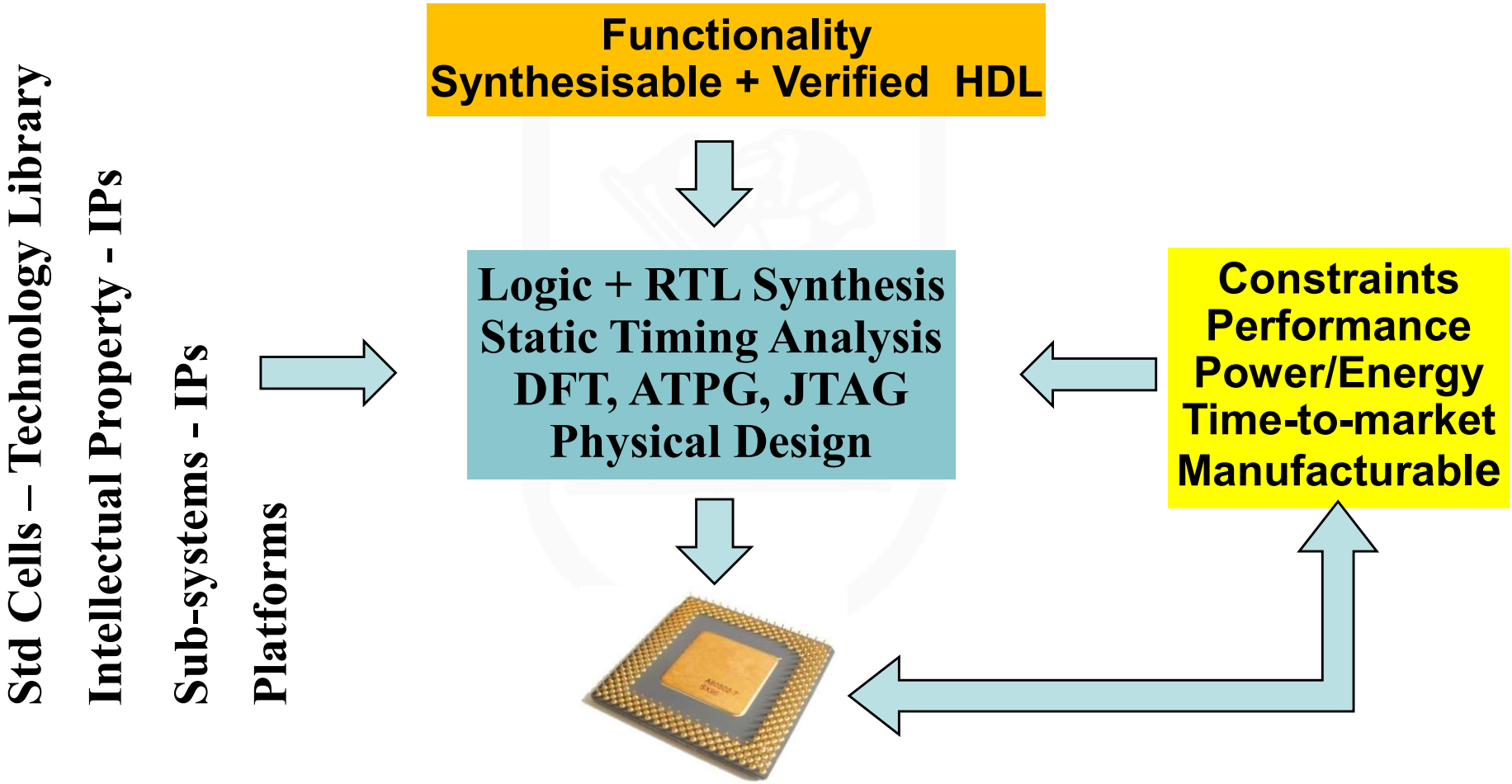
What the ASIC flow is about ?



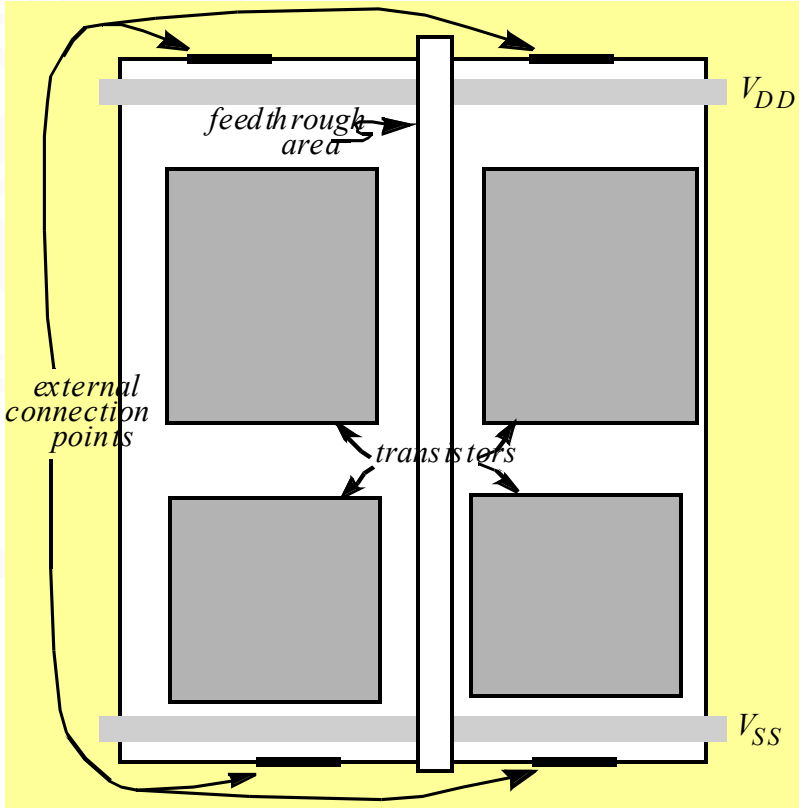
The Industrial Perspective

- **ASICs are everywhere**
  - Telecommunication
  - Consumer Electronics
  - White Goods
  - Automobile Industry
  - Process Industry
  - Computers
- **ASIC Designers**
  - Corresponds to Software Engineers/Designers/Architects in the Hardware domain
- **ASIC Roles/Jobs/Tasks**
  - System Designer/Architect
  - Logic Designer
  - Verification Engineer
  - DFT(Design For Test) Engineer
  - Front End ASIC Designer
  - Back End ASIC Designer
- **Related Roles**
  - Integration and Test
  - Embedded Engineers/Designers
- **ASIC Advantages**
  - Performance
  - Cost
  - Power Consumption
  - Differentiation
- **ASIC Disadvantages**
  - High Non Recurring Engineering (NRE) cost
  - Large Volumes
  - Rigidity
- **ASIC Alternatives**
  - FPGAs
  - Software
  - Reconfigurable Architectures

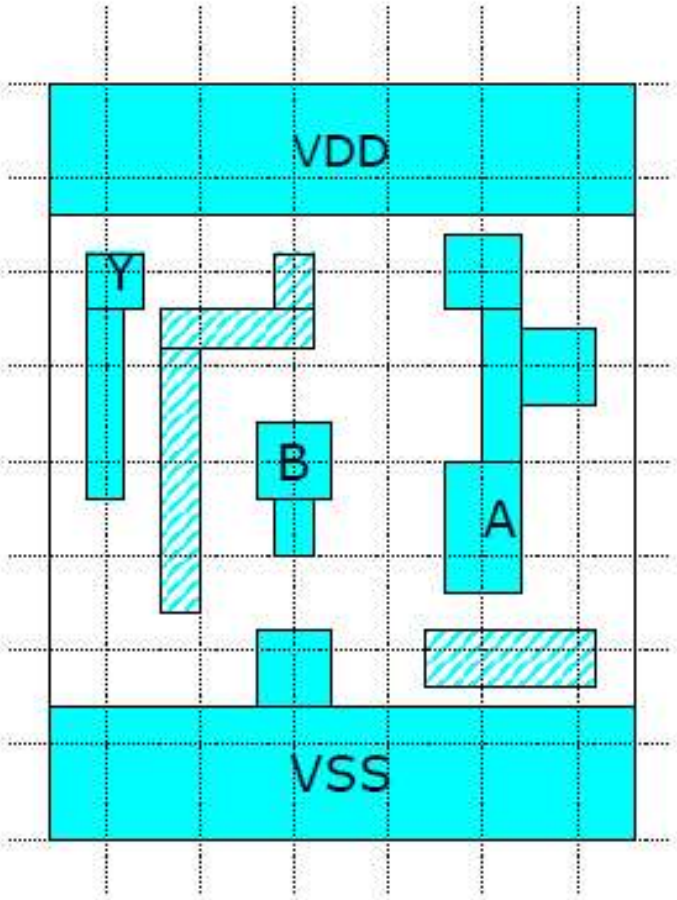
Standard Cells

A cell that performs a dedicated function. For instance 2 input NAND, XOR, D flip-flop etc. Designer works with cells and is not bothered about transistor level details.

```
library (xyz) {
    cell (nand2) {
        area: 5;
        pin (a1, a2) {
            direction: input;
            capacitance: 1;
            }
        pin (o1) {
            direction: output;
            function: !(a1 * a2);
            timing () {
                intrinsic_rise: 0.37;
                intrinsic_fall: 0.56;
                rise_resistance: 0.1234;
                fall_resistance: 0.4567;
                related_pin: “a1 a2”;
                }
            }
        }
}
```

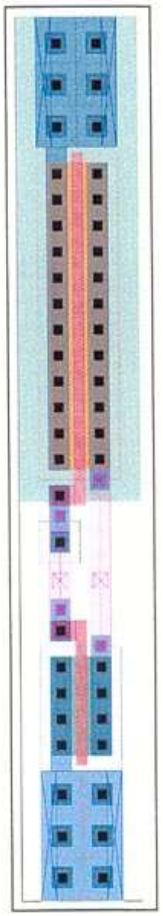


Standard Cells (cont.)

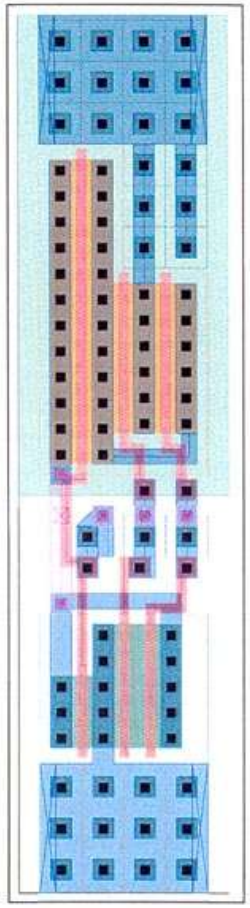



```
MACRO ADD1
   CLASS CORE ;
   FOREIGN ADD1 0.0 0.0 ;
   ORIGEN 0.0 0.0 ;
   LEQ ADD ;
   SIZE 19.8 BY 6.4 ;
   SYMMETRY x y ;
   SITE coresite
   PIN A
      DIRECTION INPUT ;
      PORT
      LAYER Metal1 ;
      RECT 19.2 8.2 19.5 10.3
      ......
      END
   END A
   OBS
      ......
   END
END ADD1
```

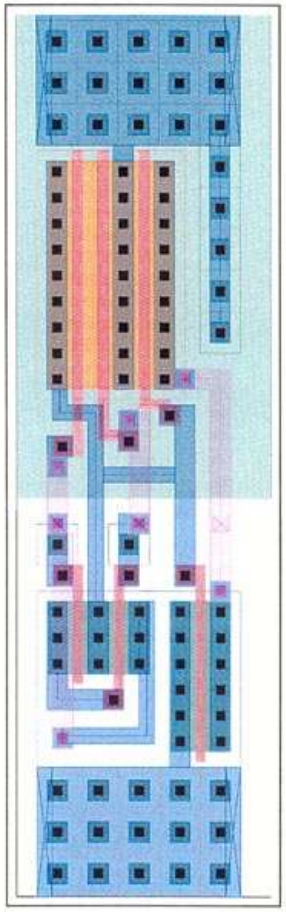
Standard Cells (cont.)



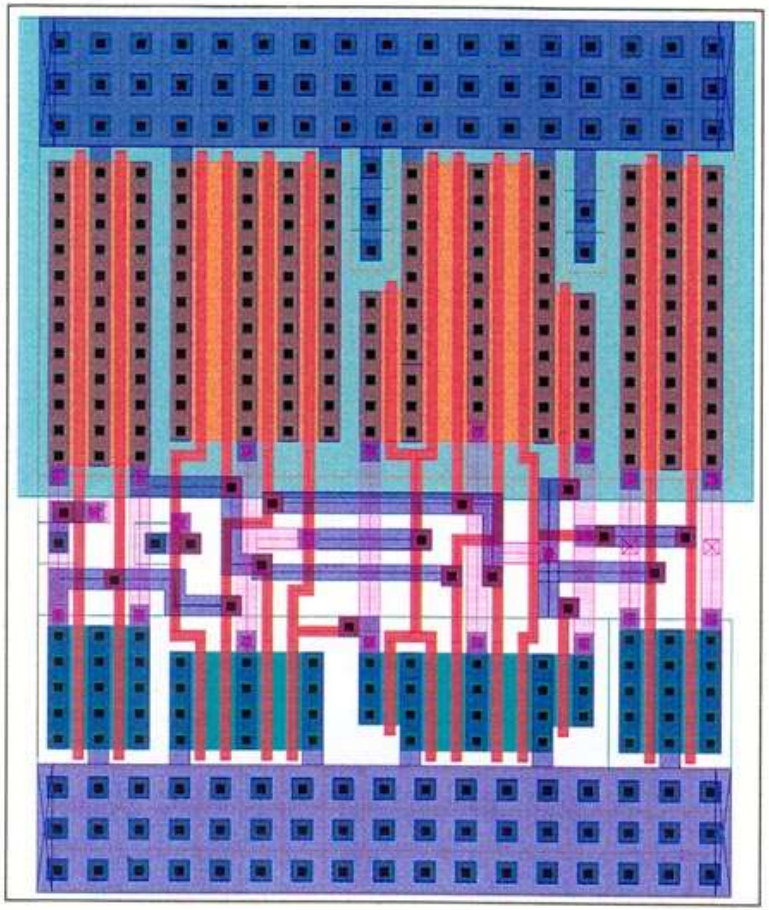
INV



AND



XOR



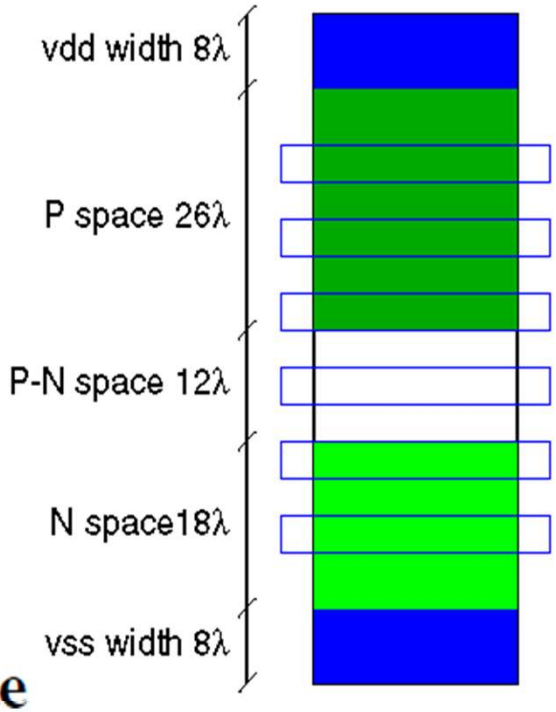
DFF

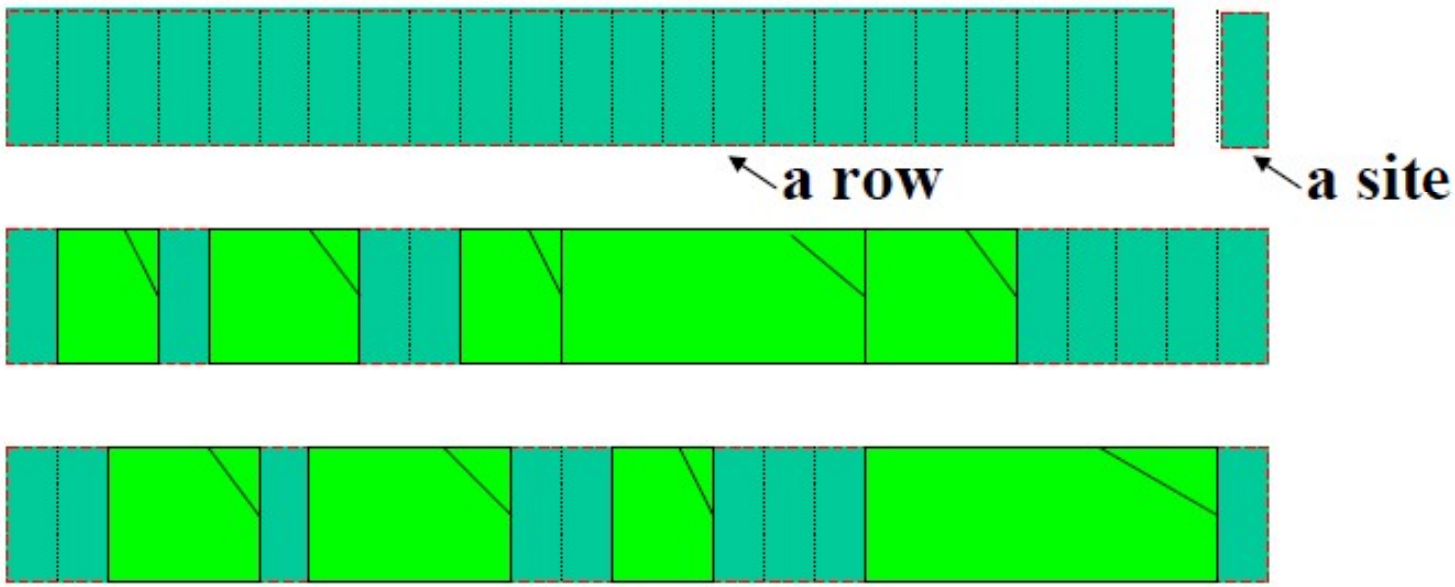
Standard Cell Layout (cont.)

- Cells in the same row or adjacent rows can be connected by running wires through the adjacent channel.
- To connect cells in non-adjacent rows, vertical routing space on the sides is used or special feed through cells are instantiated.
- Rat's nest estimates are a good way to judge wire congestion and long nets.
- It is possible that in some rows, some space may be left unused, as either no cell would fit there or routing considerations dictate placing cells elsewhere.
- White space can be used for filler-cap cells.
- Spare-cells are usually sprinkled through rows.

Standard Cell Layout

- Cells are of same height but different width.
- Layout is organized in rows of cells separated by routing channels.
- Routing channels are not fixed-height. This is dependent on the channel routing density.

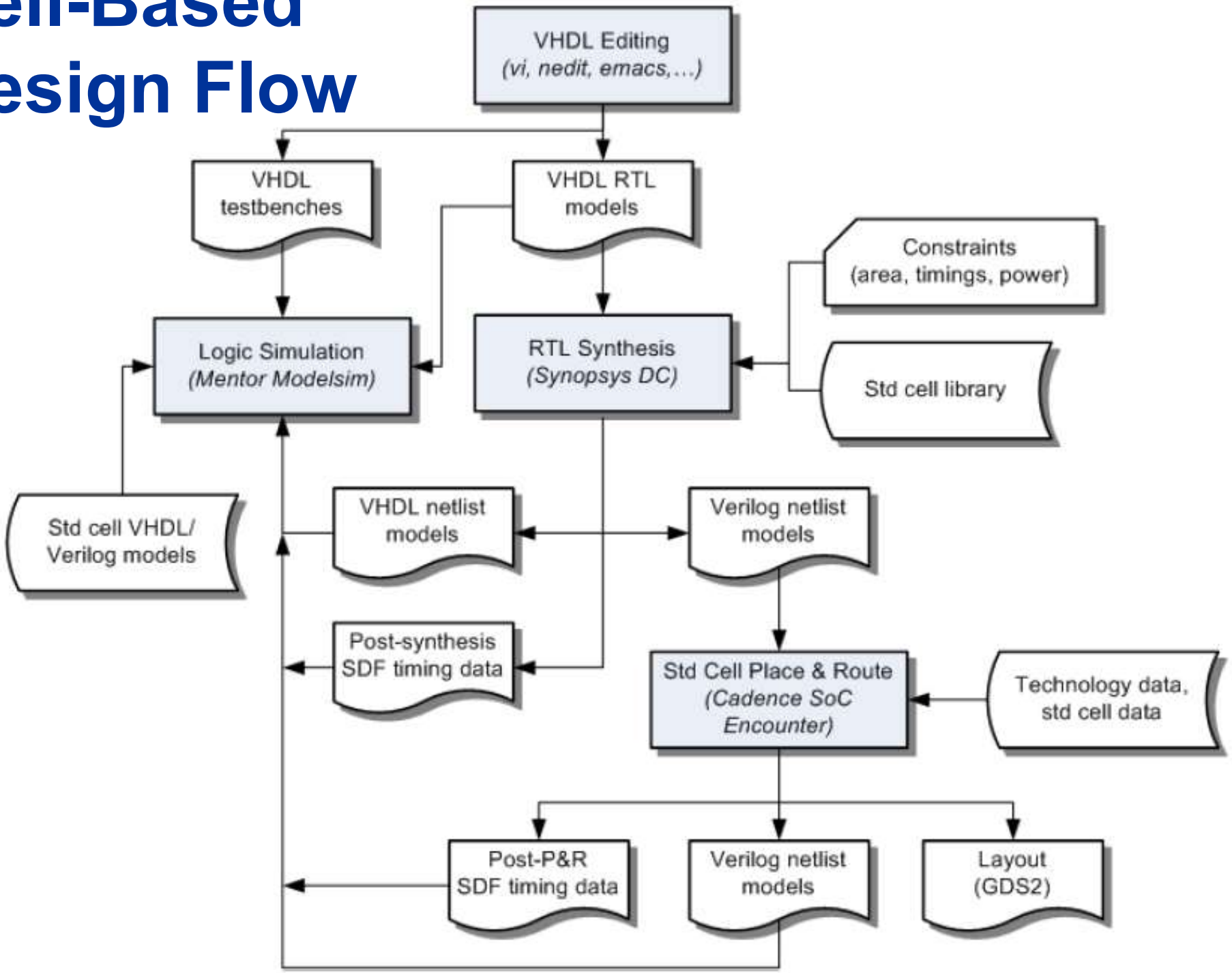




Standard Cells (cont.)

- **Cell library holds relevant information about cells.**
  - E.g., name, functionality, delays, resistance, capacitance, layout, area, pin topology, etc.
  - All cells in a library have same standardized layouts, i.e., all cells have the same height, rails placement and widths, etc.
- **Cell libraries work with a variety of tools.**
  - Cell Library format and content is defined by Synth tool.
  - The library contains information on how to scale the cell parameters with variation in process parameters and operating conditions.
  - The cell parameters are designed with some safety margin.
  - Cells that would not work in a wide range of parameters are simply excluded from the library. This often includes dynamic logic.

Cell-Based Design Flow

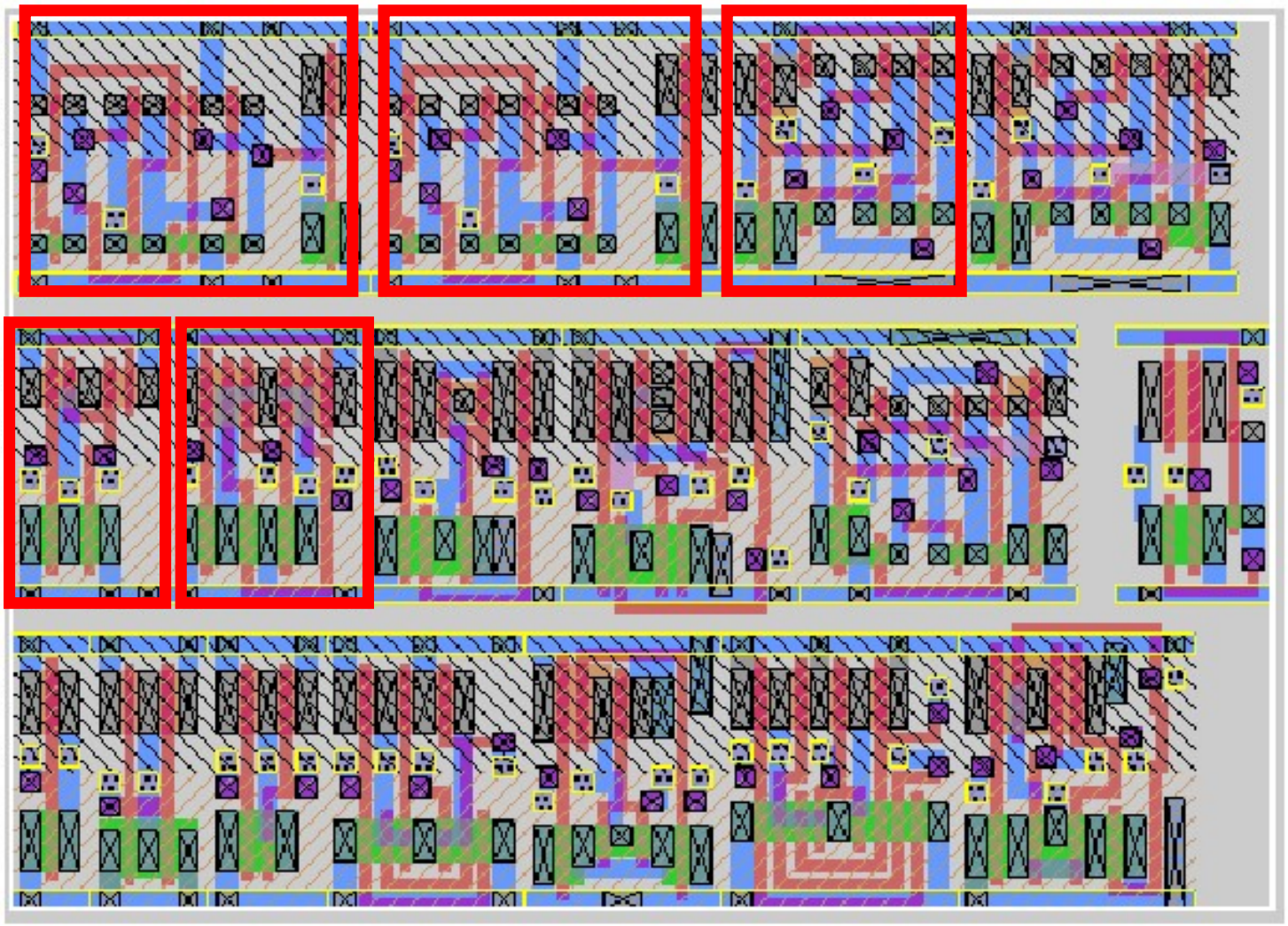


Standard Cells (cont.)

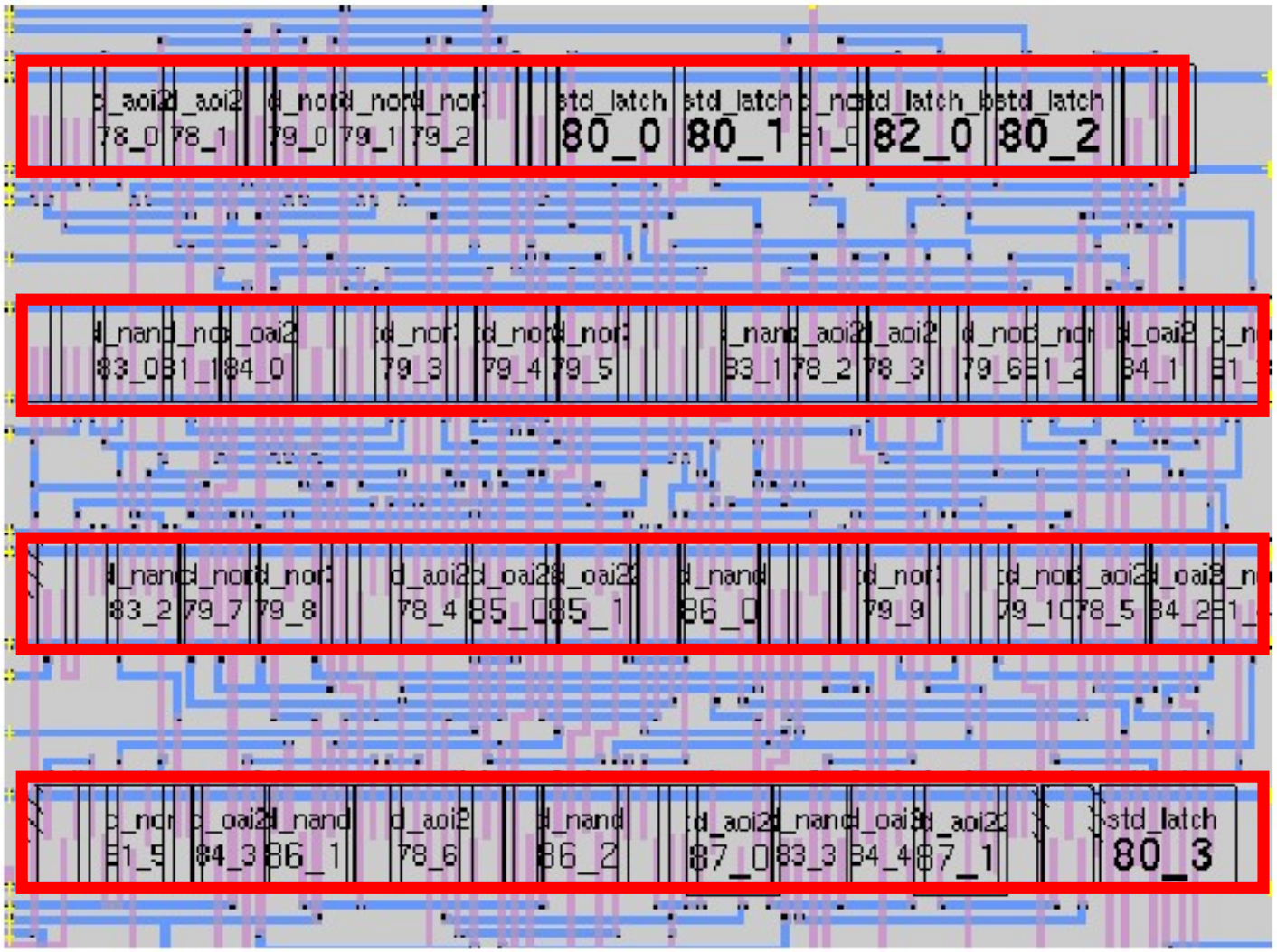
- **Hierarchical cells**
  - Higher level functions can be composed from the primitive elements.
  - Porting the library to newer technology is a non-trivial task.
  - Technology independent libraries are being proposed, which can be semi-automatically ported.

- **Smaller die size but higher manufacturing cost**
  - Compared to array based designs.
  - Gate array packages come in discrete sizes and designs commonly leave out some gates unused.
  - Gate arrays also have limited routing capacity reducing utilisation.

Standard Cells (cont.)

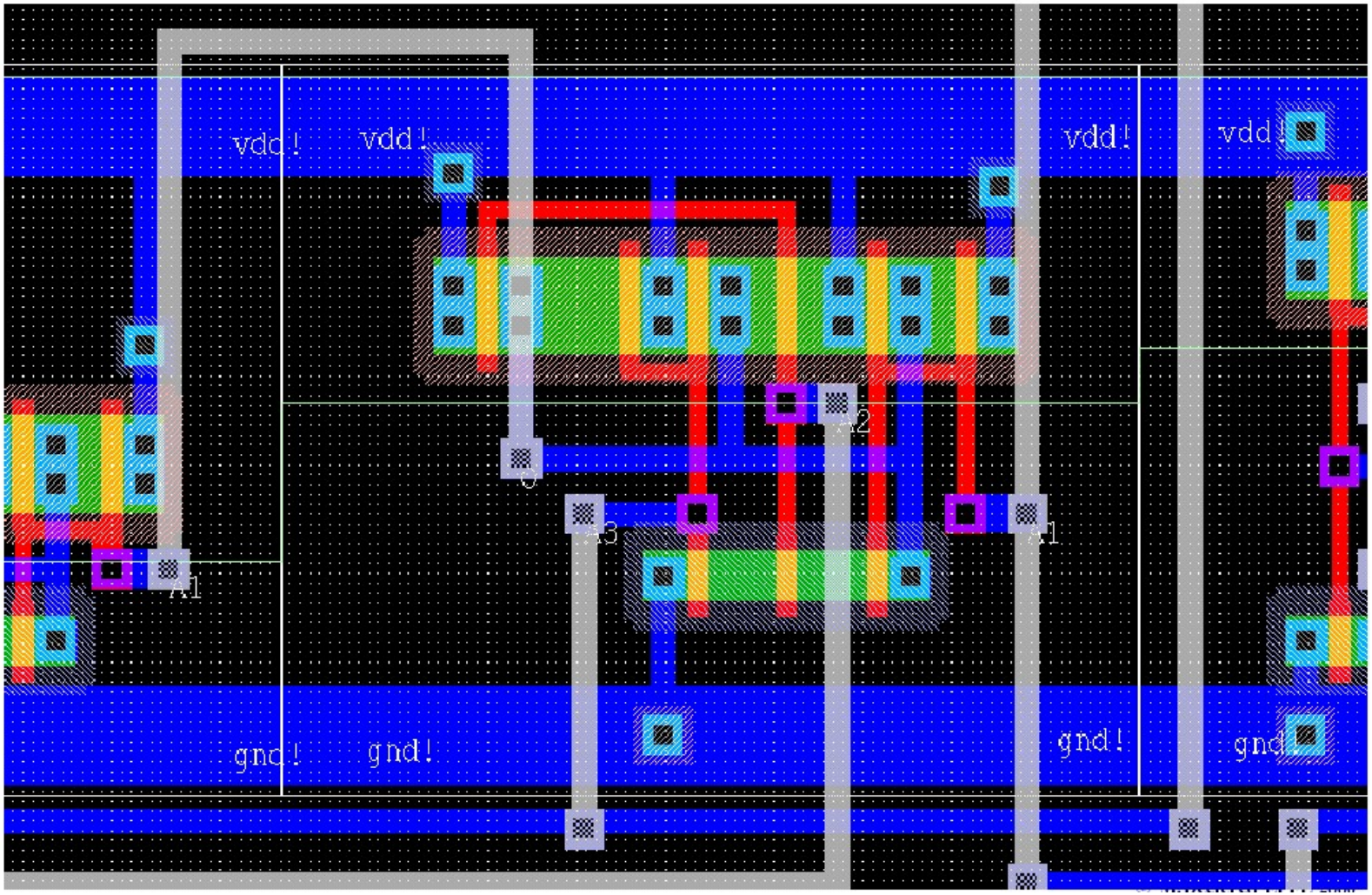


Standard Cells (cont.)

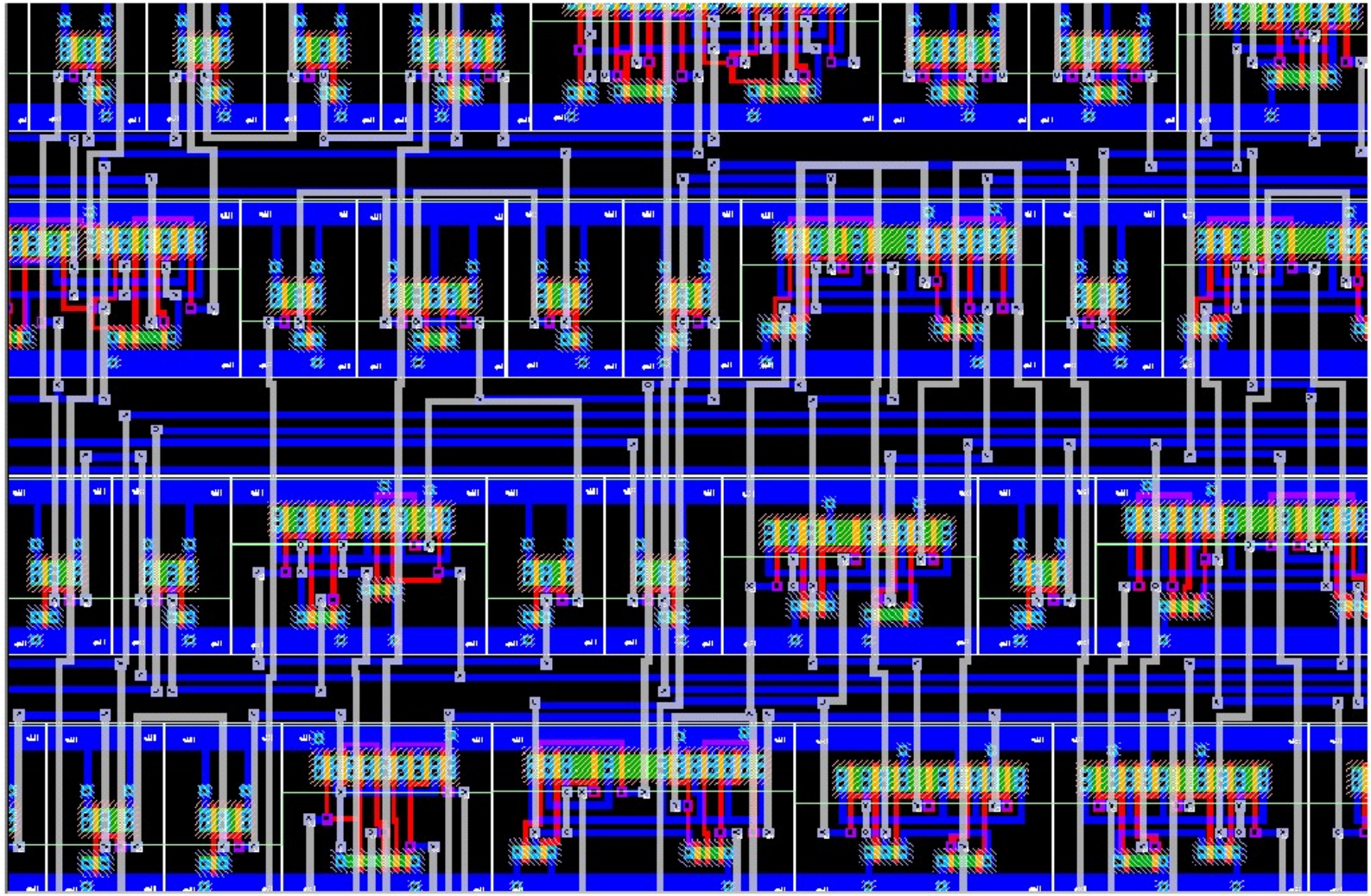


“routing” between the cells completed in the channel

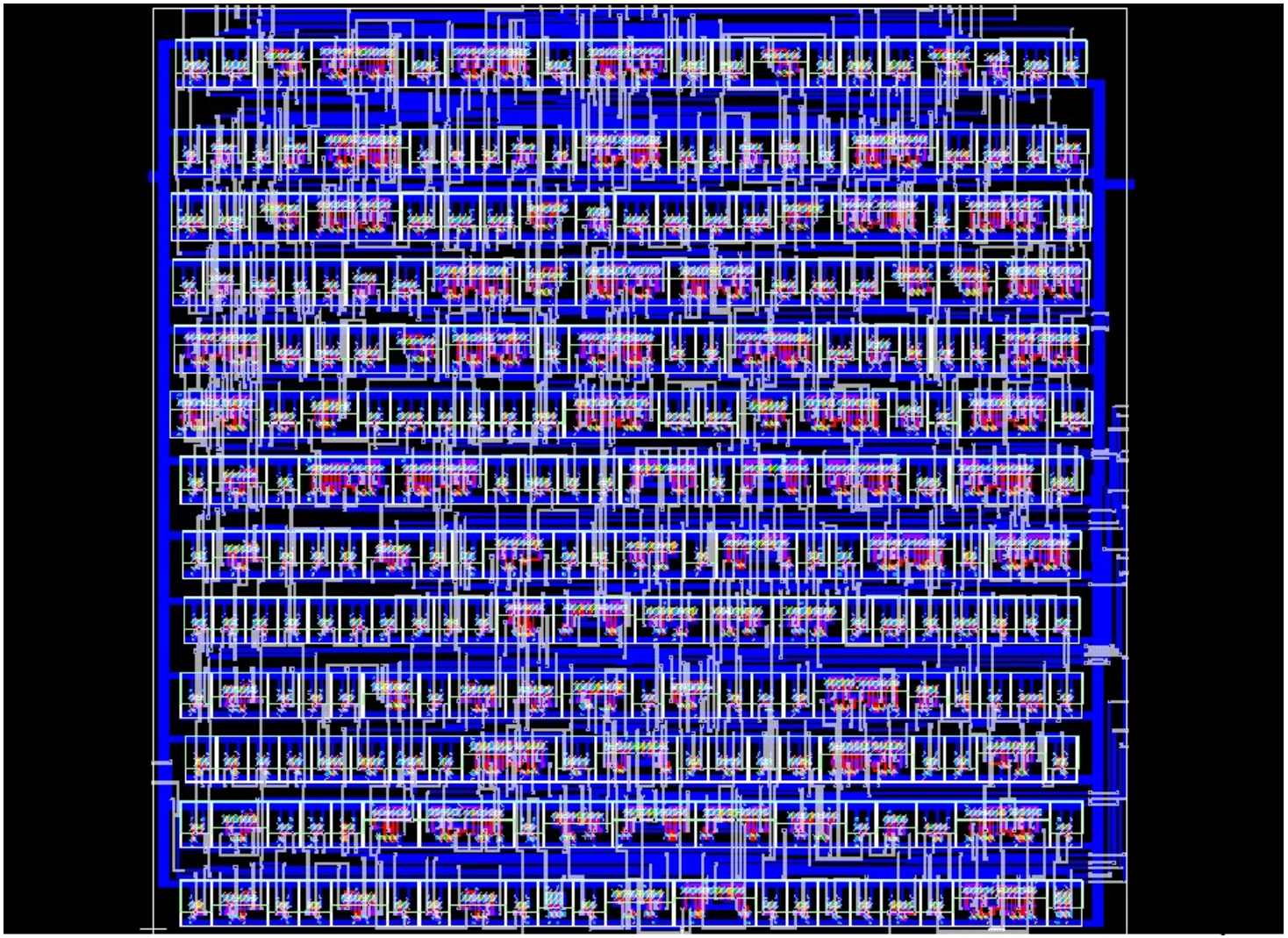
Standard Cells (cont.)



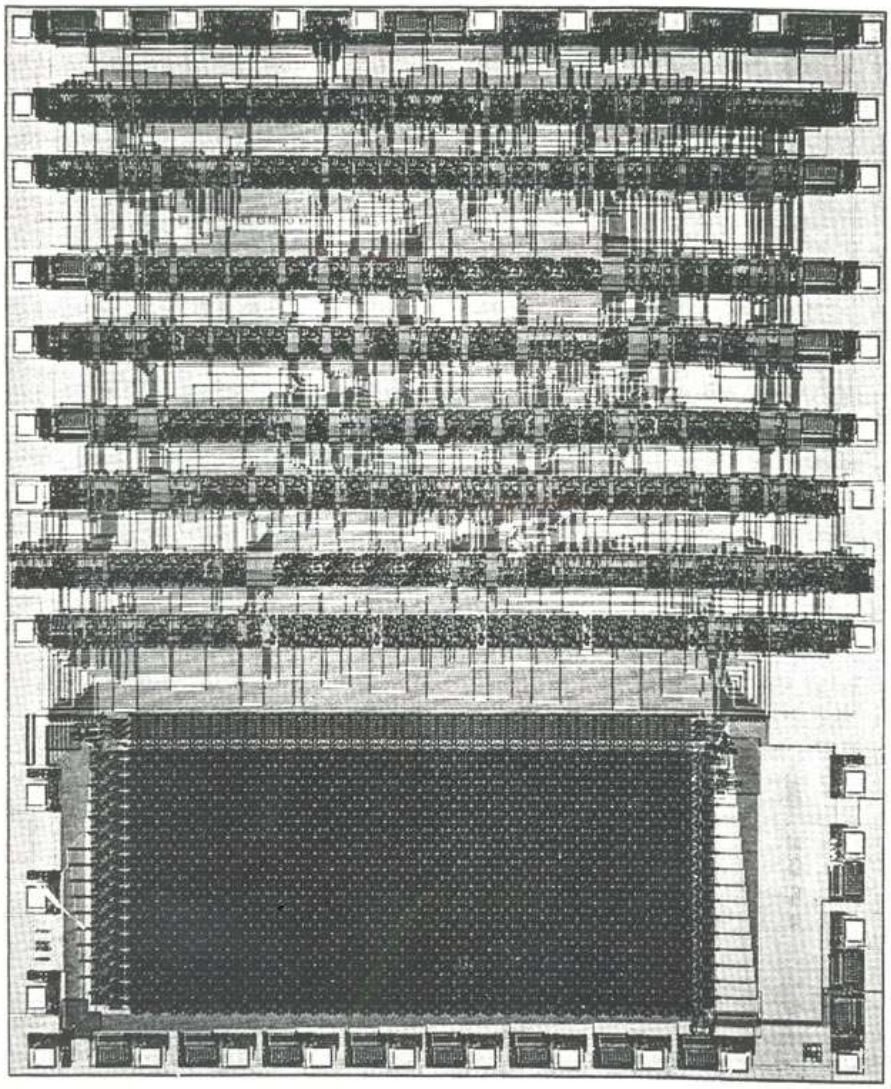
Standard Cells (cont.)



Standard Cells (cont.)



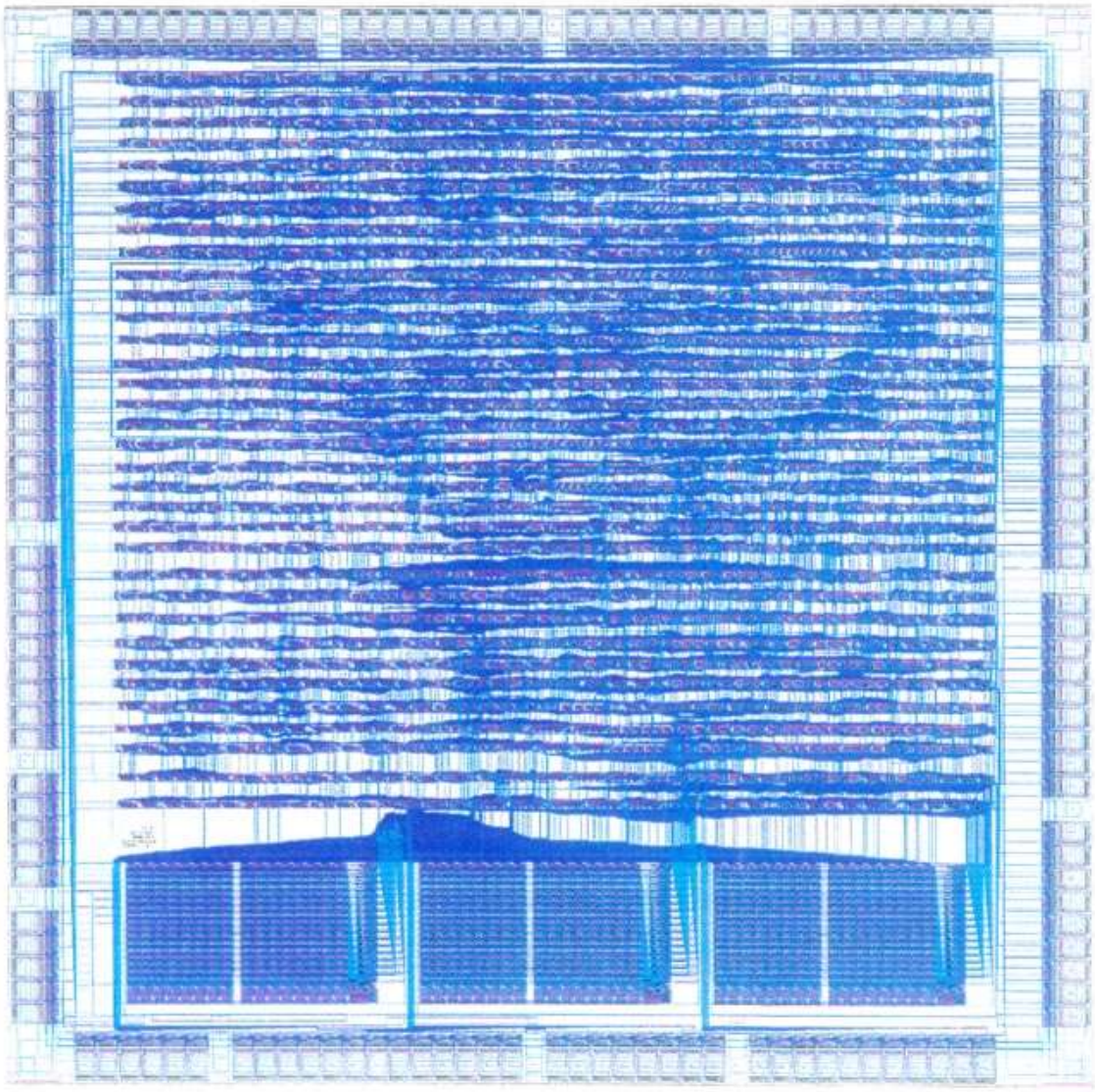
Standard Cells (cont.)



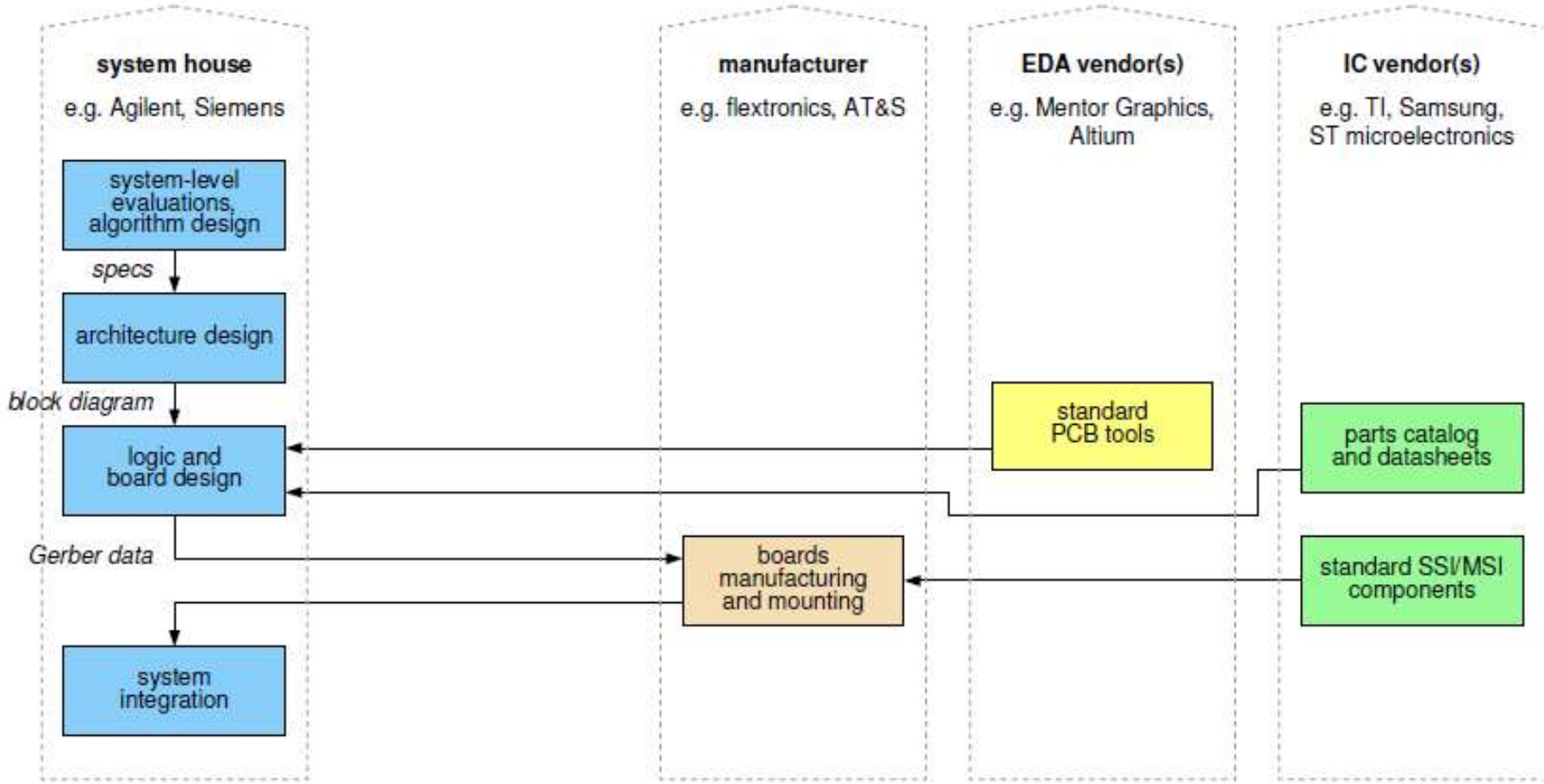
Rows of standard cells with routing channels between them

Memory array

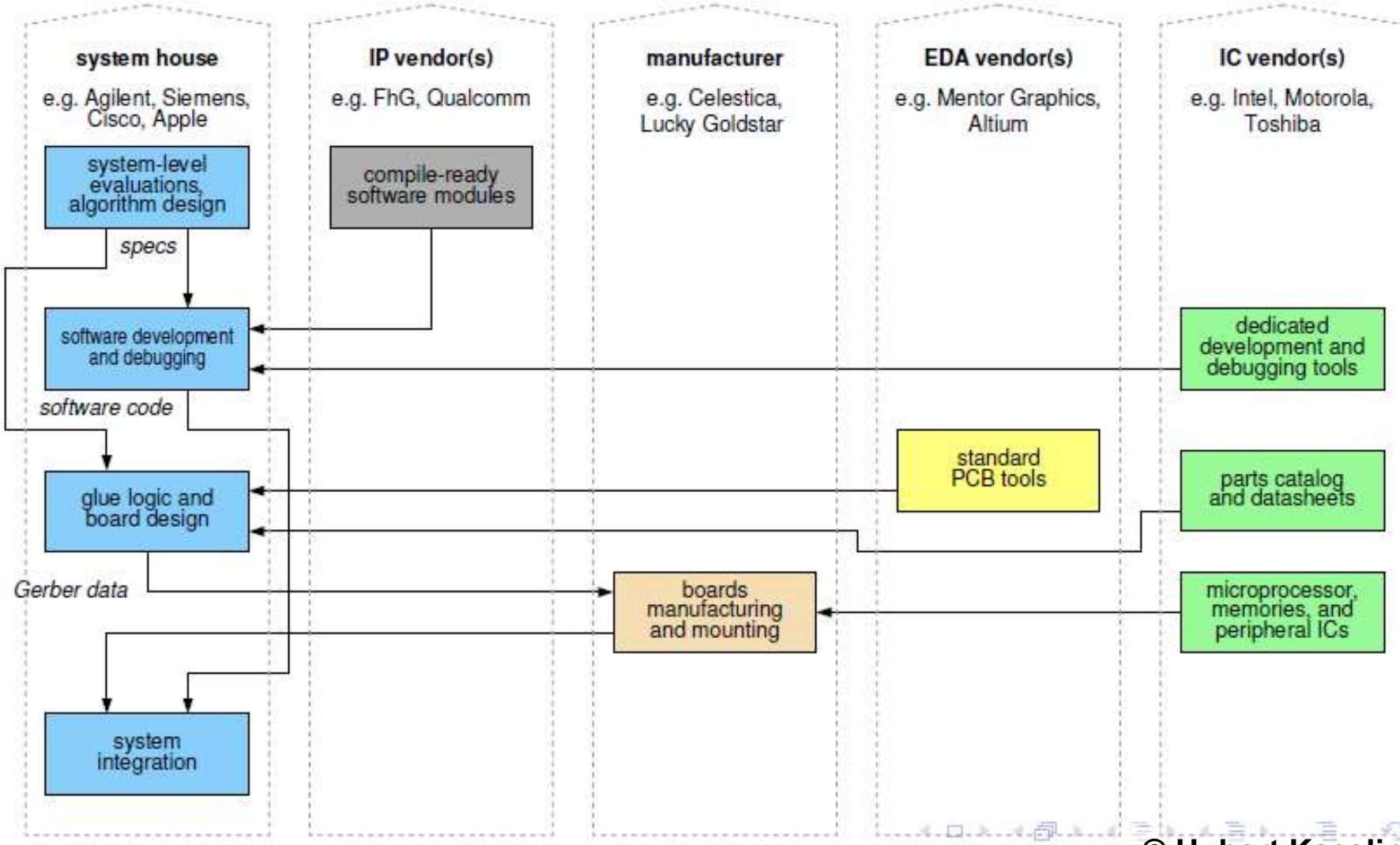
Standard Cells (cont.)



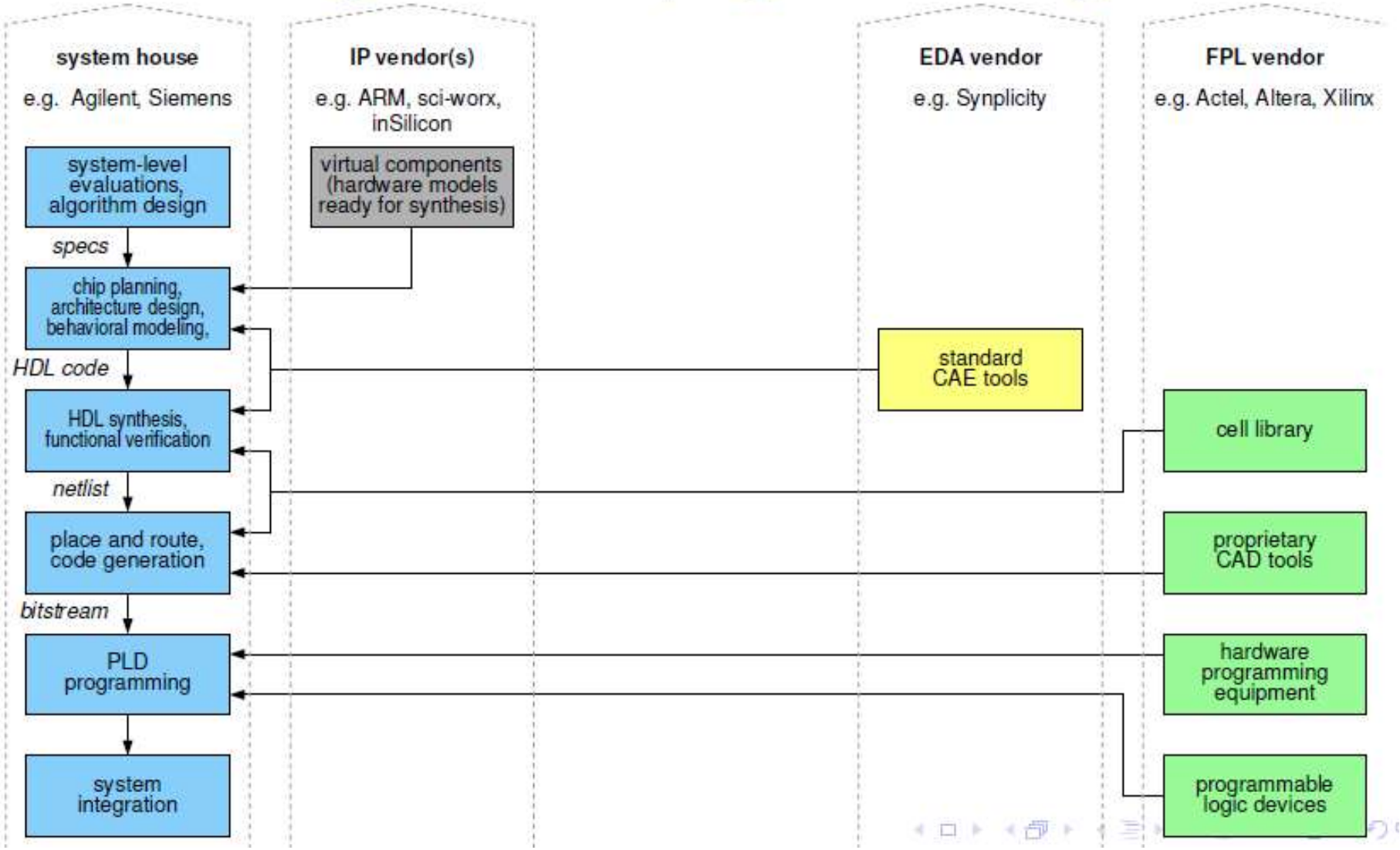
Roles and Business Models: Standard Parts



Roles and Business Models: With Processors

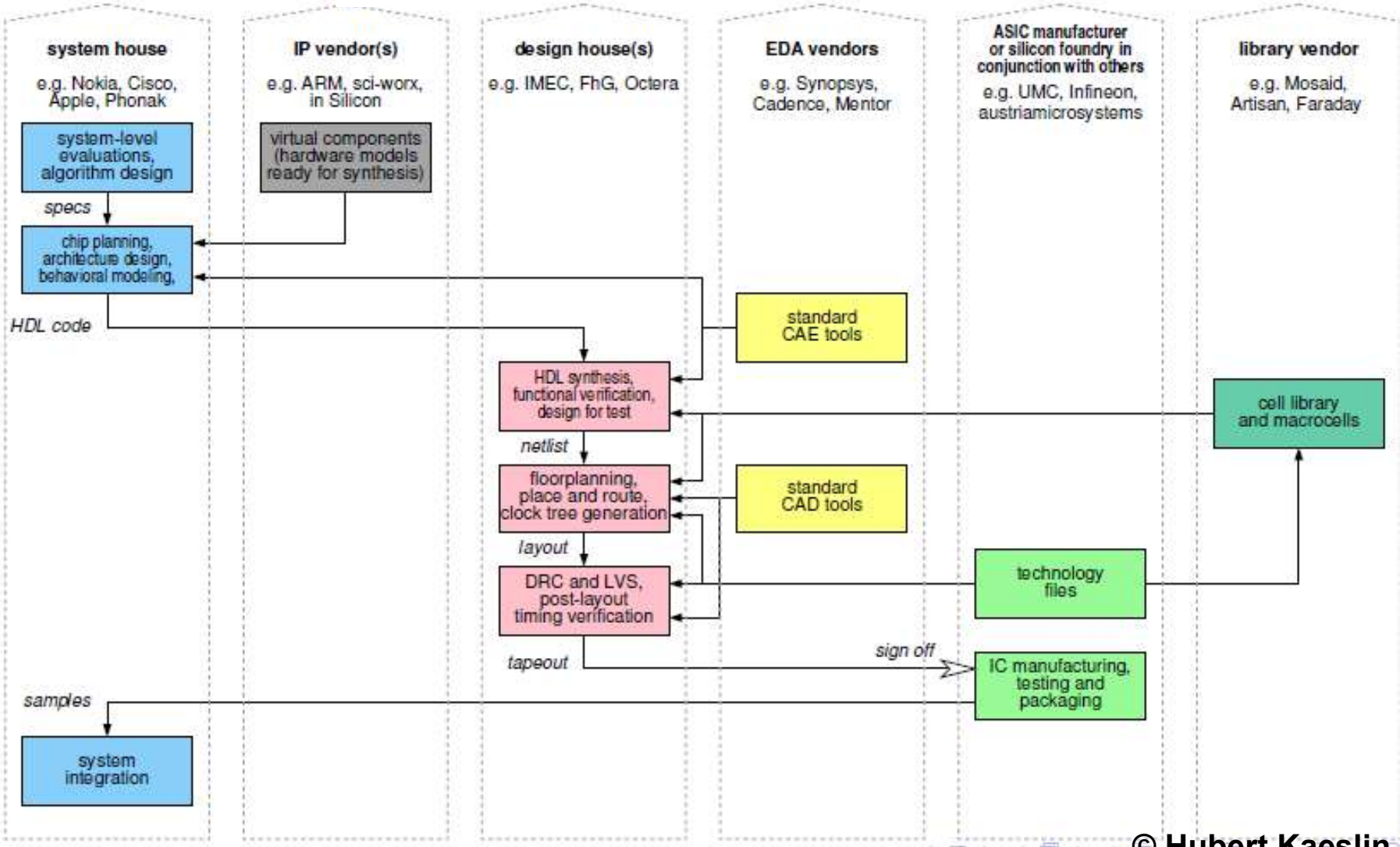


Roles and Business Models: With FPGA/PLDs



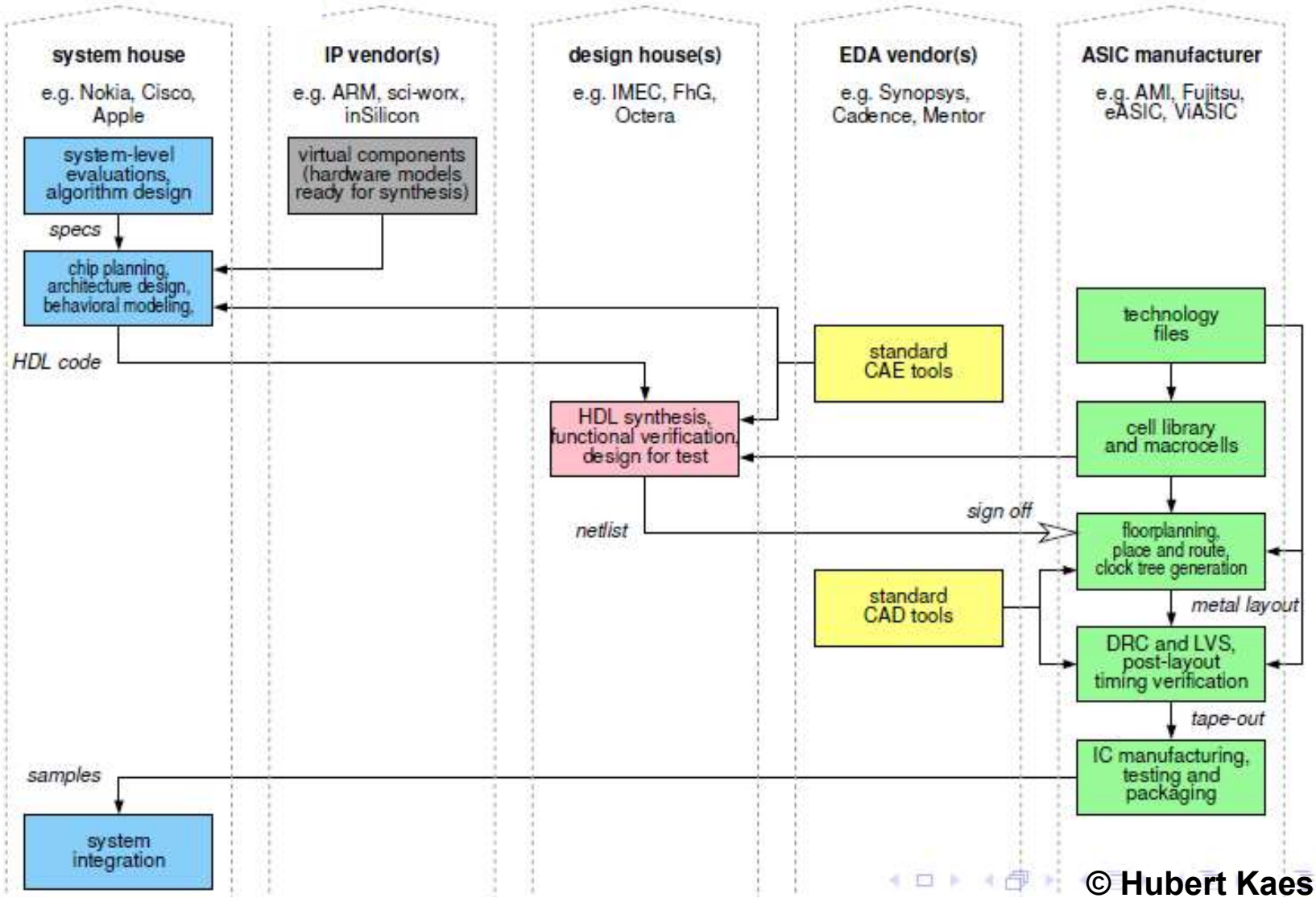
© Hubert Kaeslin

Roles and Business Models: With ASICs



© Hubert Kaeslin

Roles and Business Models: With ASICs

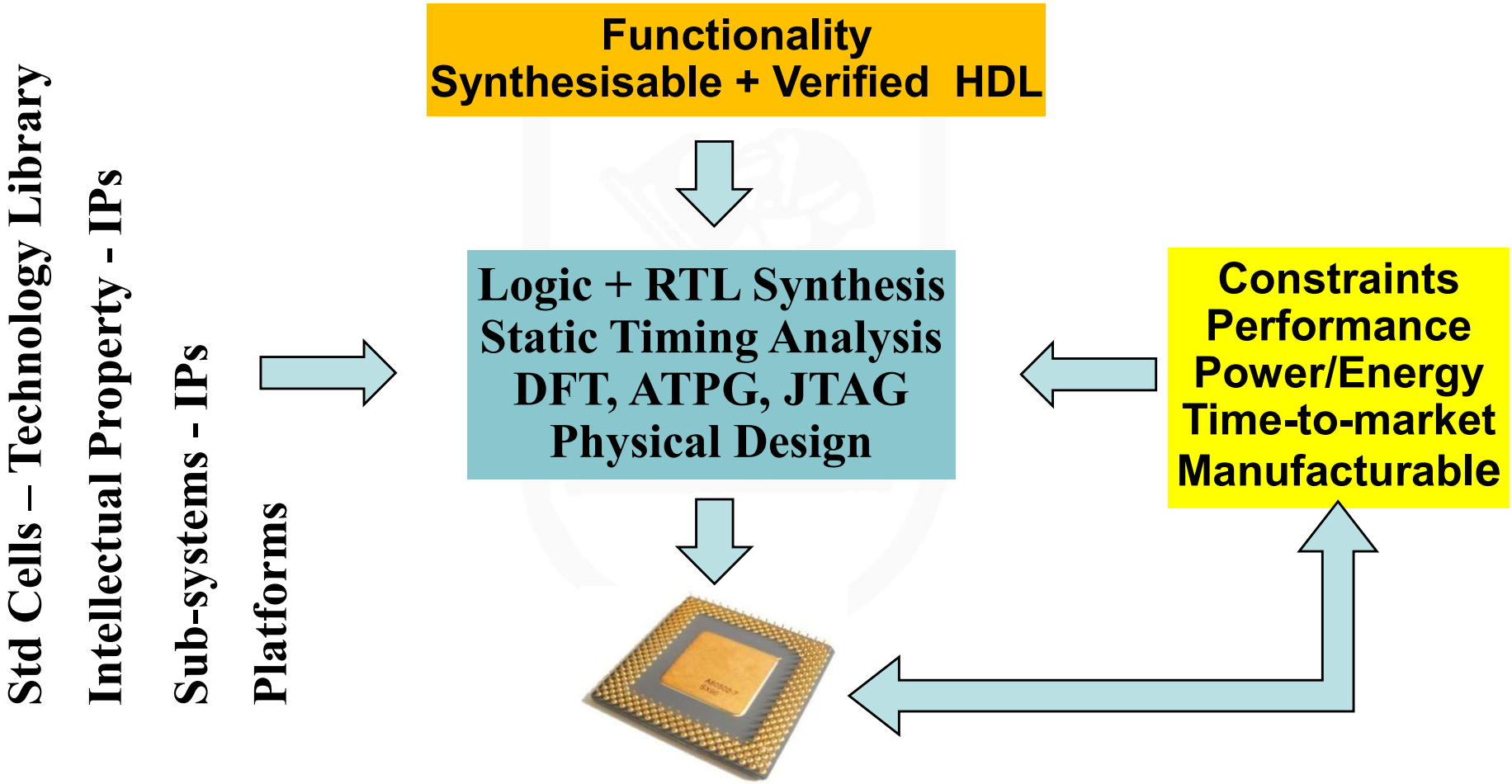


© Hubert Kaeslin

IC Design Flows

- The MOS Transistor
- Analog and Circuit Design
- Digital Logic Families
- Productivity Gap
- Digital Design Flows

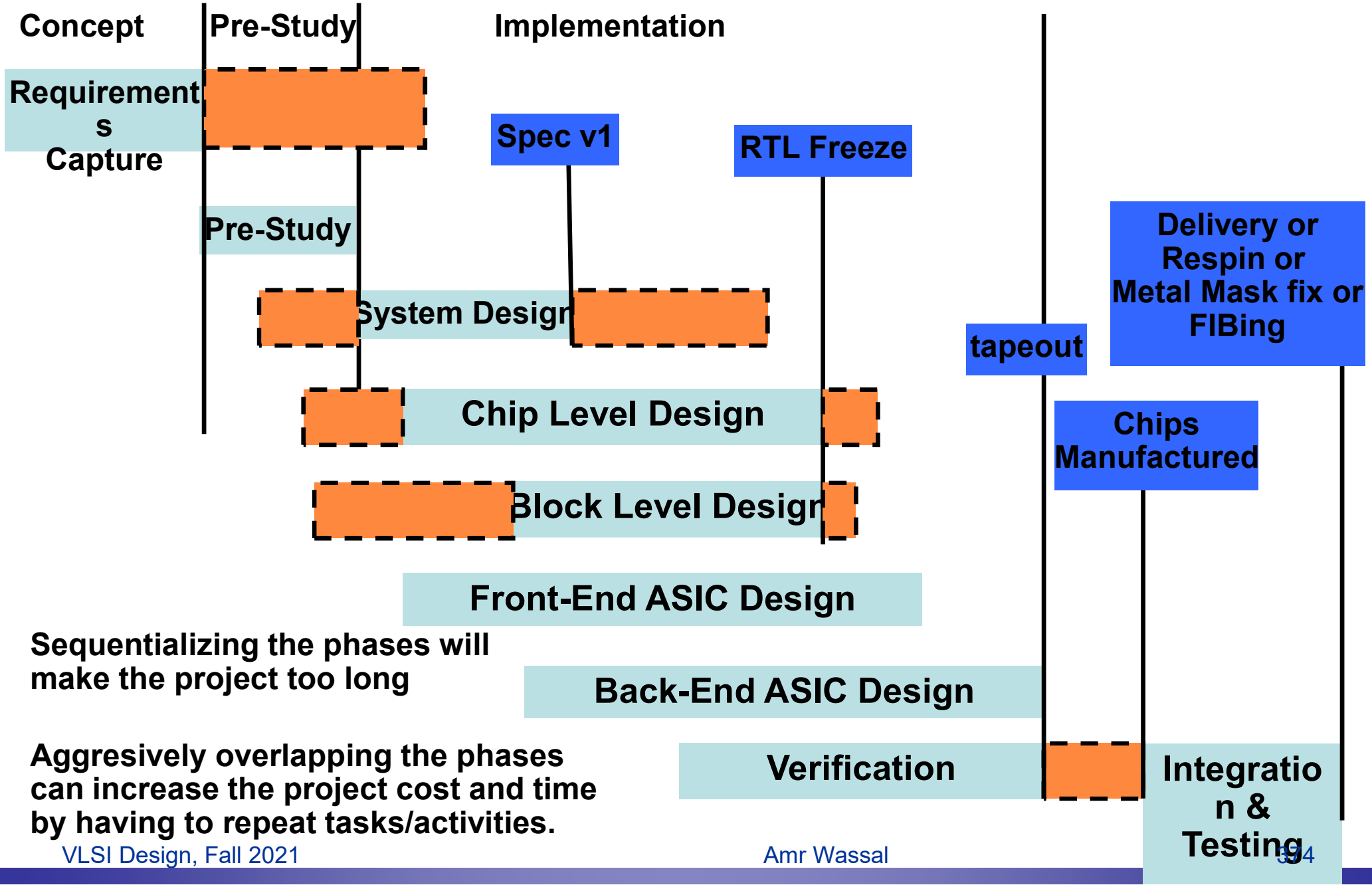
What the ASIC flow is about ?



ASIC Project Phases

- **Requirements Capture**
  - Market Survey, Business Opportunities, Customer Interaction
  - Requirements Specification – CRS (Customer Requirements Specification)
    - Hierarchical list of requirements – each tagged with an id.
    - Peformance, Cost and Delivery Schedule
- **PreStudy**
  - Estimate Die Size, Package, Power Consumption, Performance, Development, Manufacturing, Integration and Test Times
  - Buy or Develop Decisions
  - Explore architectural and algorithmic alternatives
  - Interact with Customer, Marketting and refine Requirements doc.
  - Quit or go ahead decision
- **System Design**
- **Implementation – Design and Verification**
- **Integration & Test**
- **Maintenance**

ASIC Project Phases (cont.)



Sequentializing the phases will make the project too long

Aggresively overlapping the phases can increase the project cost and time by having to repeat tasks/activities.

ASIC Specification & Implementation

- **Multiple Styles, Notations, Languages**
- **Usage depends on the phase of ASIC project and the tool flow**
- **Some specifications are manually entered, others are automatically generated**
- **Top Level**
  - Manually entered
  - Schematic Capture style stitching getting increasingly popular – look at Platform Express from Mentor Graphics, Core Assembler from Synopsys.
- **Block Level: Either IP/Sub-System or an algorithm**
- **Many abstraction levels involved in ASIC projects**
- **Two primary synthesis class are used in industry**
  - RTL/Logic Synthesis – the mainstay of the industry
  - Algorithmic Synthesis – getting rapidly accepted

Abstractions and their usage

- **Functional Level**

  e.g., SORT function. Only the input, output relationship is specified. Used heavily in System Design Phase, Algorithmic Development

- **Algorithmic Level/Behavioural Level**

  e.g., Quick Sort, Linear Sort etc. Only relative order of operations is specified. Used in system model as an executable specification. Input to HLS tools

- **State Machine, Register Transfer Level, Behavioural RTL**

  Absolute time is assigned for operation execution. Implementation of combinatorial logic: next state function, arithmetic operators etc. is not specified. The main abstraction level for synthesisable hardware specification. Output of HLS tools

- **Logic/Gate Level**

  Combinatorial Logic for operations is now specified but their realisation at device level is not specified. Specification of micro-architectural building blocks like arithmetic blocks, registers, muxes etc.

- **Device Level**

  Specification of standard cell libraries.

Domains and their Usage

```
if IR(3) = 0'then'
    PC      := PC + 1;
else
    DBUF    := MEM(PC);
    MEM(SP) := PC + 1;
    SP      := SP - 1;
    PC      := DBUF;
end if;
```

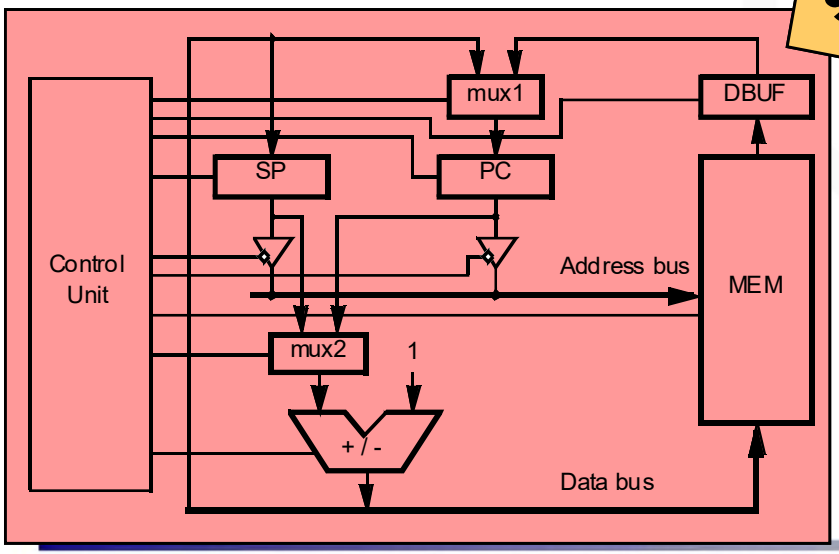
Behavioural,
Algorithmic/RTL specification as input to synthesis tools

Behavior

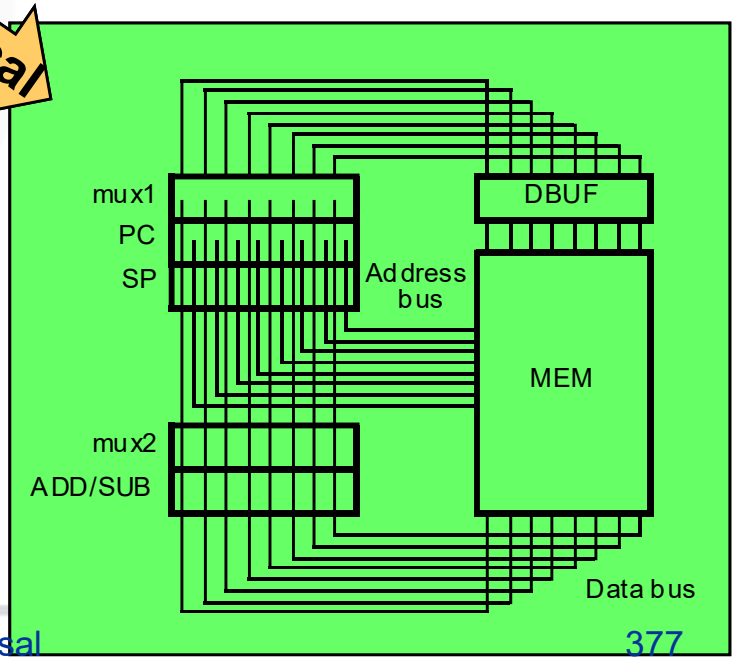
Structure

Physical

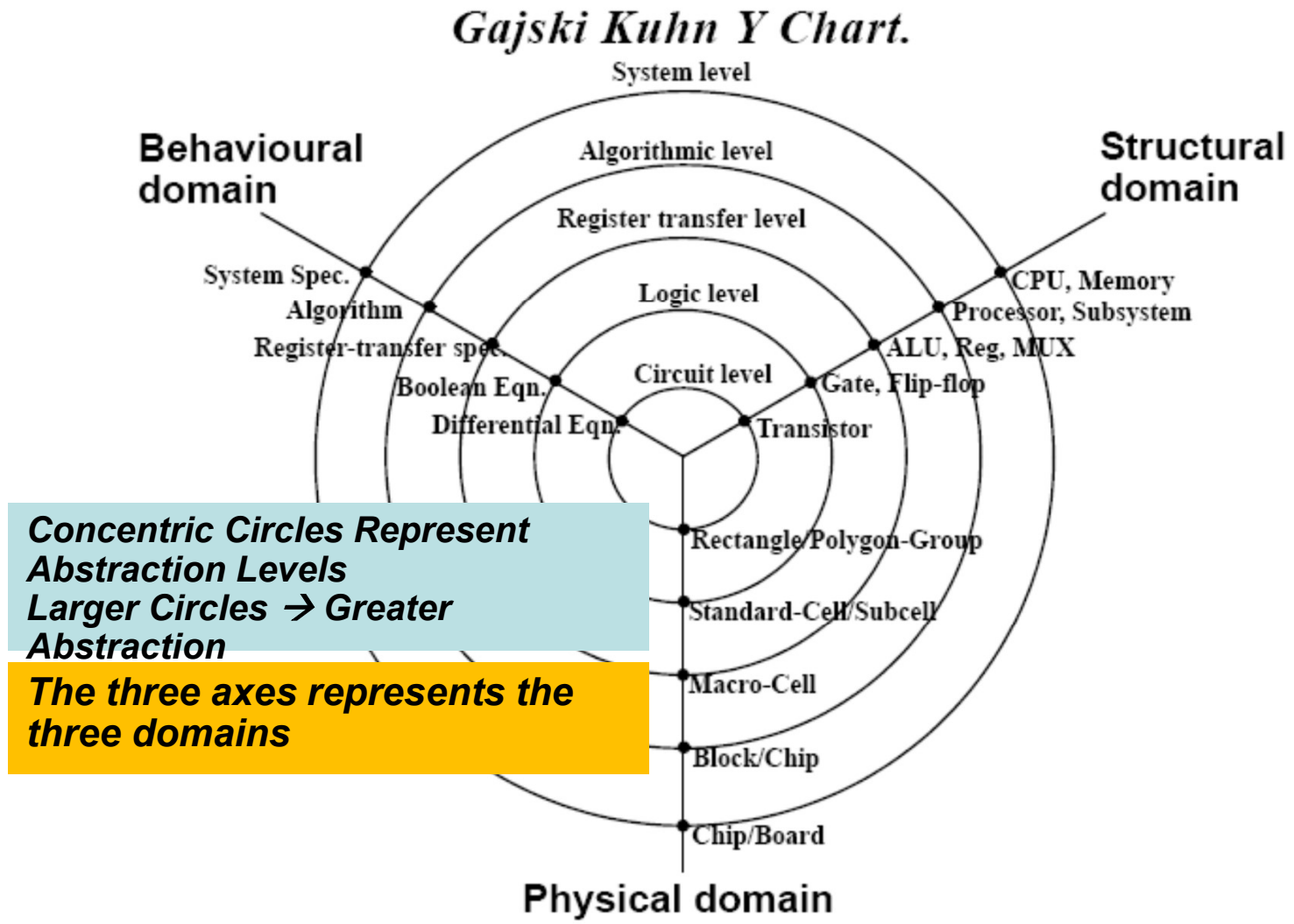
Structural
Used to specify toplevel, Output of Synthesis tools



Physical
Used to specify Floorplan, output of physical synthesis too



The Taxonomy of VLSI Design Space

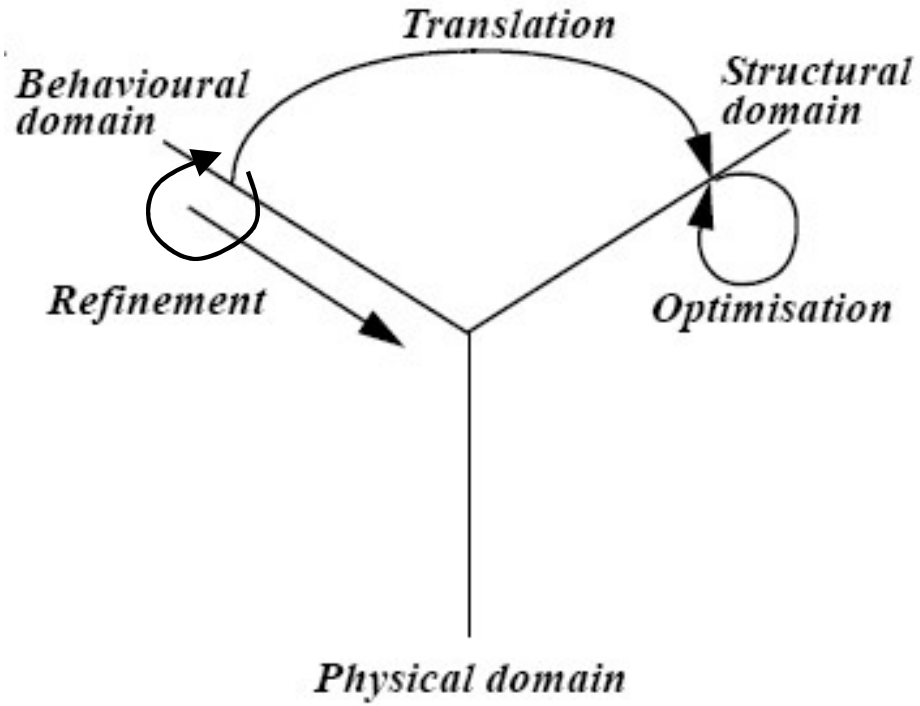


Concentric Circles Represent Abstraction Levels
Larger Circles → Greater Abstraction

The three axes represents the three domains

Synthesis & Analysis

Synthesis is essentially refining a more abstract specification to a relatively more detailed specification.

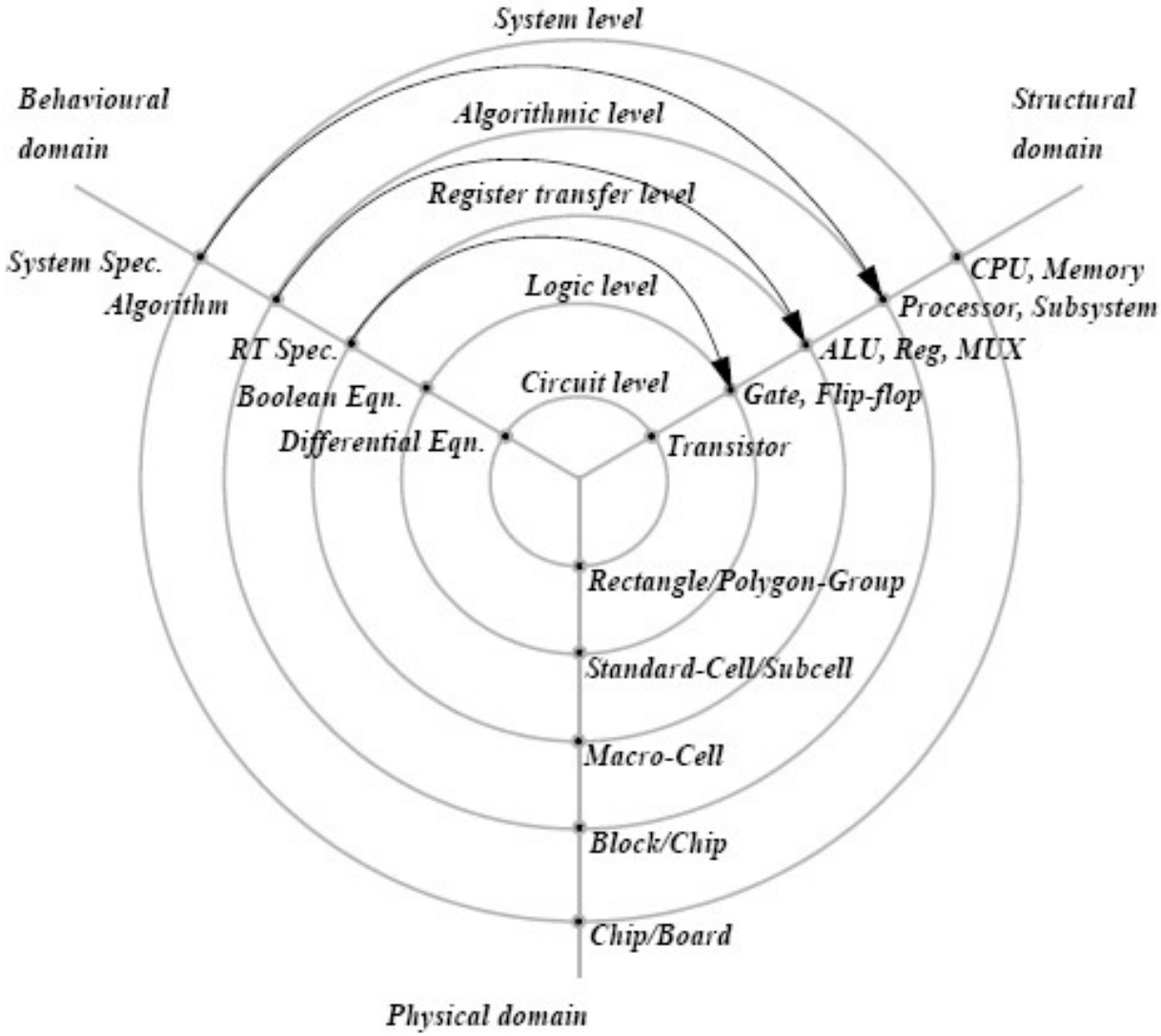


Synthesis
=
Domain Translation
+
Refinement:
Higher Abstraction to Lower Abstraction
+
Optimisation

Analysis is the opposite of synthesis. It involves analysing the detailed specification like gate level specification to extract a more abstract timing information. Static Timing Analysis(STA).

Types of Synthesis

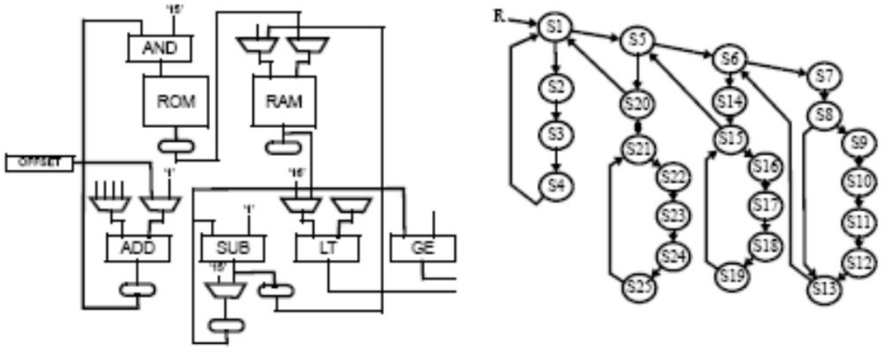
- *Behaviour to Structure*
  - *System*
  - *Algorithmic*
  - *FSM/Logic*
  - *Circuit*
- *Structure to Layout*
  - *System Partitioning*
  - *Chip Floorplanning*
  - *Module generation*
  - *Cell generation*



Algorithmic Level Synthesis

```
for i := 2 TO MAX do
    for j := MAX downto i DO
        if input [j-1] > input [j ] then
            begin
                tmp := input [ j -1 ];
                input [ j - 1] := input [ j ];
                input [ j ] := tmp;
            end if
    end loop;
end loop;
```

Behavioural Synthesis.

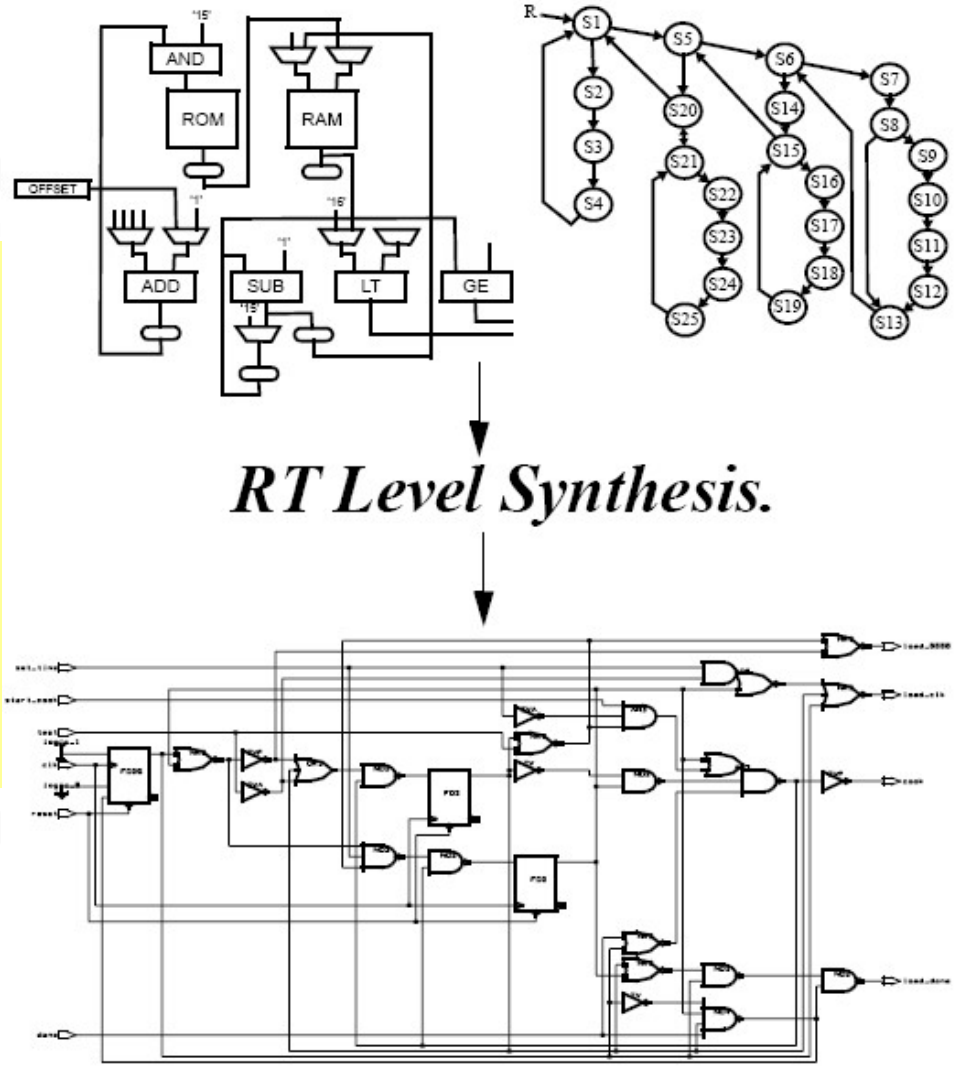


- *Also known as High Level Synthesis.*
- *CatapultC from Mentor, Behavioural Compiler from Synopsys, Forte Design Systems etc.*

- *Datapath*
  - *Scheduling*
    Assign operations to clock steps/FSM states
  - *Allocation*
    Allocate type and nr of resources
  - *Binding*
    Assign operations to resource instances
- *Control*
  FSM Synthesis

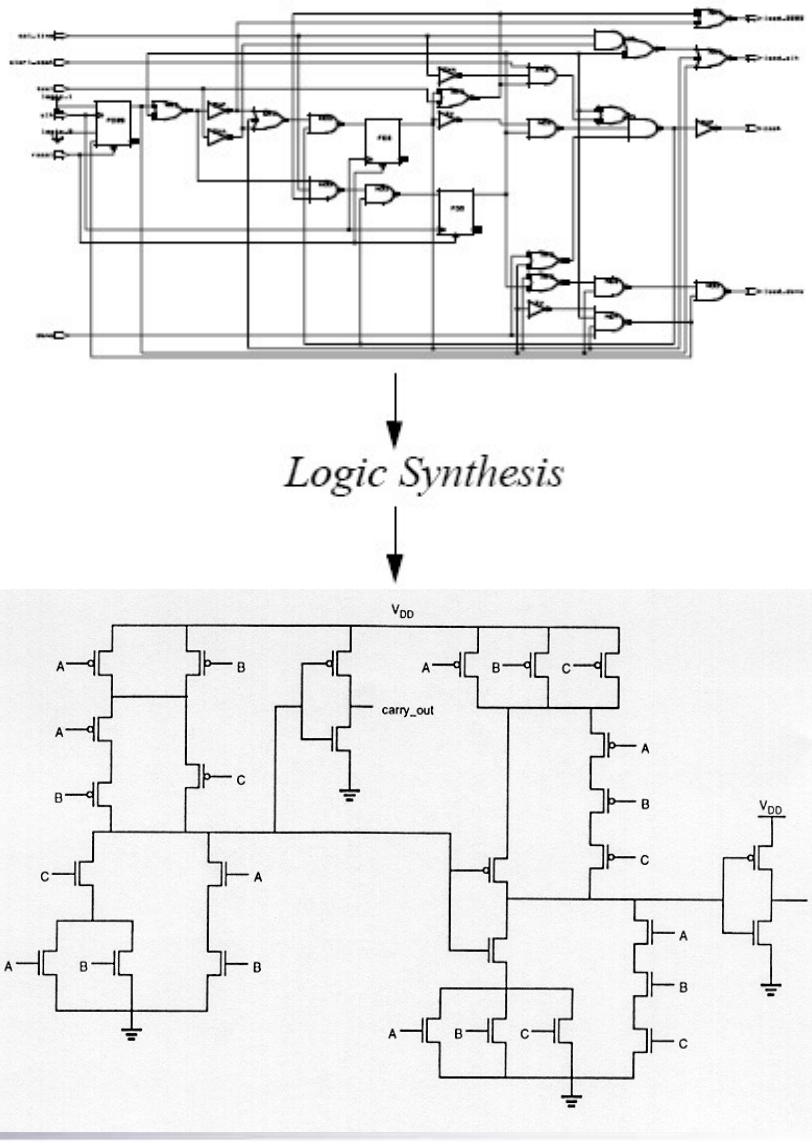
RTL Synthesis

- **State Minimisation**
  Merge compatible states
- **State Encoding**
  Encode states to minimise/speed optimise next state logic
- **Synthesis of Next State and Output Functions**
- **Functional Unit Synthesis**
  Arithmetic optimisation



Logic Synthesis

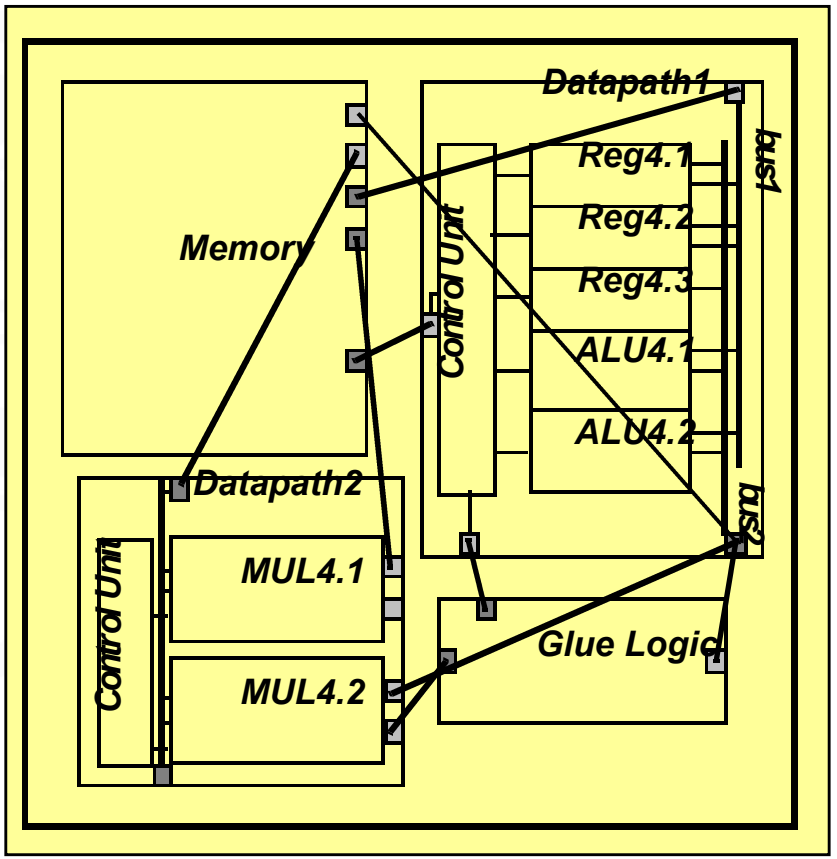
- **Boolean Minimisation**
- **Technology Mapping**
  - ***Realize boolean logic using minimal nr of cells from the technology library that does not violate any design rule and meets design goals.***
- **Can be viewed as the final step in RTL synthesis.**



Note that the CMOS circuit diagram does not correspond to the logic diagram above. Shown here merely as an example.

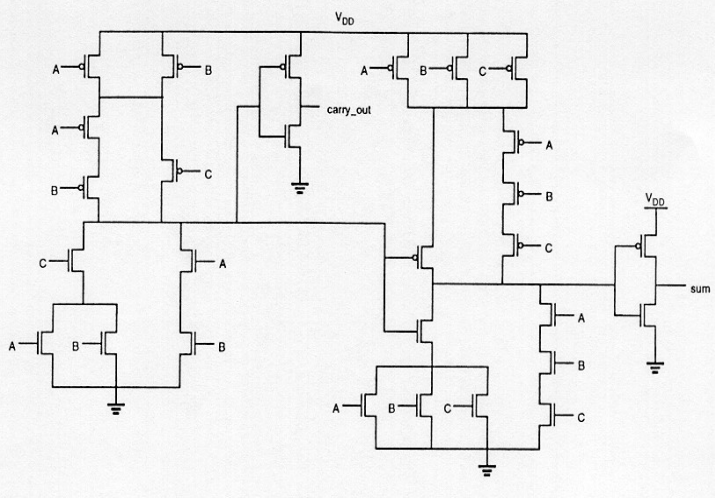
Floor-planning

- Task of deciding how the chip area is to be utilised.
- The leaf cells can be Memories, processor cores, data-paths and glue-logic and wiring necessary to connect them.
- Early floor-planning works with estimates of the component sizes and wire densities to provide an estimate of design area and its partitioning.
- May also provide some estimates of delays through critical nets and of clock skews.

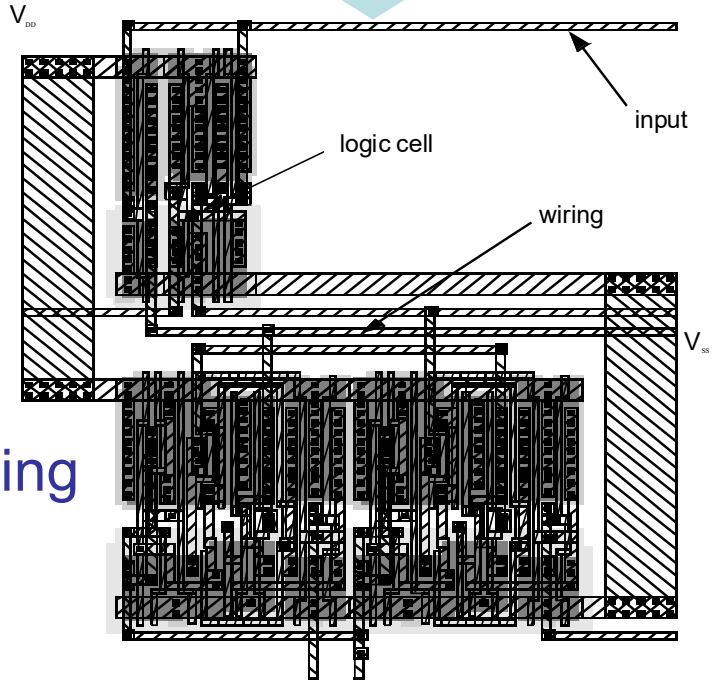


Physical Design

- **Placement**
  - Decides the positions of components.
  - Quality of placement is decided based on routing results.
- **Global Routing**
  - Assigns wires to channels defined during the floorplanning phase.
- **Detailed Routing**
  - Assigns wires to tracks.
- **Clock Tree Synthesis**
  - Distribute clock across chip while optimizing skew, clock insertion delay and power consumption.
- Timing-Driven P&R is the state-of-the-art.

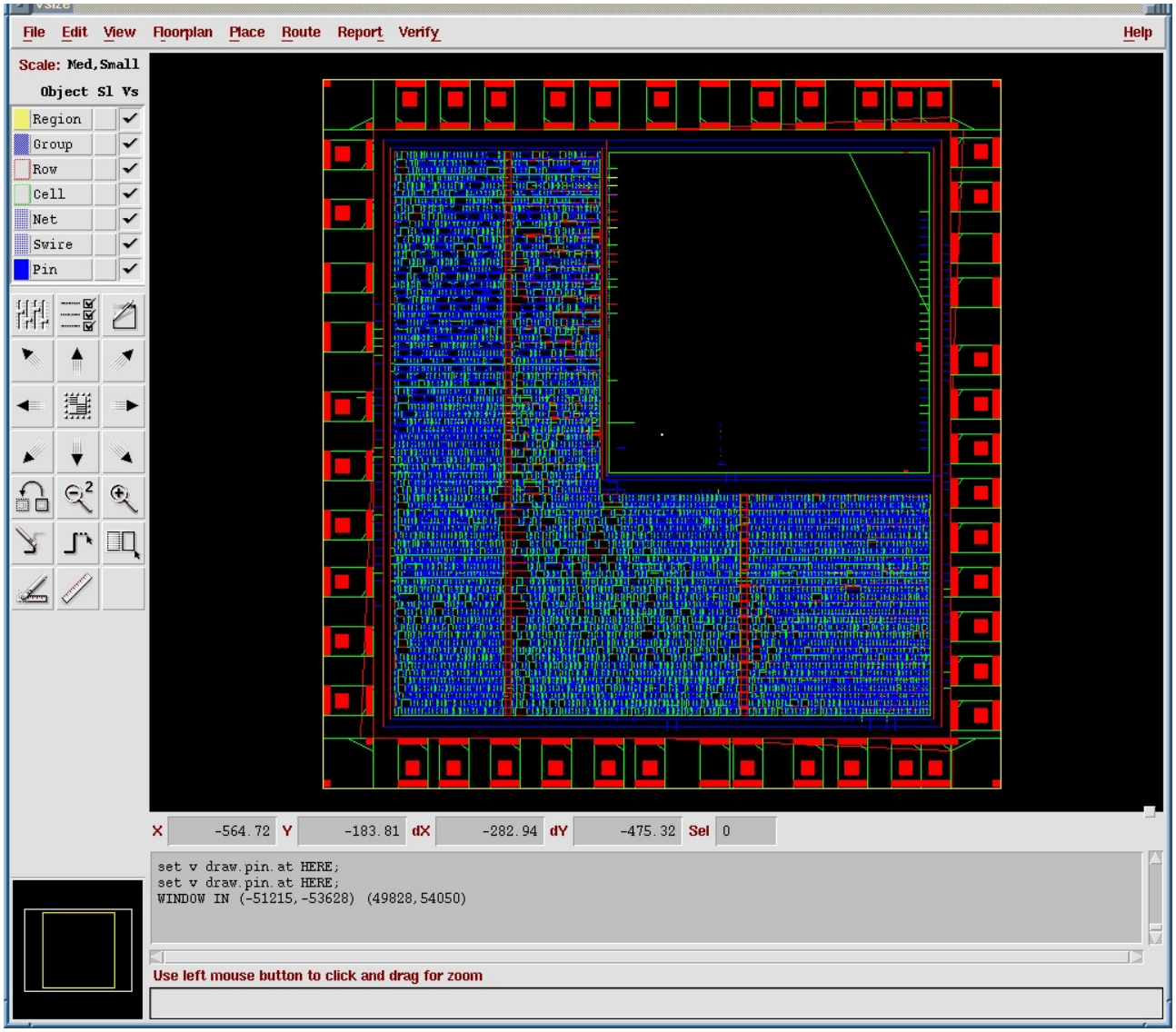


Physical Synthesis



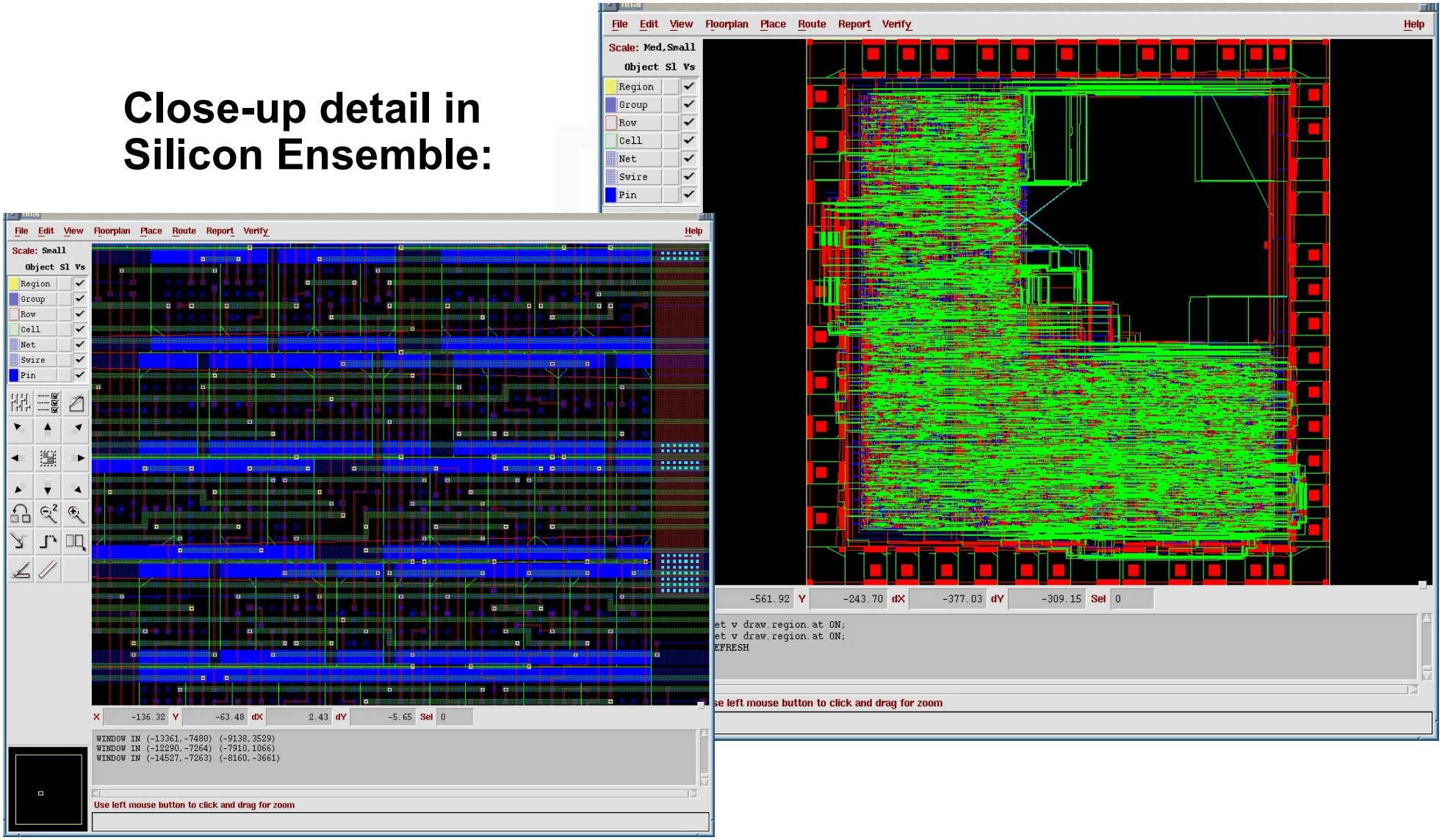
An example of standard cell layout.

Automatic Placement of Cells



Automatic Routing Between Cells

Close-up detail in Silicon Ensemble:



Design Project Directory Structure

```
<project-name>/          # project directory home
   .synopsys_dc.setup    # setup file for Synopsys tools
   modelsim.ini          # setup file for Modelsim tool
   DOC/                  # documentation (pdf, text, etc.)
   HDL/                  # VHDL/Verilog source files
      GATE/              # gate-level netlists
      RTL/               # RTL descriptions
      TBENCH/            # testbenches
   IP/                   # external blocks (e.g., memories)
   LAY/                  # full-custom layout files
   LIB/                  # design libraries
      MSIM/              # Modelsim library (VHDL, Verilog)
      SNPS/              # Synopsys library (VHDL, Verilog)
   PAR/                  # place & route files
      BIN/               # commands, scripts
      CONF/              # configuration files
      CTS/               # clock tree synthesis files
      DB/                # database files
      DEX/               # design exchange files
      LOG/               # log files
      RPT/               # report files
      SDC/               # system design constraint files
      TEC/               # technology files
      TIM/               # timing files
   SIM/                  # simulation files
      BIN/               # commands, scripts
      OUT/               # output files (e.g., waveforms)
   SYN/                  # synthesis files
      BIN/               # commands, scripts
      DB/                # database files
      RPT/               # report files
      SDC/               # system design constraint files
      TIM/               # timing files
   TST/                  # test files
      BIN/               # commands, scripts
      RPT/               # report files
      TV/                # test vectors
```

Basic Design Flow Steps

Step 1) VHDL model editing (tool: text editor)

Step 2) Pre-synthesis VHDL simulation (tool: Modelsim) *[2.2]*
- 2.1) Compilation of the RTL VHDL model and related testbench
- 2.2) Simulation of the RTL VHDL model

Step 3) Logic synthesis (tool: Synopsys Design Compiler)
- 3.1) RTL VHDL model analysis *[3.2]*
- 3.2) Design elaboration (generic synthesis) *[3.3]*
- 3.3) Design environment definition (operating conditions, wire load model) *[3.4]*
- 3.4) Design constraint definitions (area, clock, timings) *[3.5]*
- 3.5) Design mapping and optimization (mapping to gates) *[3.6]*
- 3.6) Report generation *[3.7]*
- 3.7) VHDL gate-level netlist generation *[3.8]*
- 3.8) Post-synthesis timing data (SDF) generation for the VHDL netlist *[3.8]*
- 3.9) Verilog gate-level netlist generation *[3.8]*
- 3.10) Design constraints generation for placement and routing *[3.9]*

Step 4) Post-synthesis VHDL simulation (tool: Modelsim) *[2.3]*
- 4.1) Compilation of the VHDL/Verilog netlist and related testbench
- 4.2) Simulation of the post-synthesis gate-level netlist with timing data

Step 5) Placement and routing (tool: Cadence Encounter)
- 5.1) Design import (technological data + Verilog netlist) *[4.2]*
- 5.2) Floorplan specification *[4.3]*
- 5.3) Power ring/stripe creation and routing *[4.4]*
- 5.4) Global net connections definition *[4.5]*
- 5.5) CAP cell placement *[4.6]*
- 5.6) Operating conditions definition *[4.7]*
- 5.7) Core cell placement *[4.8]*
- 5.8) Post-placement timing analysis *[4.9]*
- 5.9) Clock tree synthesis (optional) *[4.10]*
- 5.10) Design routing *[4.11]*
- 5.11) Post-route timing optimization and analysis *[4.12]*
- 5.12) Filler cell placement *[4.13]*
- 5.13) Design checks *[4.14]*
- 5.14) Report generation *[4.15]*
- 5.15) Post-route timing data extraction *[4.16]*
- 5.16) Post-route netlist generation *[4.17]*
- 5.17) GDS2 file generation *[4.18]*
- 5.18) Design import in Virtuoso *[4.19]*

Step 6) Post-layout VHDL/Verilog simulation (tool: Modelsim) *[2.4]*
- 6.1) Compilation of the Verilog netlist and related testbench
- 6.2) Simulation of the post-synthesis or post-PaR gate-level netlist with PaR timing data

cf.

- **"OpenCores HDL modeling guidelines," OpenCores, 2009.**
- **"Top-down digital design flow," Alain Vachoux, Microelectronic Systems Lab STI-IMM-LSM, 2006**

Cell-Based Design Flow