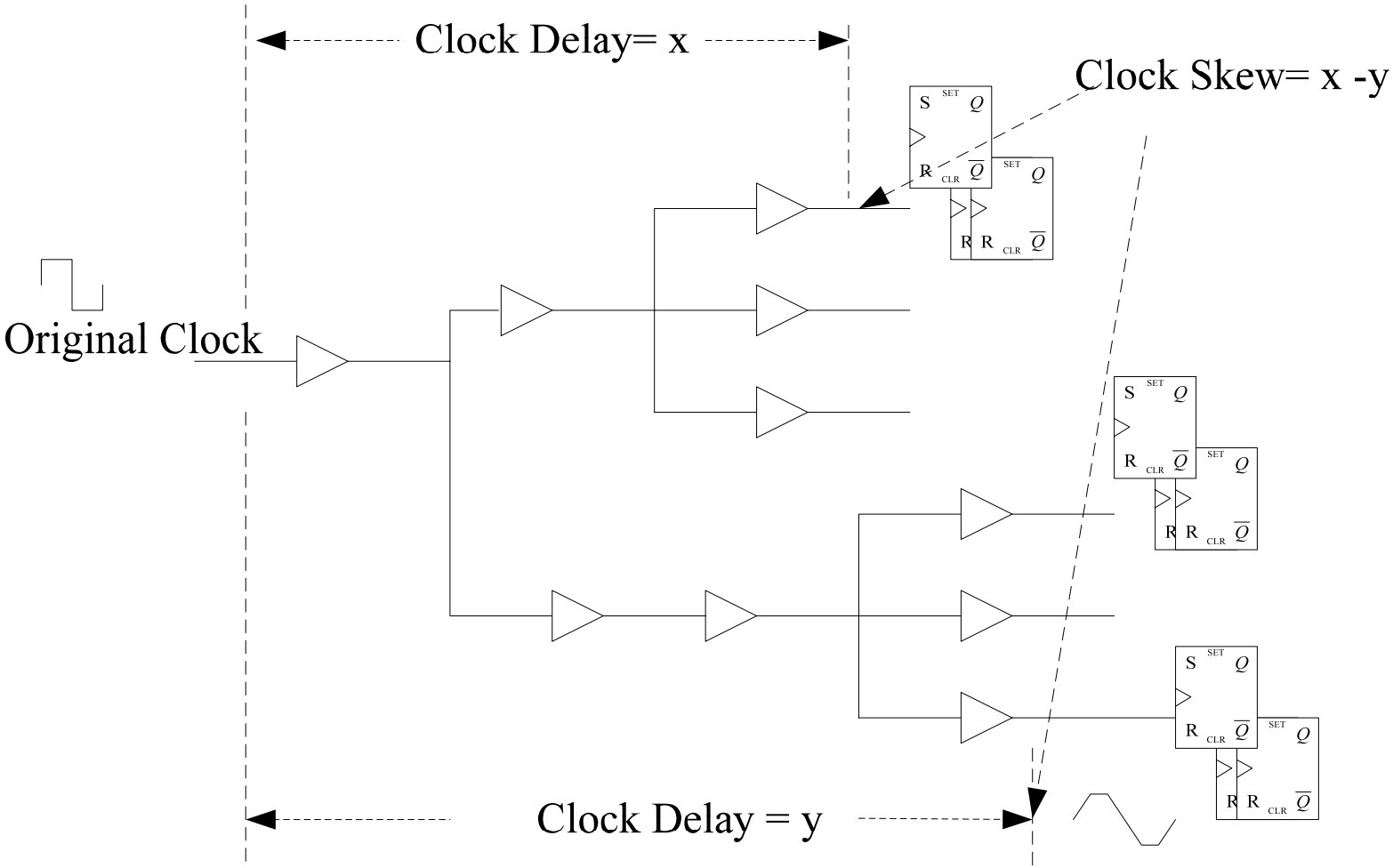
- **Placement: place the gates on floorplan**
- **In-placement optimization**
  - Why: timing information difference between synthesis and layout (wire delay)
  - How: change gate size, insert buffers
  - Should not change the circuit function!!

Single Processor: Gate Placement

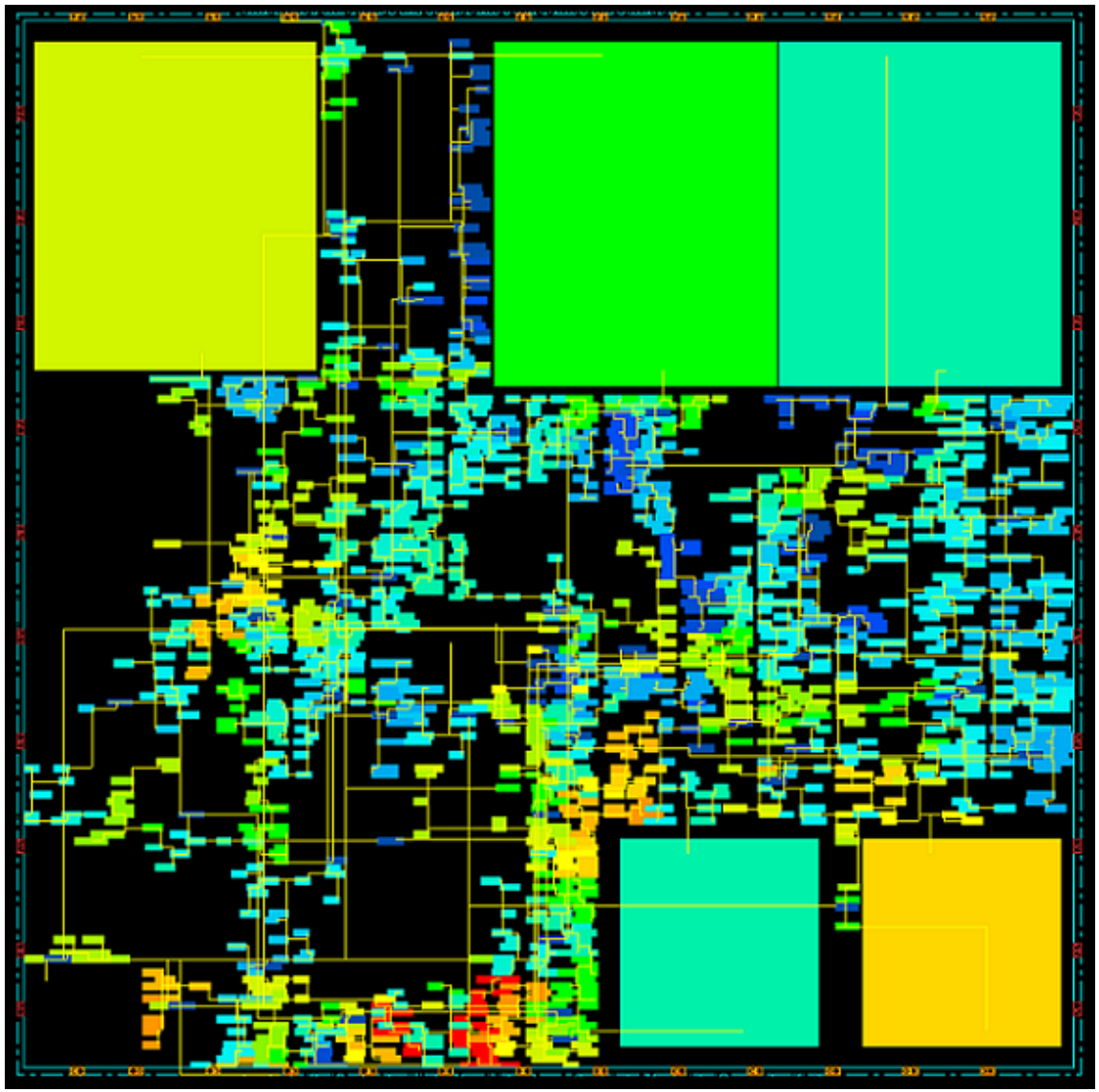


Clock Tree

- **Main parameters: skew, delay, transition time**



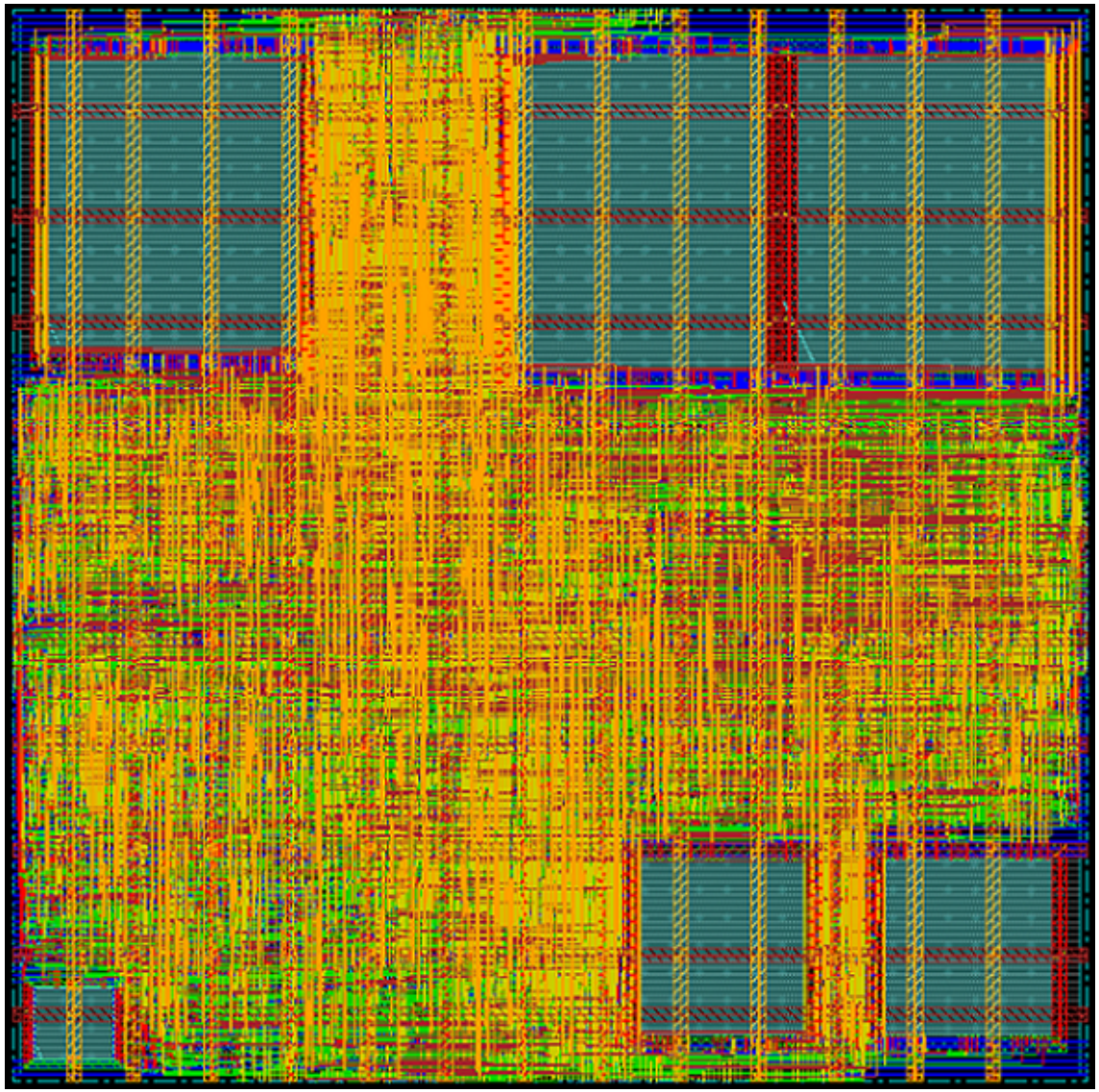
Single Processor Clock Tree



Routing

- **Connect the gates using wires**
- **Two steps**
  - Connect the global signals (power)
  - Connect other signals

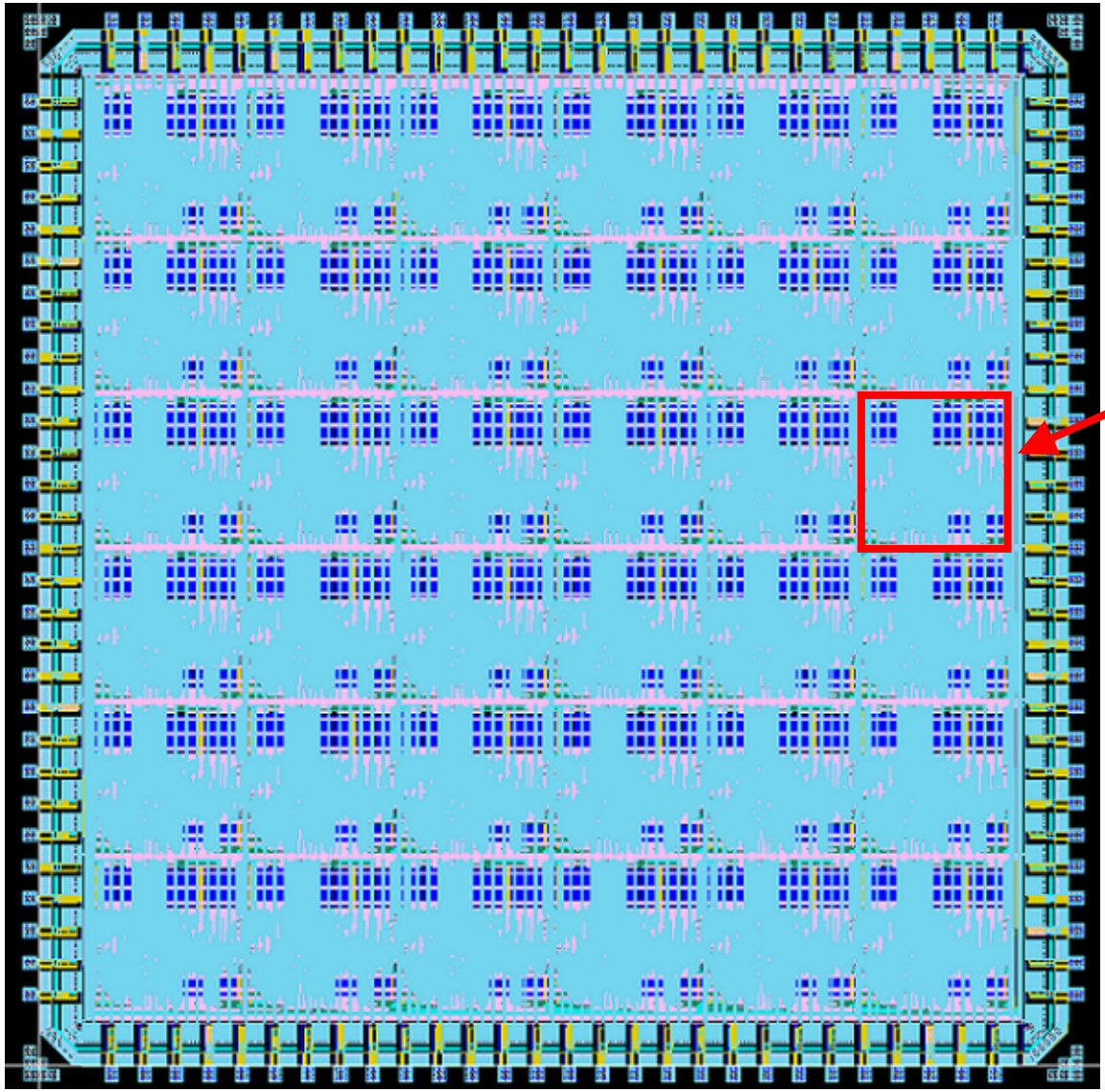
Single Processor Layout



Area:
0.8mm x 0.8mm

Estimated clock frequency:
450 MHz

First Generation 6x6 AsAP Layout



Area: 30mm²
36 processors
114 I/O pads

Single Processor

Verification After Layout

- **DRC (Design Rule Check)**
- **LVS (Layout vs. Schematic)**
  - GDS vs. (Verilog + Spectre/Spice module)
- **Gate level Verilog dynamic simulation**
  - Mainly check the function
  - Different with synthesis result: clock, OPT
- **Gate level static analysis**
  - Check all the paths

Useful Design and Simulation Tools

- **Dynamic HDL Simulation**
  - Modelsim (Mentor), NC-verilog (Cadence), Active-HDL
- **Dynamic Analog Simulation**
  - Spectre (Cadence), Hspice (Synopsys)
- **Synthesis**
  - RTL Compiler (Cadence)
  - Design-Compiler, Design-Analyzer (Synopsys)
- **Placement & Routing**
  - Encounter & Virtuoso (Cadence)
  - Astro (Synopsys)

Useful Verification Tools

- **DRC & LVS**
  - Calibre (Mentor)
  - Diva (Cadence – used in EEC 116/119AB)
  - Dracula (Cadence)
- **Static Analysis**
  - Primetime (Synsopsys)

Next Topics: Low Power and DFM

- **Low power design principles and circuit techniques**
  - Voltage scaling, activity factor reduction, clock gating, leakage reduction
- **Design for Manufacturability**
  - Parameter variations in CMOS digital circuits
  - Yield maximization and worst-case design