EECS 244: Overview of the IC Design Flow

Prof. Kurt Keutzer

EECS

keutzer@eecs.berkeley.edu

Class News

- **No class Monday – Labor Day**

The Purpose of the Class Project

- **Your first couple years of graduate school are about making the transition from**
  - **Capable textbook reader → scholar of the published research literature**
  - **Solitary student → Active team member**
  - **Assimilating well defined information → pursuing open questions**
  - **Excellent course/test taker → creative researcher**
- **The project portion of the course is to help you make this transition**

Project Outline

- Motivation
- Problem statement
- **Prior work**
- Investigative approach
- Results
- Summary
- Conclusions
- Future Work

How Conducted

- Research will be conducted in groups of 2-3
- Individual projects highly discouraged
- Research may be coordinated with other class projects: EECS249, EECS290N etc.
- Research will culminate in a:
  - Powerpoint presentation/demo – with explanatory notes
  - Written report

Tips for a Great Project

- Use your skill set
- Circuits, devices, processing, software development, system-level applications
- Great idea:
- Topical – e.g. system level, deep submicron effects, power
- Tractable – can make an impact in a semester, have all the software, examples, data files that you need
- Get started early
- Get mentorship (senior grad students, post-docs, prof of course)
- Follow deadlines
- Formulate the problem clearly
- Formulate your results clearly

Some Successful 244 Projects

- ``Getting to the Bottom of Deep Submicron'', D. Sylvester, K. Keutzer, In *Proceedings of the International Conference on Computer-Aided Design, November, 1998, pp. 203-211.*
- ``Towards True Crosstalk Noise Analysis'', P. Chen, K. Keutzer, In *Proceedings of the International Conference on Computer-Aided Design, November, 1999, pp. 132-137.*
- ``Impact of Systematic Spatial Intra-Chip Gate Length Variability on Performance of High-speed Digital Circuits'' M. Orshansky, L. Milor, P. Chen, K. Keutzer, C. Hu, In *Proceedings of the International Conference on Computer-Aided Design, November, 2000, pp. 62-67.*
- `` Bus Encoding to Prevent Crosstalk Delay '', B. Victor, K. Keutzer, *Proceedings of the International Conference on Computer-Aided Design, November, 2001.*
- "Constraint Driven Communication Synthesis", A. Pinto, L. Carloni, A. Sangiovanni-Vincentelli, DAC 2002
- "Multi-Domain Clock Skew Scheduling", Kaushik Ravindran, Andreas Kuehlmann, Ellen Sentovich, ICCAD 2003.

Important Class Dates

- **Project teams + topics (1 paragraph) due 9/13**
- **Full project proposals due 9/29**
- **Exam 1 Handed out 10/6**
- **Exam 1 collected at BEGINNING of class 10/11**
- **Preliminary project report due 11/1**
- **Exam 2 Handed out 11/3**
- **Exam 2 collected at BEGINNING of class 11/8**

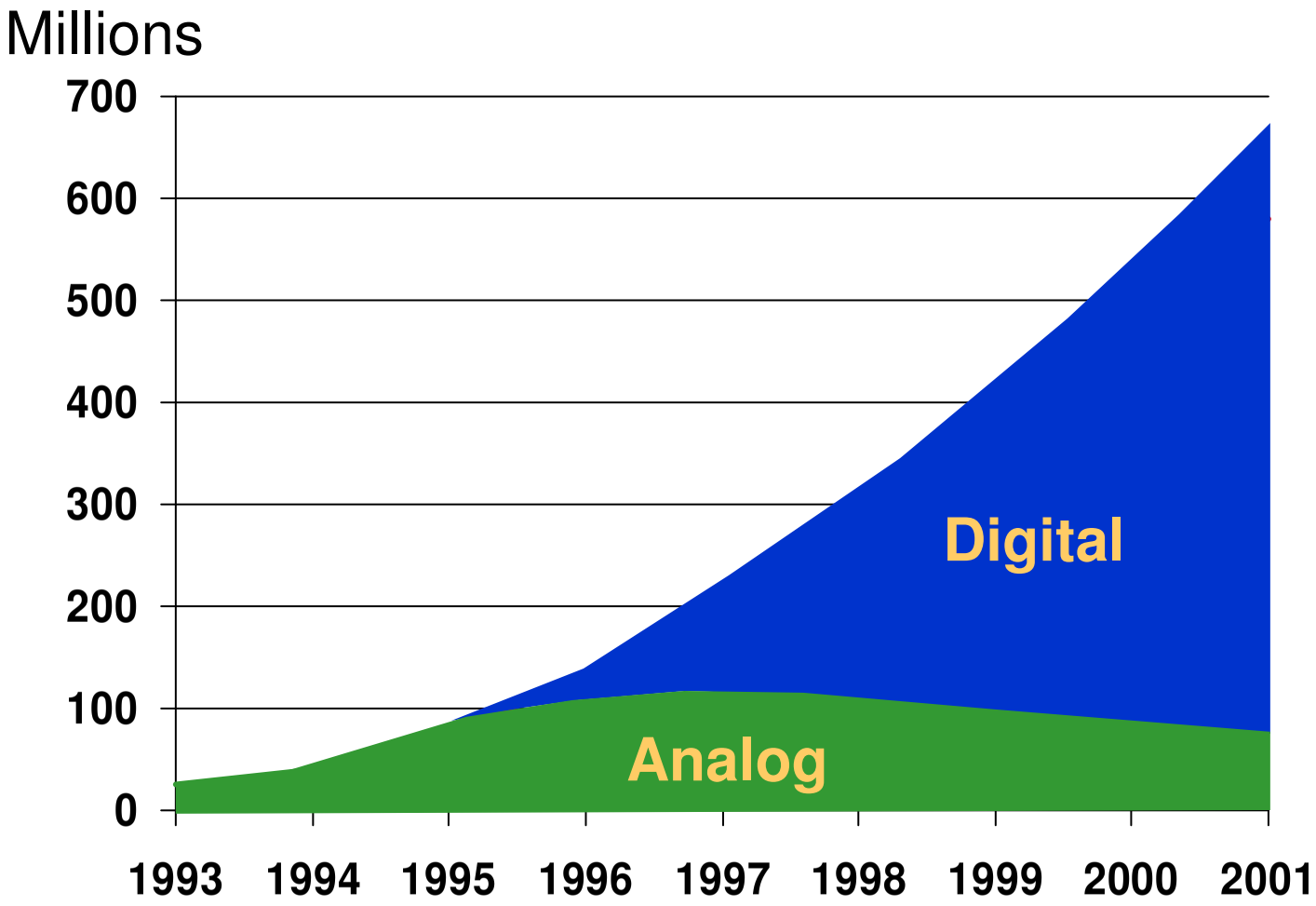
- **Final project presentations – between 12/6 and 12/8**

Application

- **Motivated by:**
  - **A bright idea**
  - **A market opportunity**
    - **An emerging market**
    - **A high growth market**
  - **A technological breakthrough**

For example - wireless telephony

Market Opportunity - World's Cellular Subscribers



Will provide a ubiquitous infrastructure for wireless data as well as voice

Source: Ericsson Radio Systems, Inc.

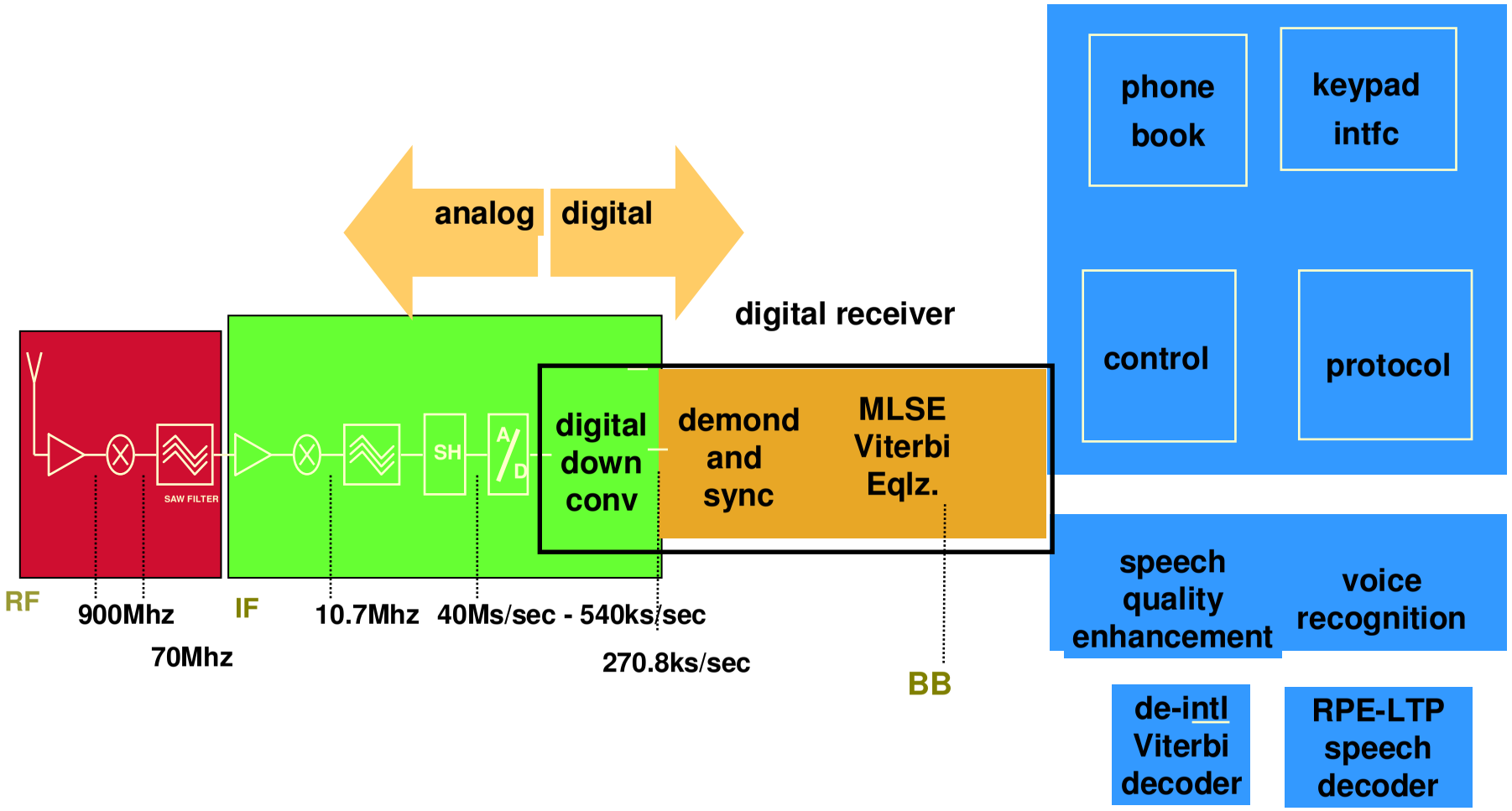
Specification

- Function, performance (power, delay, area), cost
  - A competitor's
    - Integrated circuit
    - Data sheet
  - A napkin
  - An industry standard

For example, GSM standard for cellular telephony

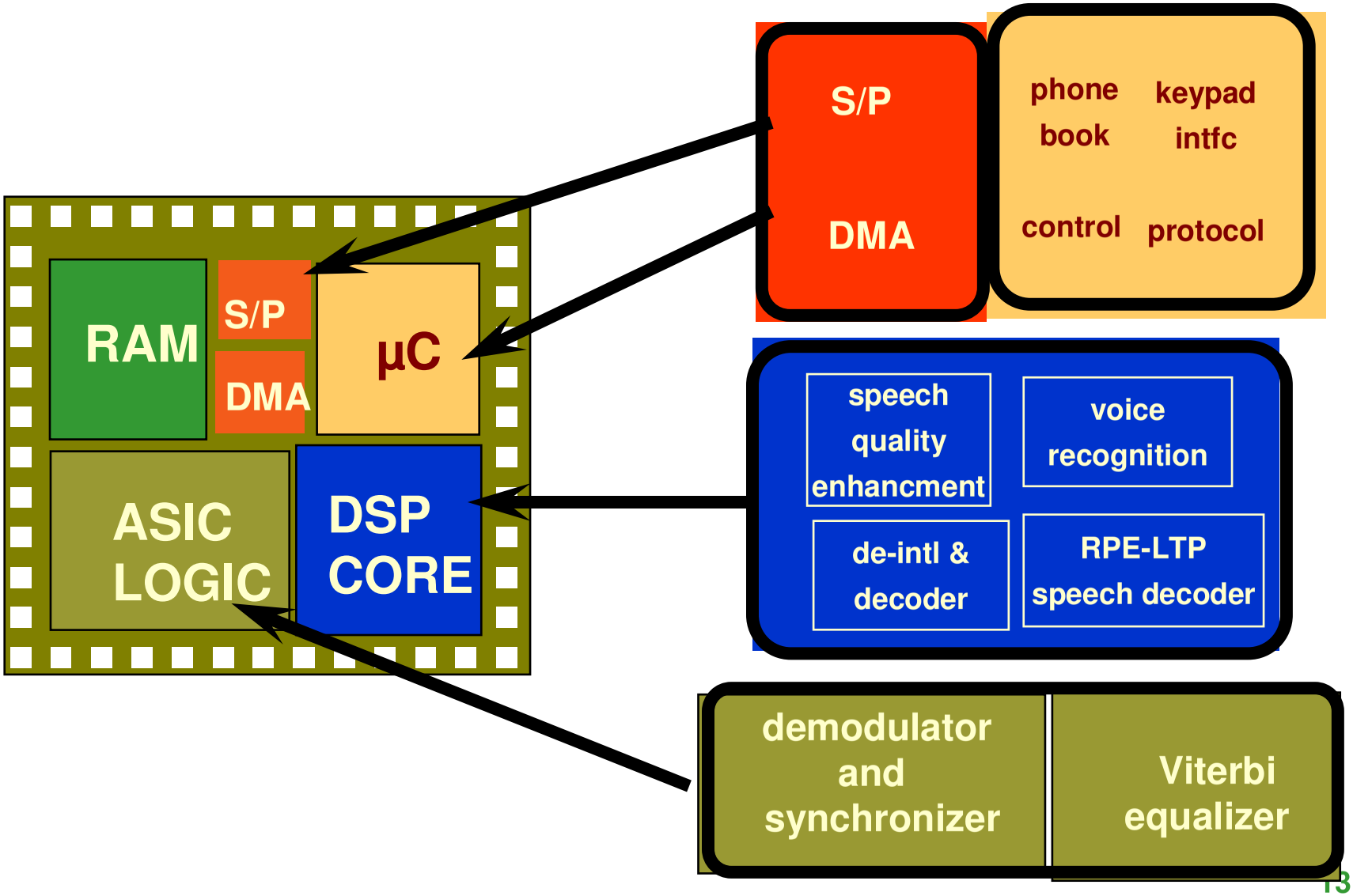
System Level Model: GSM System

- System model is organized into major components

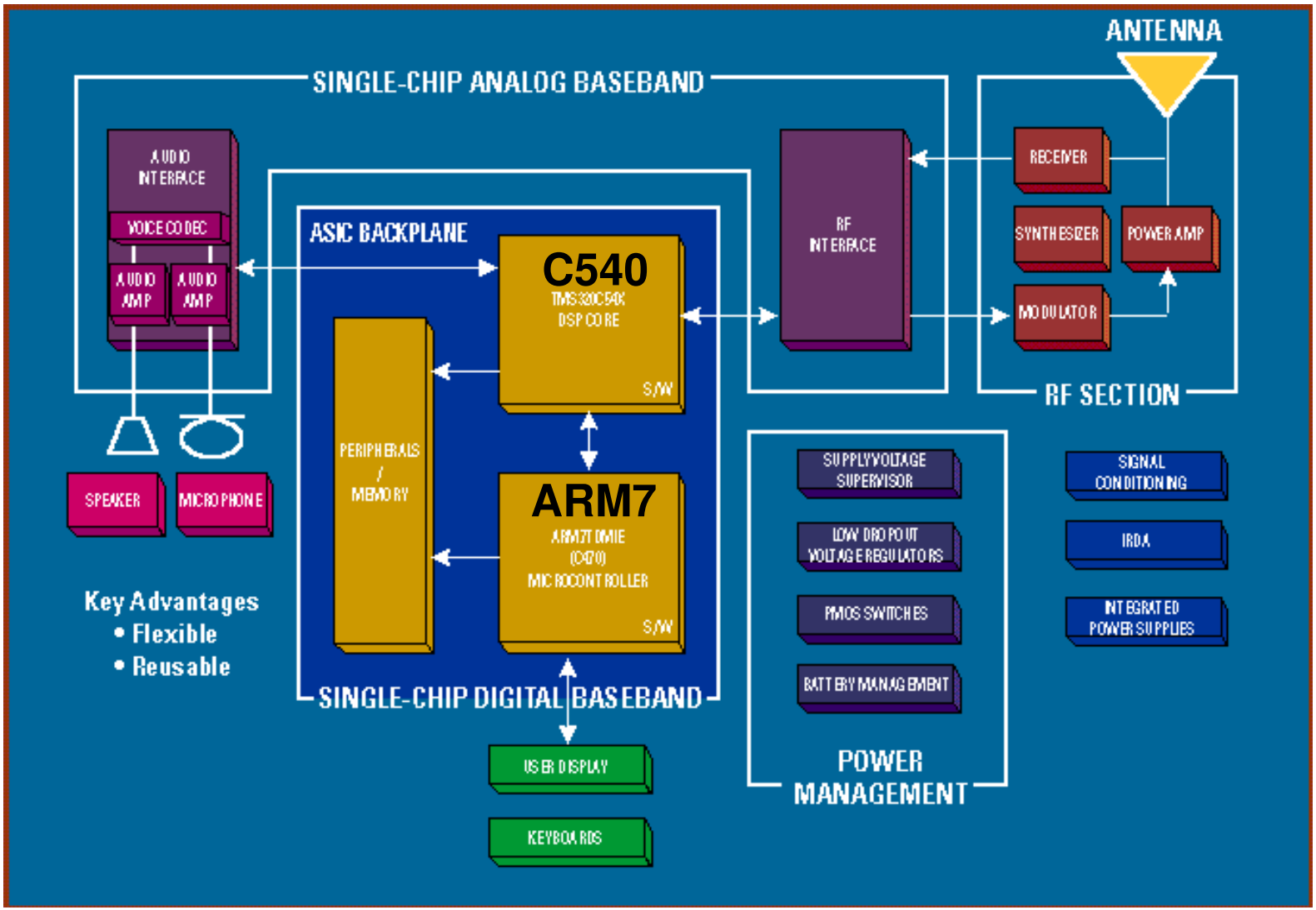


Courtesy Ravi Subramaniam

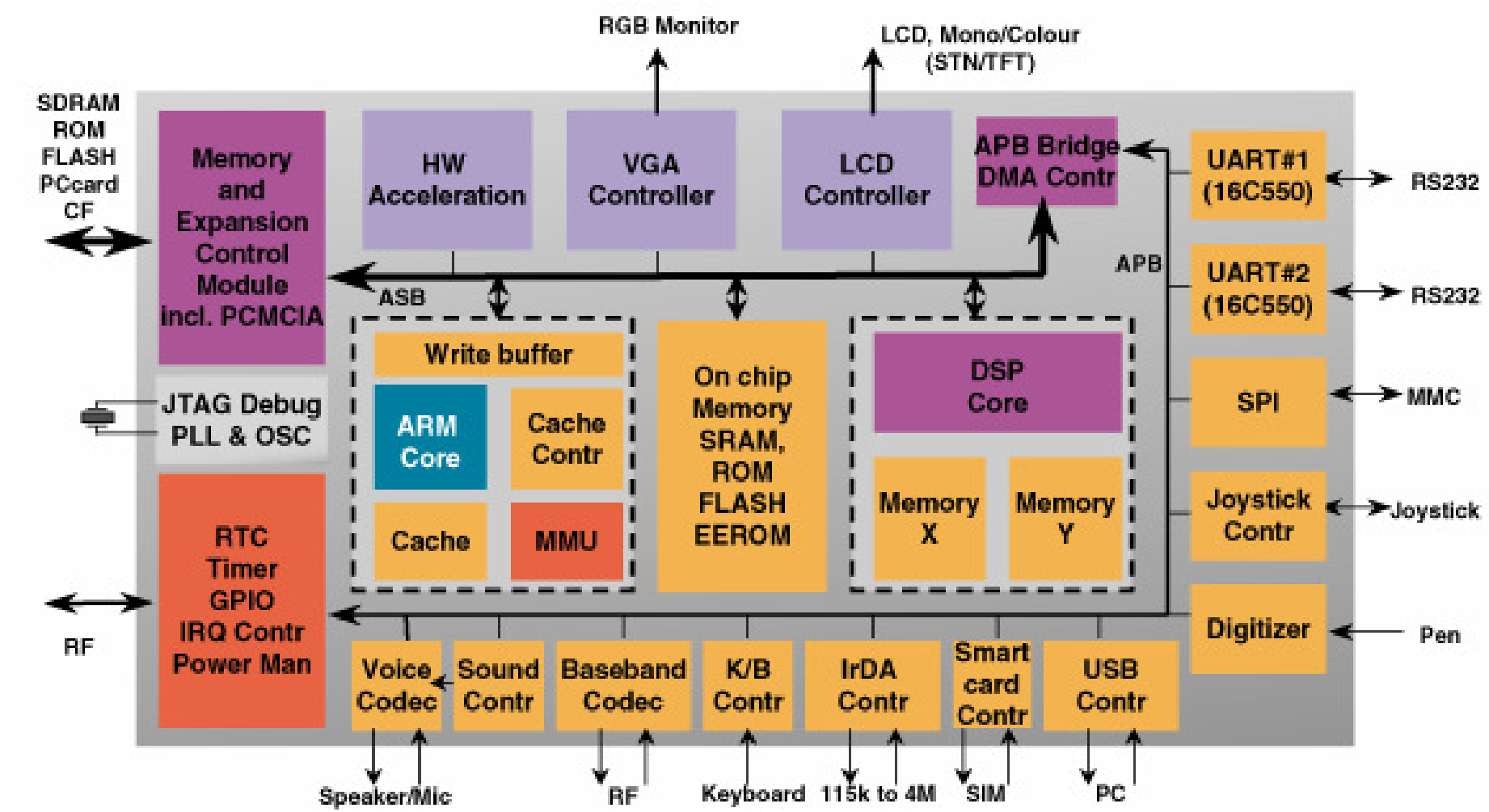
Mapping onto a system on a chip



Full Wireless Phone Organization



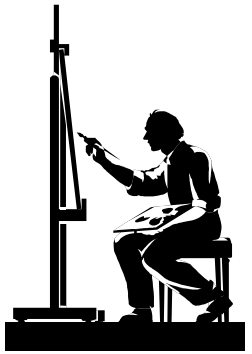
A more complex design – SOC architecture



Courtesy Synopsys

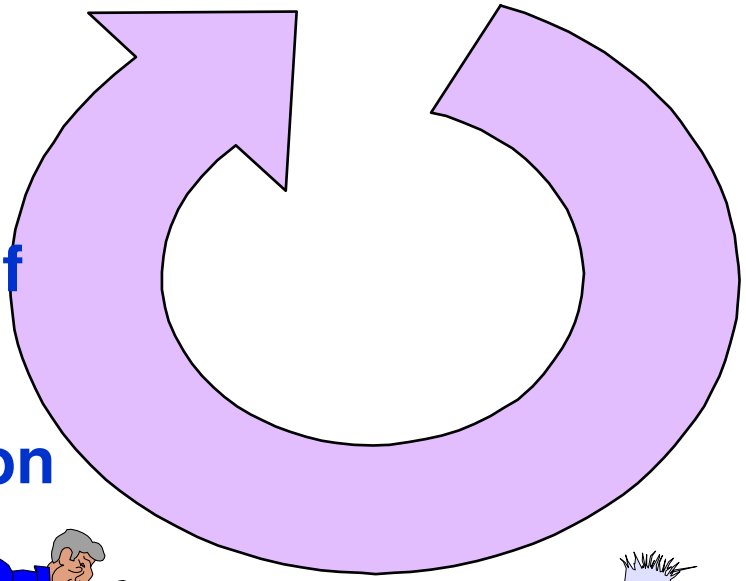
From specification to design entry

- **Design** : specify and enter the design intent



Verify:

verify the correctness of design and implementation

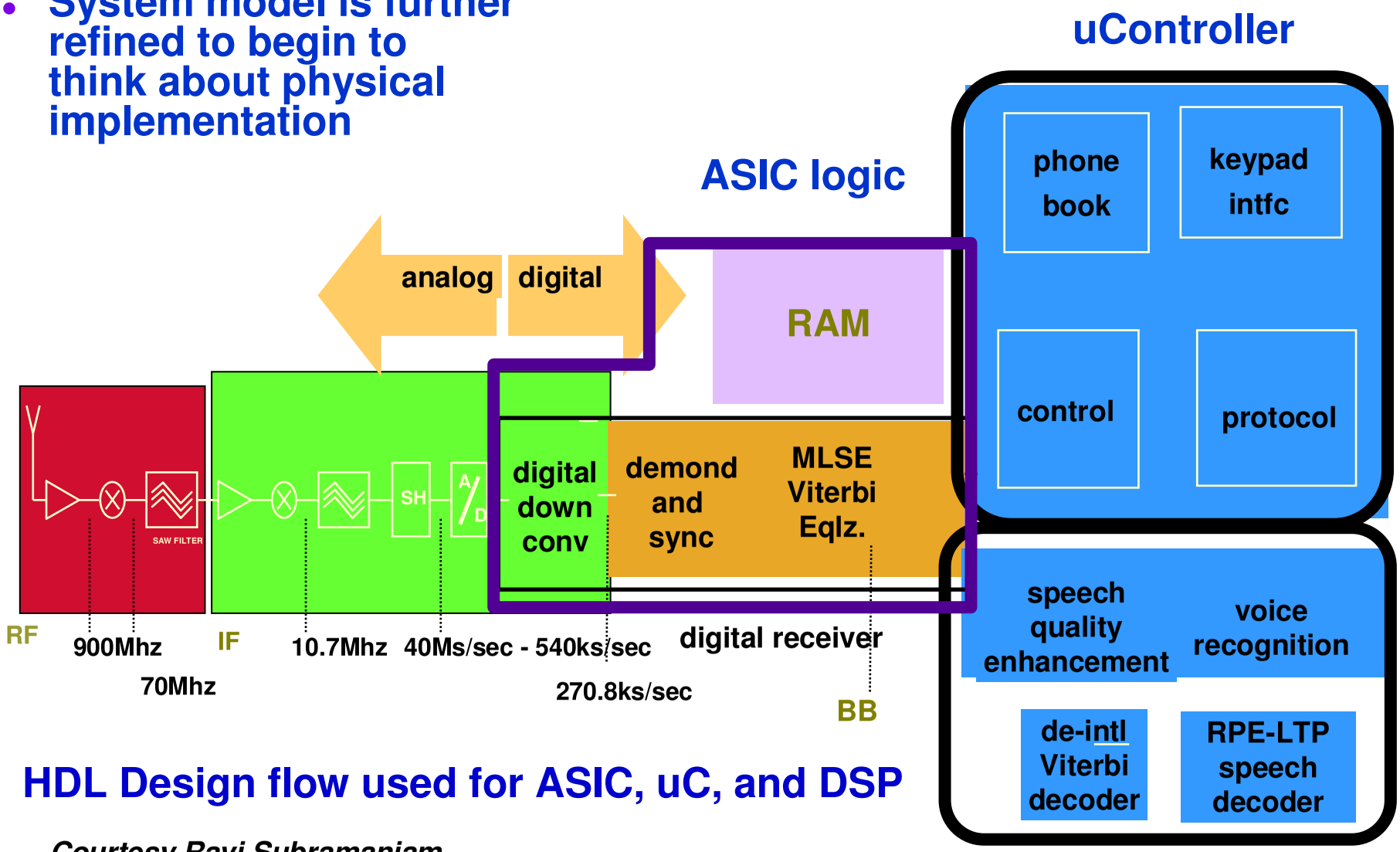


Implement:

refine the design through all phases

Targeting an IC Implementation

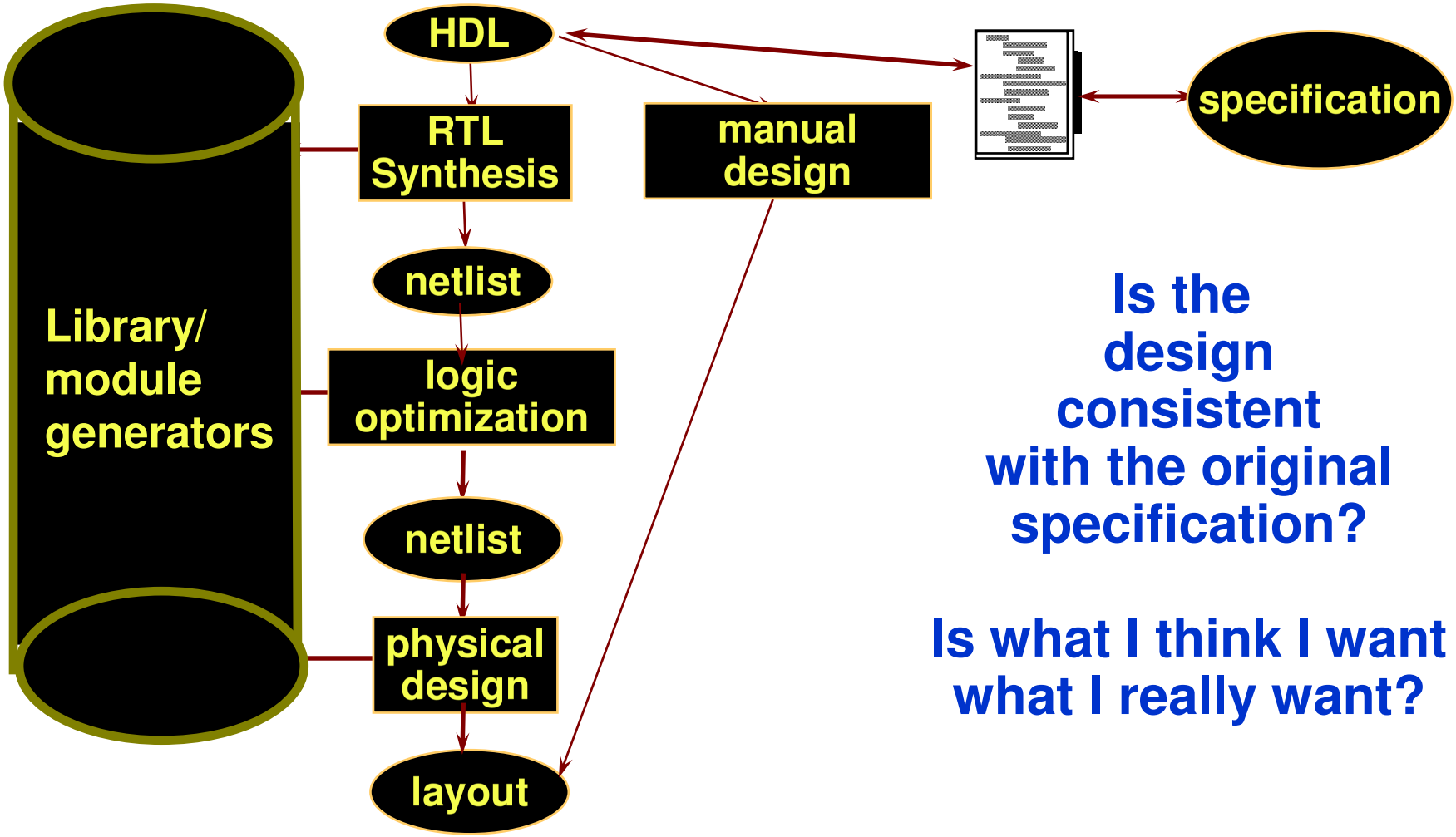
- System model is further refined to begin to think about physical implementation



HDL Design flow used for ASIC, uC, and DSP

Courtesy Ravi Subramaniam

Design Flow



Is the design consistent with the original specification?

Is what I think I want what I really want?

Current Practice: HDL at RTL Level

```
module foobar (q,clk,s,a,b);
   input clk, s, a, b;
   output q; reg q; reg d;
always @(a or b or s) // mux
      begin
        if( !s )
          d = a;
        else if( s )
          d = b;
        else
          d = 'bx;
      end // always @ (a or b or s)

always @(clk) // latch
   begin
     if( clk == 1 )
       q = d;
     else if( clk !== 0 )
       q = 'bx;
   end // always @ (clk)

endmodule
```

RTL level

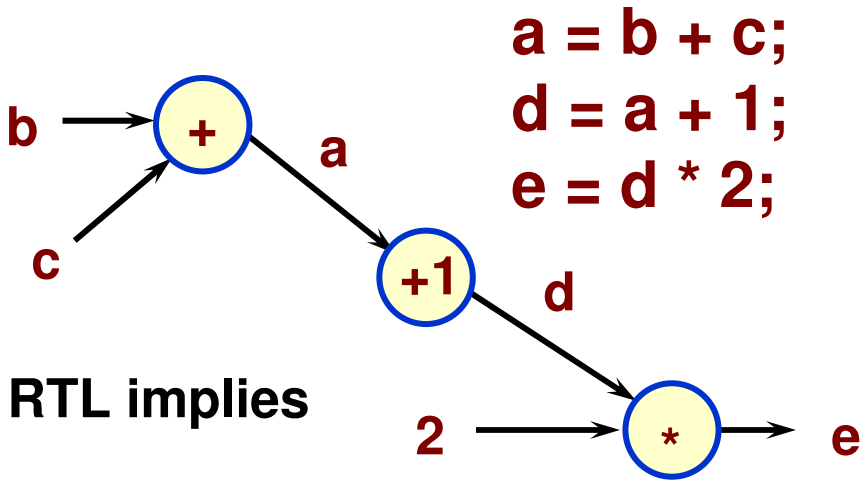
An RTL description is always implicitly structural - the registers and their interconnectivity are defined

Thus the clock-to-clock behavior is defined

Only the control logic for the transfers is synthesized.

This approach can be enhanced:

- Register inferencing
- Automating resource allocation

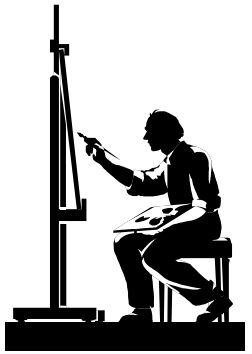


Behavioral implies

e = 2 * (b + c + 1)

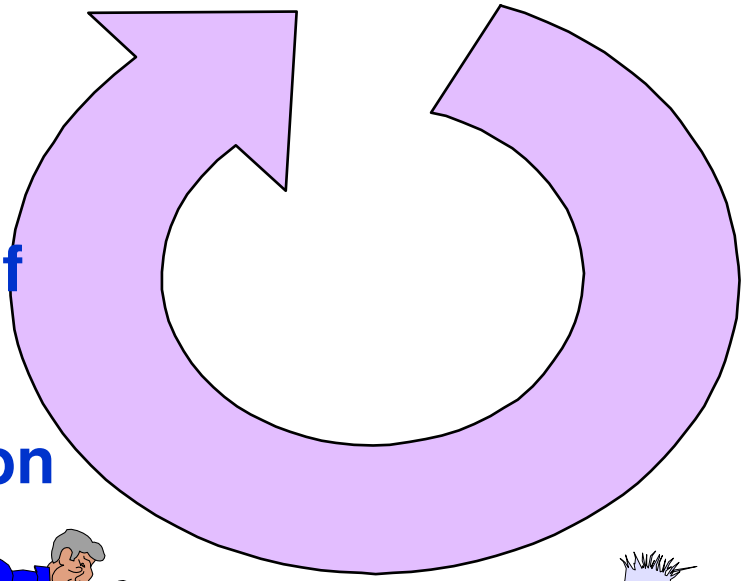
Verification

- **Design** : specify and enter the design intent



Verify:

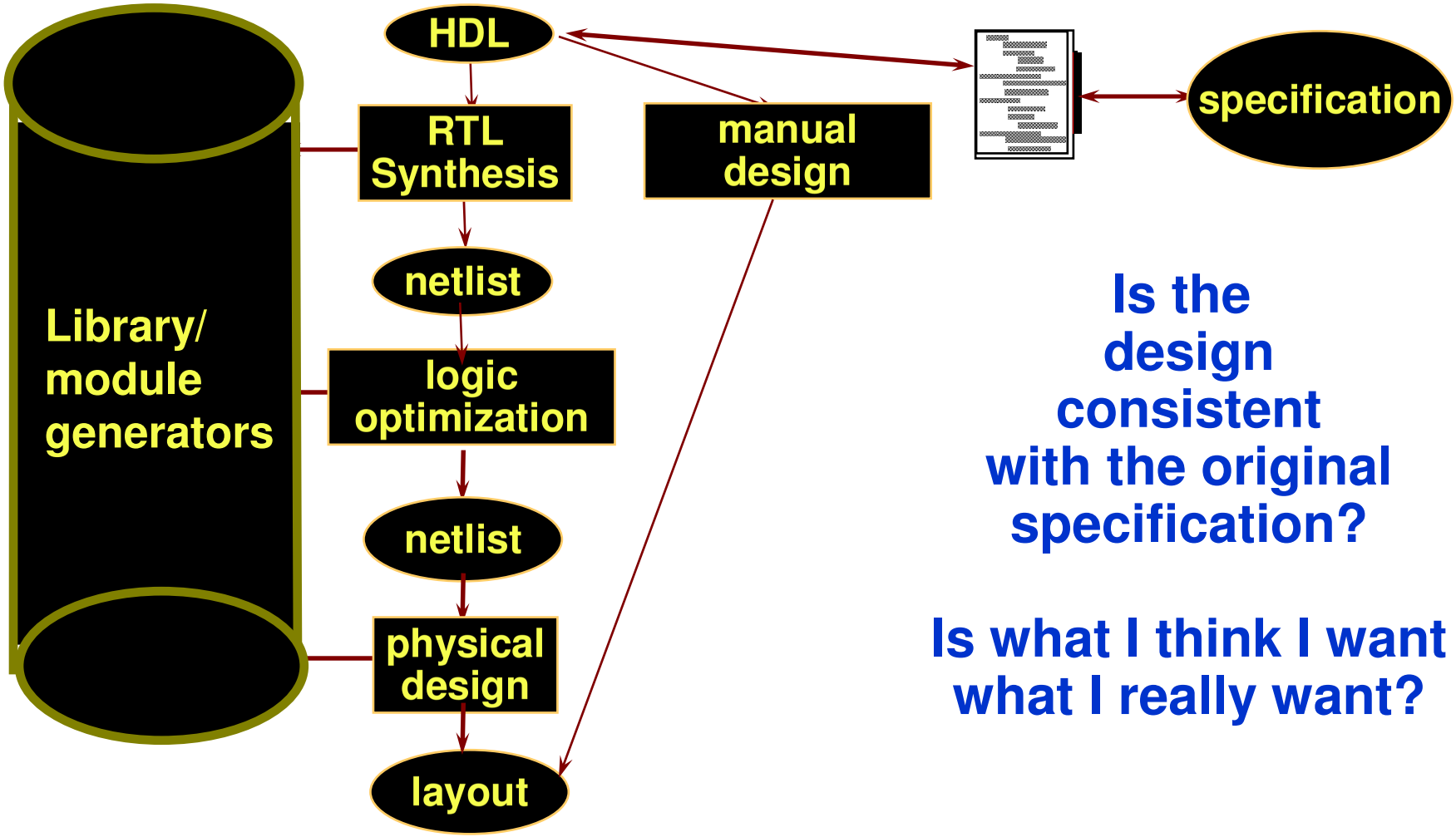
verify the correctness of design and implementation



Implement:

refine the design through all phases

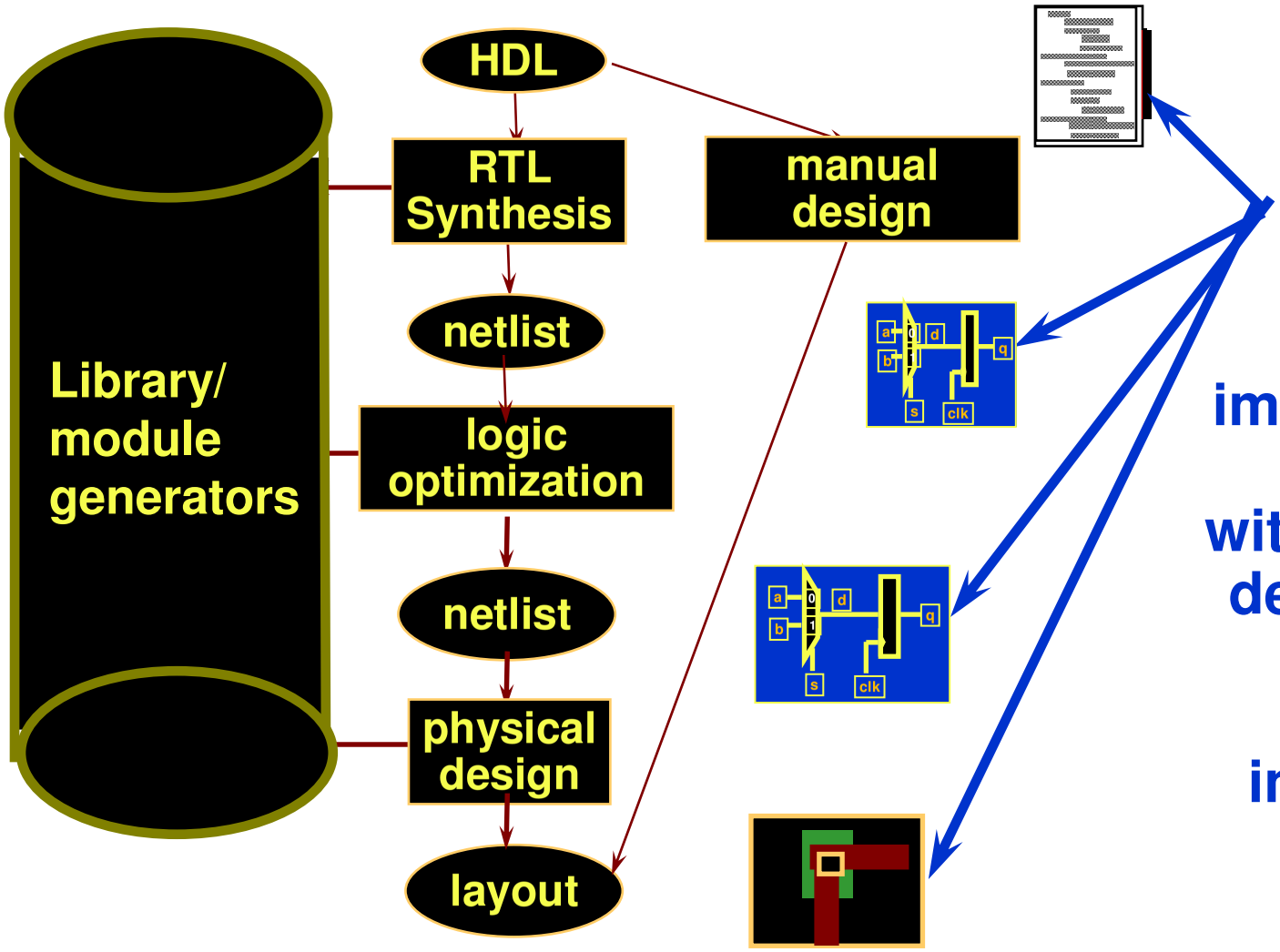
Design Verification



Is the design consistent with the original specification?

Is what I think I want what I really want?

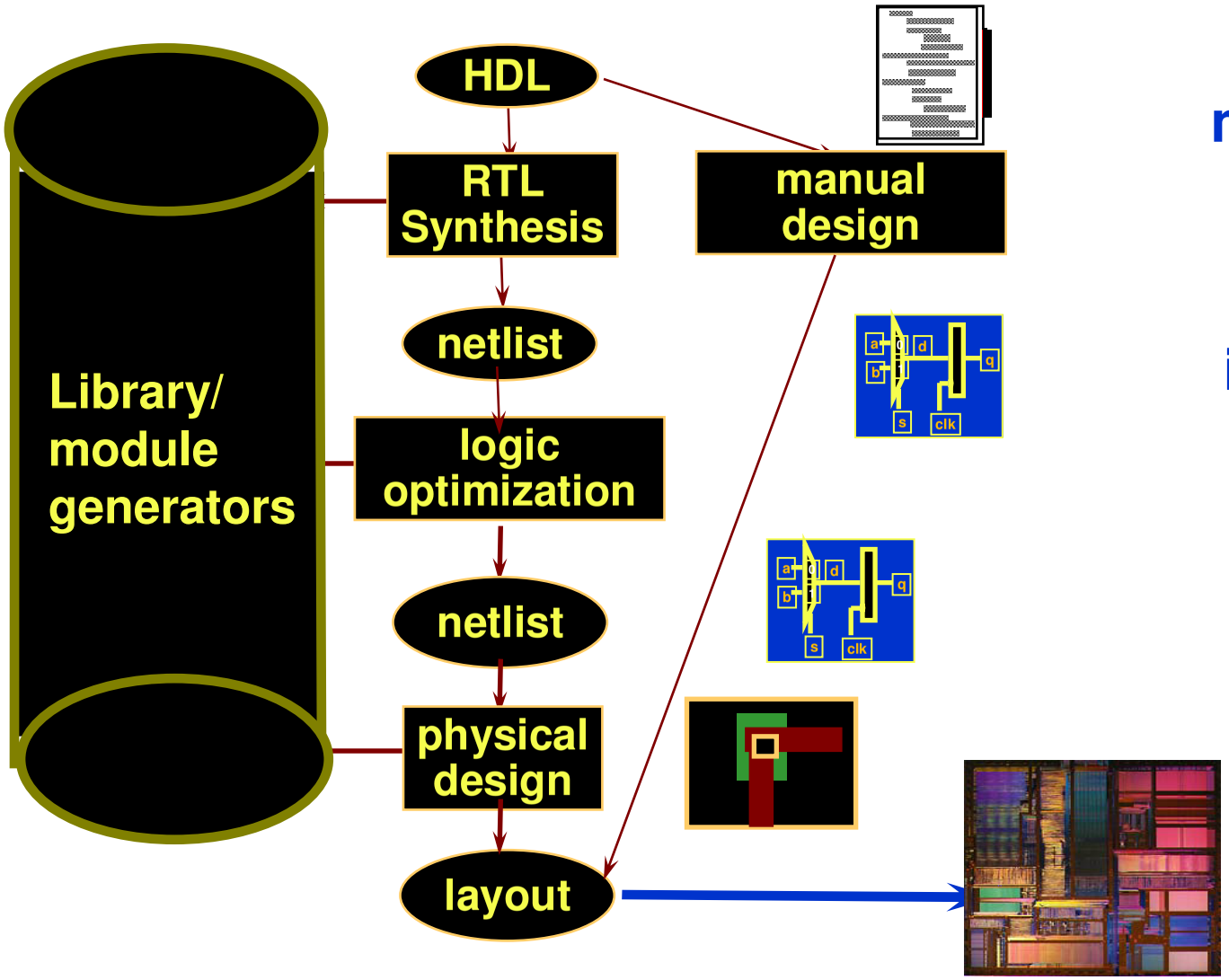
Implementation Verification



Is the implementation consistent with the original design intent?

Is what I implemented what I wanted?

Manufacture Verification (Test)



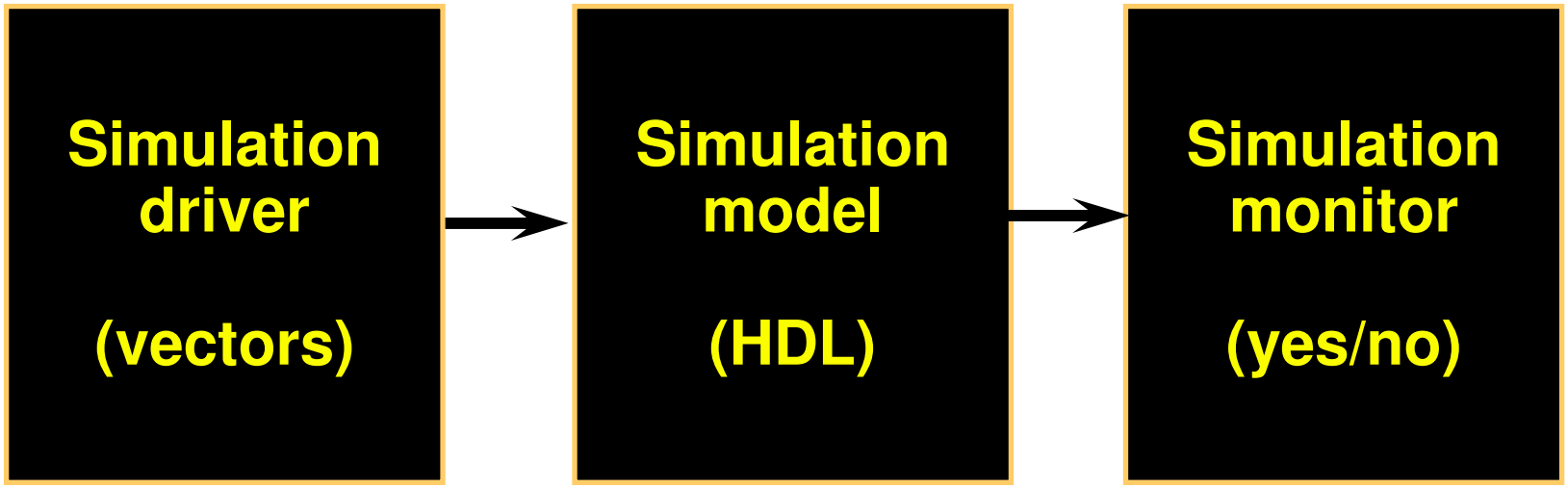
Is the manufactured circuit consistent with the implemented design?

Did they build what I wanted?

Approaches to Design Verification

- Formal verification
  - Model checking - prove properties relative to model
  - Theorem proving - prove properties of a circuit
- Simulation
  - Application of simulation stimulus to model of circuit
- Emulation
  - Implement a version of the circuit on emulator
- Rapid prototyping
  - Create a prototype of actual hardware

Software Simulation

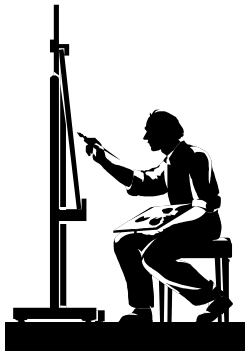


Types of software simulators

- Circuit simulation
  - Spice, Advice, Hspice
  - Timemill + Ace, ADM
- Event-driven gate/RTL/Behavioral simulation
  - Verilog - VCS, NC-Verilog, Turbo-Verilog, Verilog-XL
  - VHDL - VSS, MTI, Leapfrog
- Cycle-based gate/RTL/Behavioral simulation
  - Verilog - Speedsim
  - VHDL – Cyclone
- Generic system-level simulation - SystemC
- Domain-specific simulation
  - SPW, COSSAP,
- Architecture-specific simulation
  - VAST, Axys, Lisatek

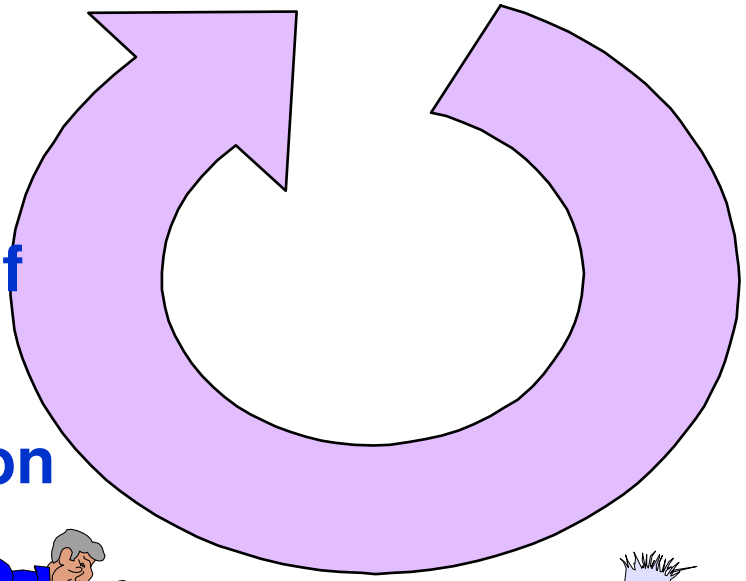
Implementation

- **Design** : specify and enter the design intent



Verify:

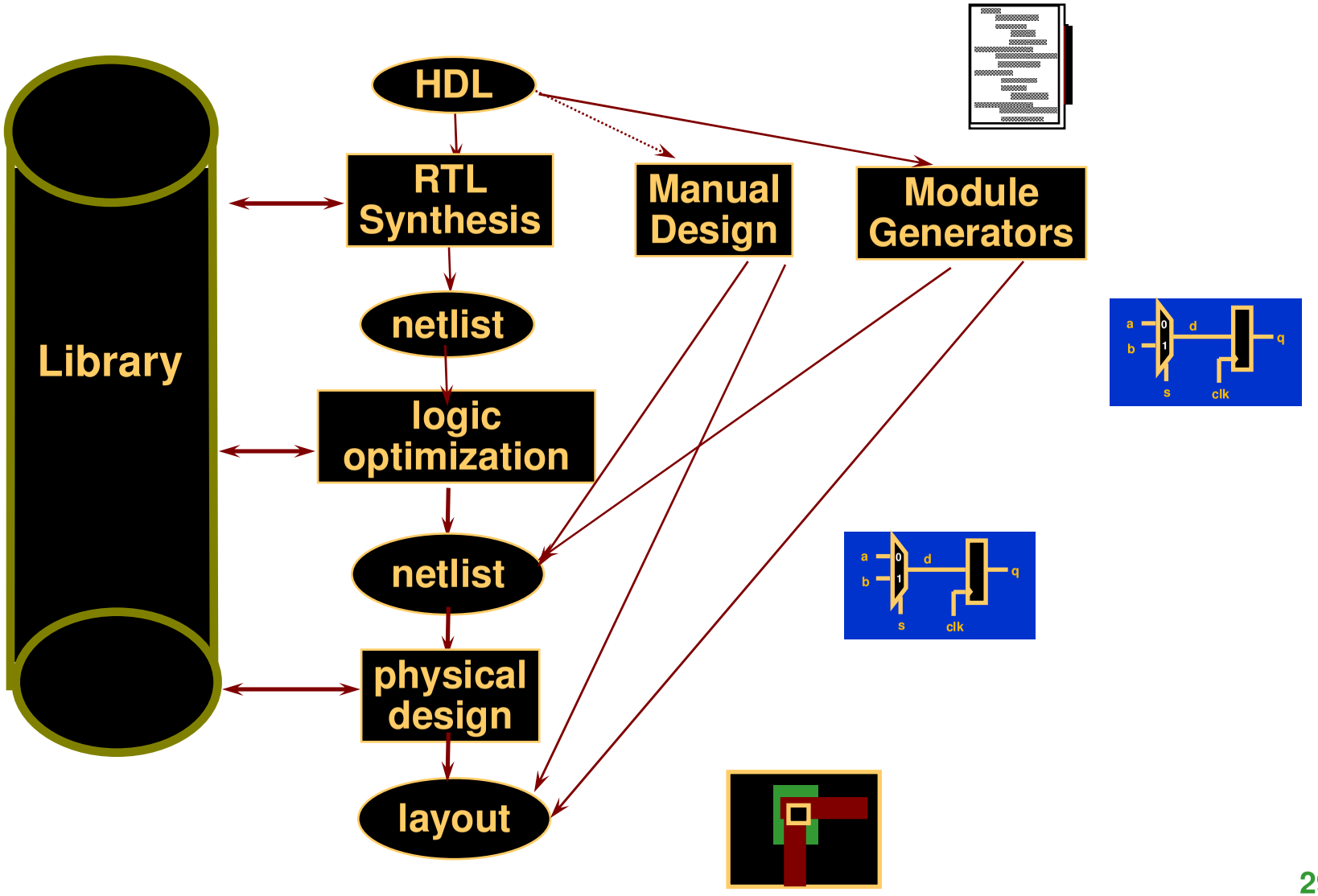
verify the correctness of design and implementation



Implement:

refine the design through all phases

RTL Design Flow



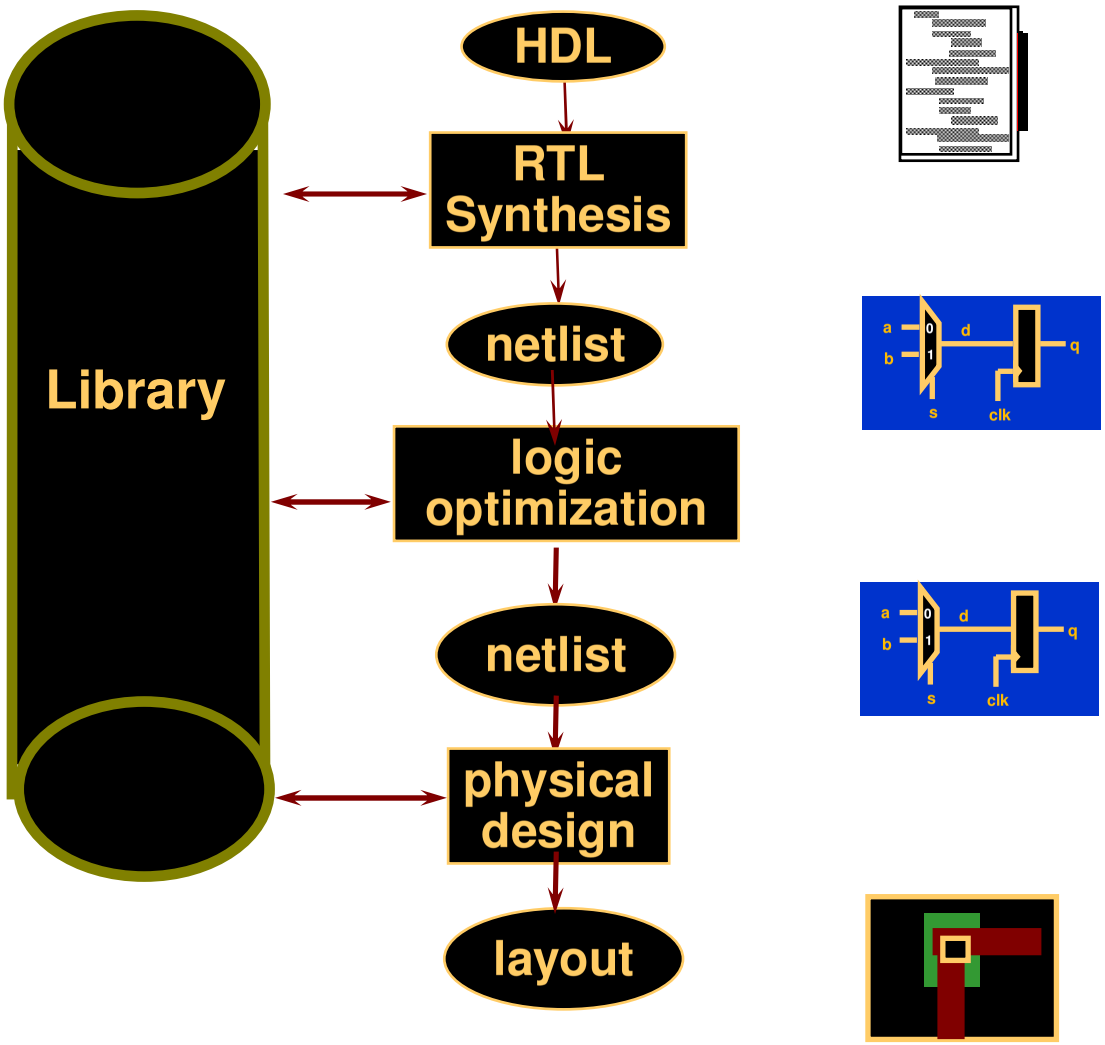
Manual Design

- Performed at
  - Gate level (100 gates/week) /gate-level editor
  - Transistor level (10 - 20 gates week)/tr level editor
- Very expensive in design cost and design time -
- Used for:
  - Analog
  - Leaf cells - libraries, memory cells
  - Datapaths in high performance designs - DSP, microprocessor etc.

Module Generators

- Parameterized generators of actual physical layout
- Typically used for:
  - Memories (word length, #words, # ports)
  - Programmable logic arrays (PLA)
  - Register files
- Occasionally used for:
  - Multipliers
  - General-purpose datapath
  - Datapaths in high performance designs - DSP, microprocessors etc.

Workhorse: RTL Synthesis Flow



Library

- Contains for each cell:
  - Functional information: cell = a *b * c
  - Timing information: function of
    - input slew
    - intrinsic delay
    - output capacitance

    non-linear models used in tabular approach
  - Physical footprint (area)
  - Power characteristics
- Wire-load models - function of
  - Block size
  - Wiring

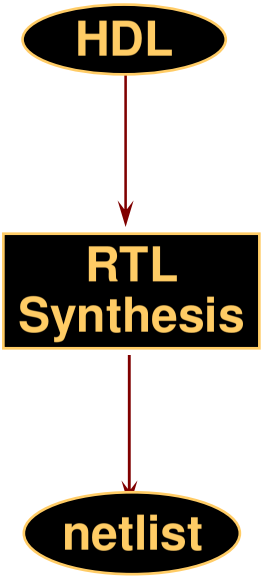
Reasonable Library Functions

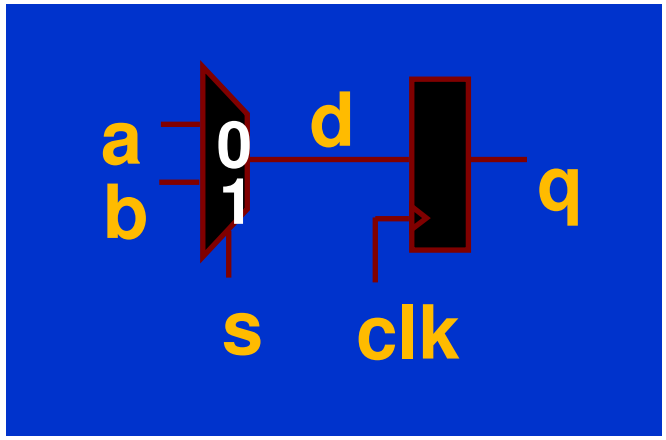
- Inverter, Buffer
- ND2-ND4; NOR2-NOR4; AND2- AND4;
- AOI21 - AOI333; OAI21 - OAI333
- XOR, XNOR
- MUX, Full Adder
- Neg-Edge Triggered D-Flip-Flop
- Pos-Edge Triggered D-FF
- J-K FF
- Above with various clears, enables
- Scan versions of each of the above
- Most of the above in 6 different power sizes:
  - 1x, 2x, 4x, 6x, 8x, 16x

RTL Synthesis

```
module foobar (q,clk,s,a,b);
  input clk, s, a, b;
  output q; reg q; reg d;

  always @(a or b or s) // mux
    begin
      if(~s )
        d = a;
      else if( s )
        d = b;
      else
        d = 'bx;
    end // always @ (a or b or s)
```

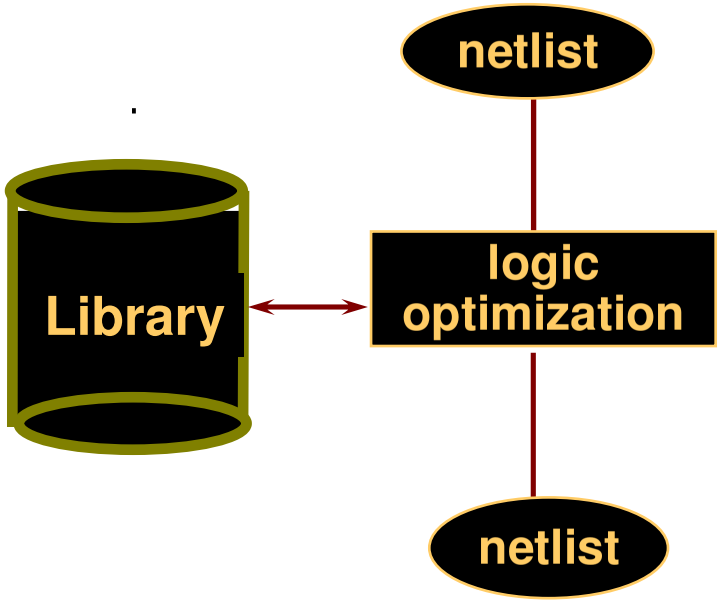




translate HDL source into netlist

Logic Optimization

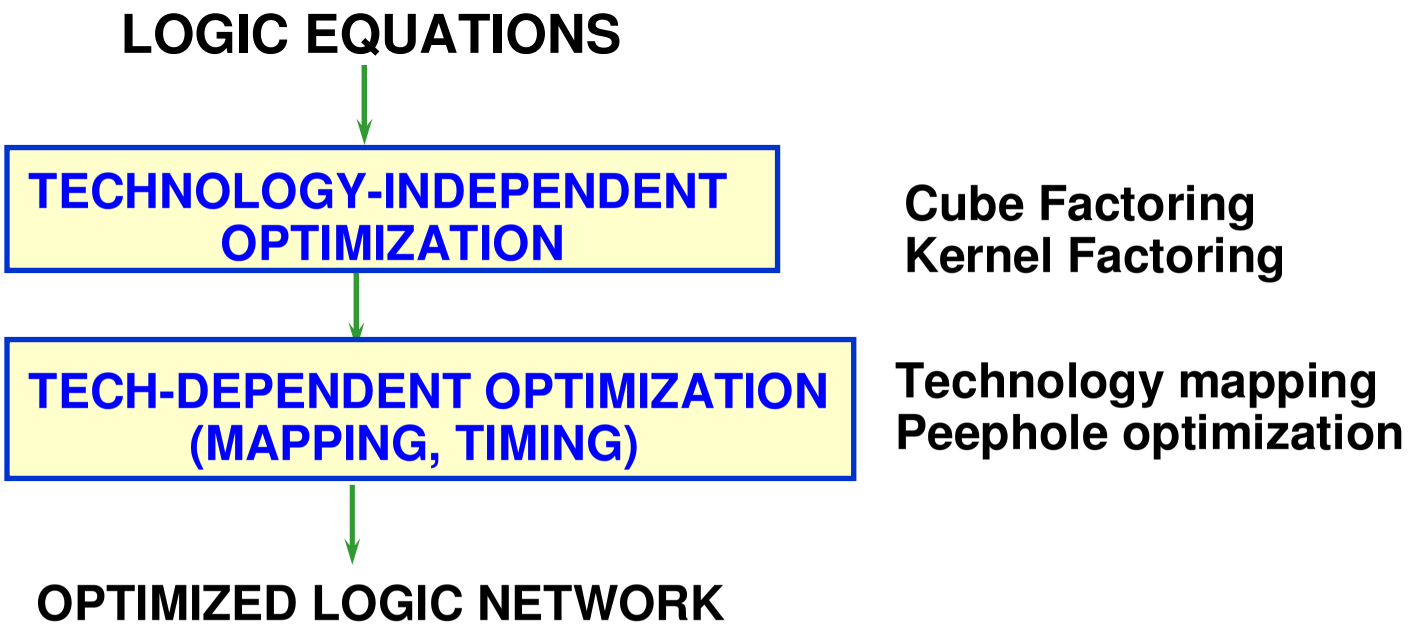
- Perform a variety of transformations and optimizations
  - Structural graph transformations
  - Boolean transformations
  - Mapping into a physical library



pre-optimized

smaller, faster
less power

Optimization Technologies



Kernel and Boolean Factorization

$$f = a\bar{b} + a\bar{c} + b\bar{a} + b\bar{c} + c\bar{a} + c\bar{b}$$

- **Algebraic factorization procedures**

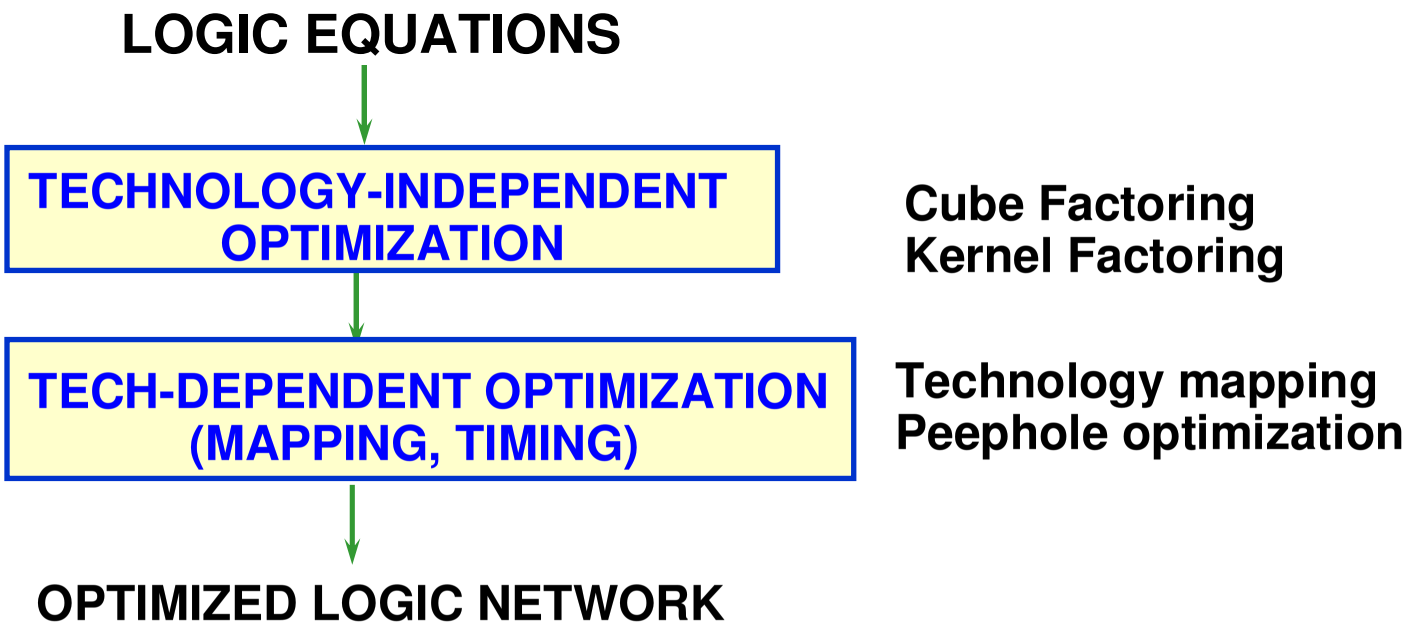
$$f = a(\bar{b} + \bar{c}) + \bar{a}(b + c) + b\bar{c} + c\bar{b}$$

- **Boolean factorization produces**

$$f = (a + b + c)(\bar{a} + \bar{b} + \bar{c})$$

"The decomposition and factorization of Boolean expressions"
RK Brayton, C McMullen, Proc. ISCAS, 1982
Devadas and Keutzer - Chapter 6&7

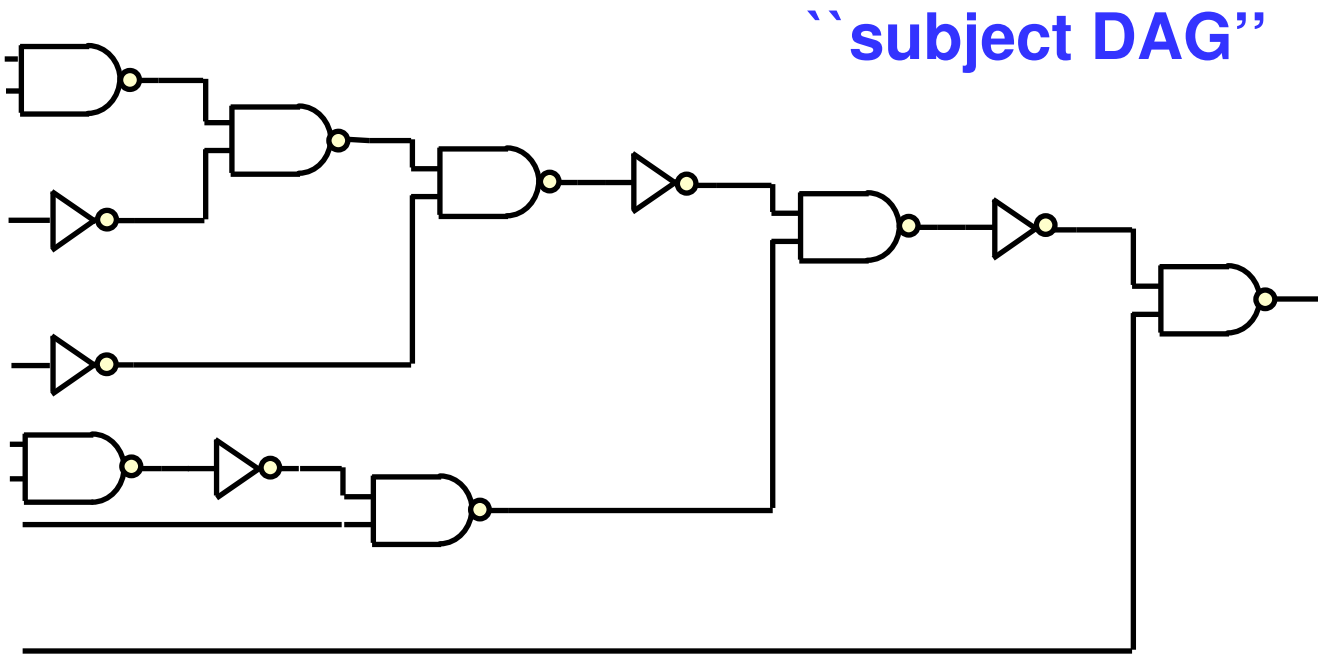
Optimization Technologies



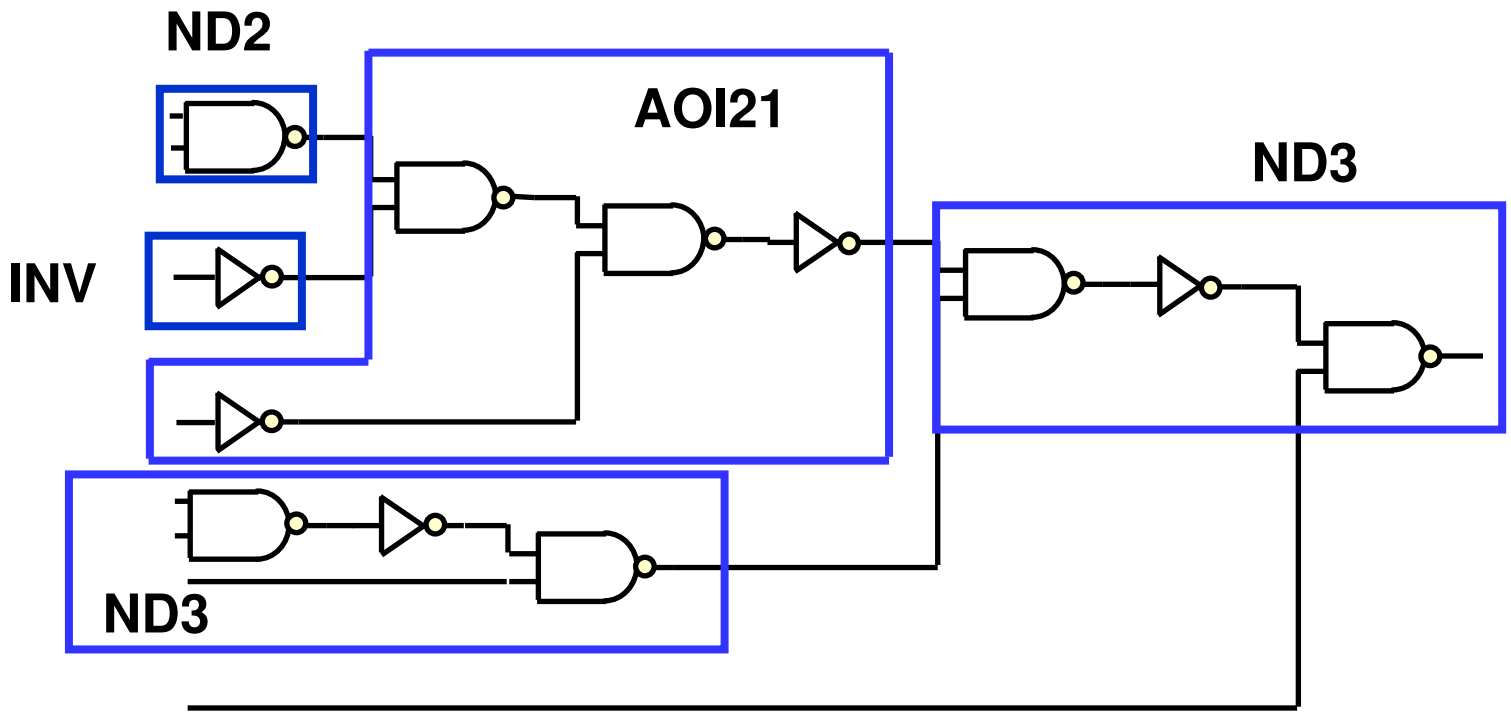
Reasonable Library

- Inverter, Buffer
- ND2-ND4; NOR2-NOR4; AND2- AND4;
- AOI21 - AOI333; OAI21 - OAI333
- XOR, XNOR
- MUX, Full Adder
- Neg-Edge Triggered D-Flip-Flop
- Pos-Edge Triggered D-FF
- J-K FF
- Above with various clears, enables
- Scan versions of each of the above
- Most of the above in 6 different power sizes:
  - 1x, 2x, 4x, 6x, 8x, 16x

Input Circuit Netlist



Find a Mapping into the Tech Library

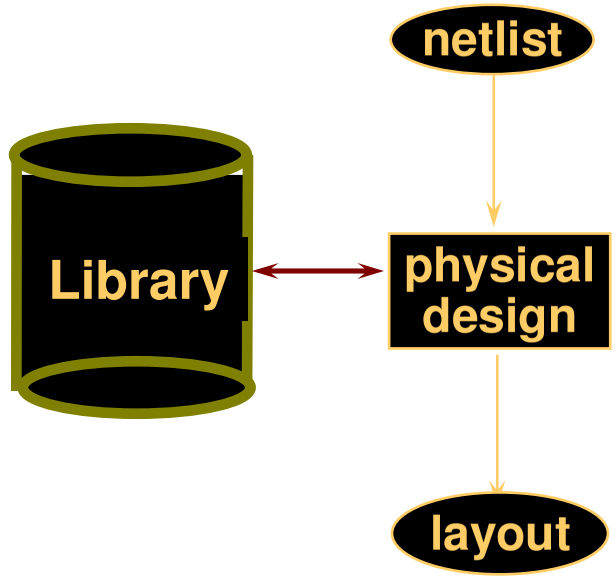


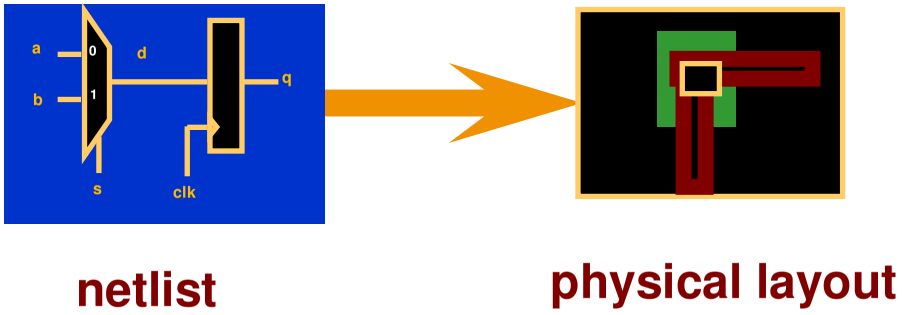
Result is a netlist in the technology library

DAGON: Technology Binding and Local Optimization by DAG Matching
K Keutzer, Proceedings of DAC, 1987, pages 341-347.

Physical Design

- Transform sequential circuit netlist into a physical circuit
  - *place* circuit components
  - *route* wires
  - transform into a mask
- Or for FPGA's
  - *place* look-up tables
  - *route* wires

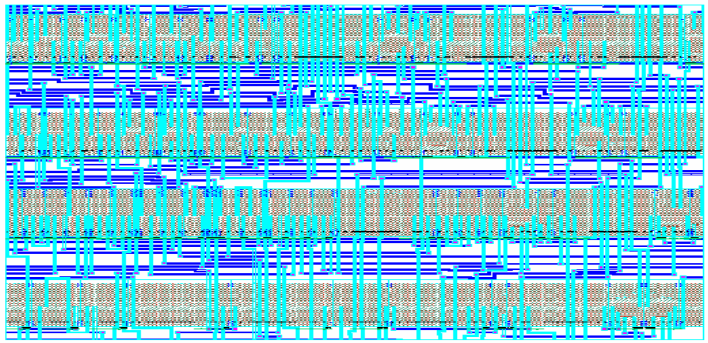




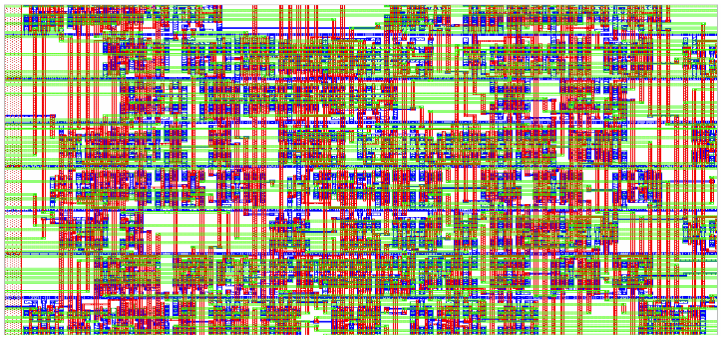
Channeled and Channel less

- Routing:
  - Channel based: Routing only in channels between gates (few metal layers: 2)
  - Channel less: Routing over gates (many metal layers: 3 - 6)
  - Often split in two steps:
    - Global route: Find a coarse route depending on local routing density
    - Detailed route: Generate routing layout

Channel based

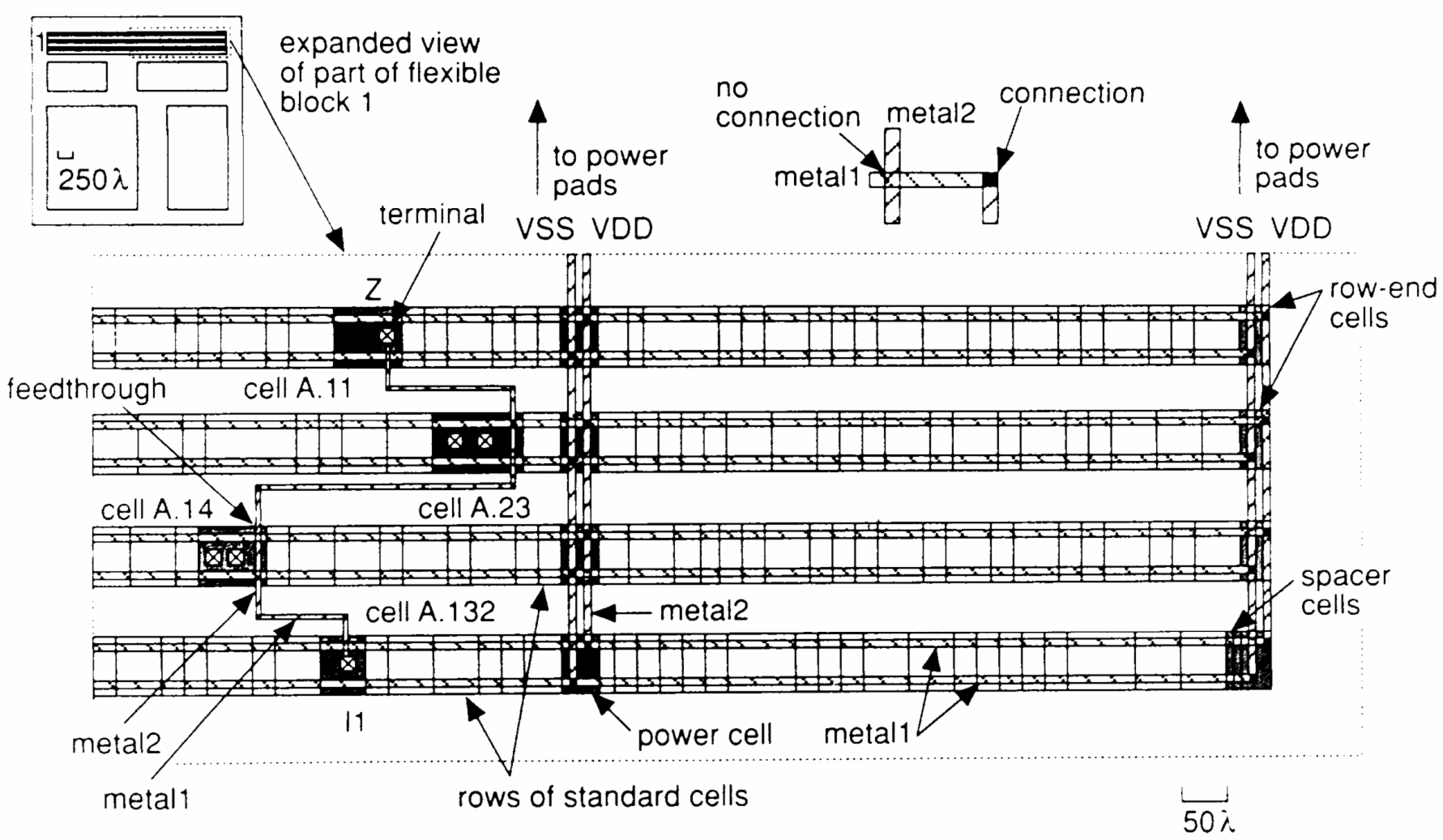


Channel less

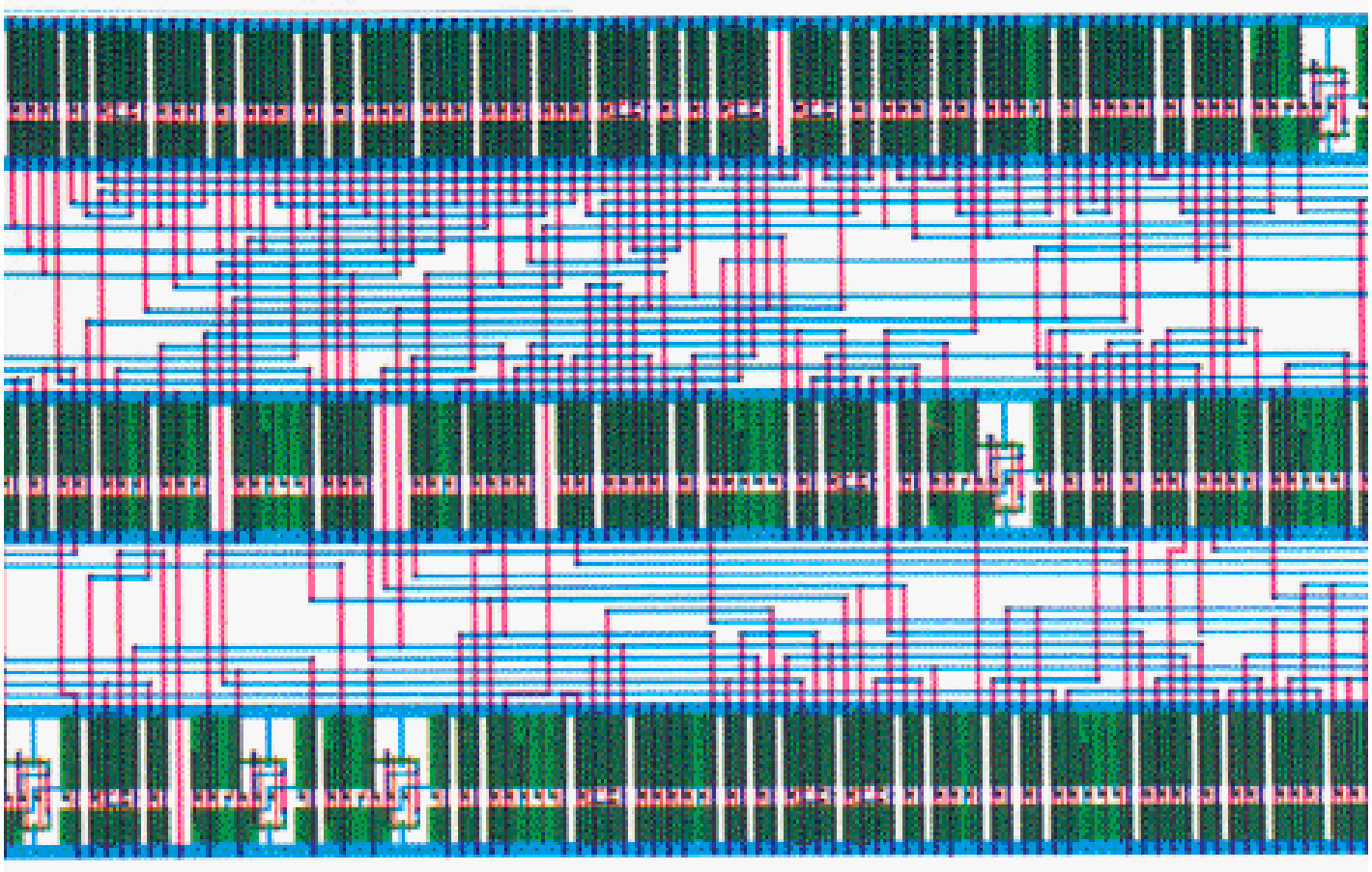


- J. Christiansen,
- CERN - EP/MIC
- Jorgen.Christiansen@cern.ch

Channeled Gate Array

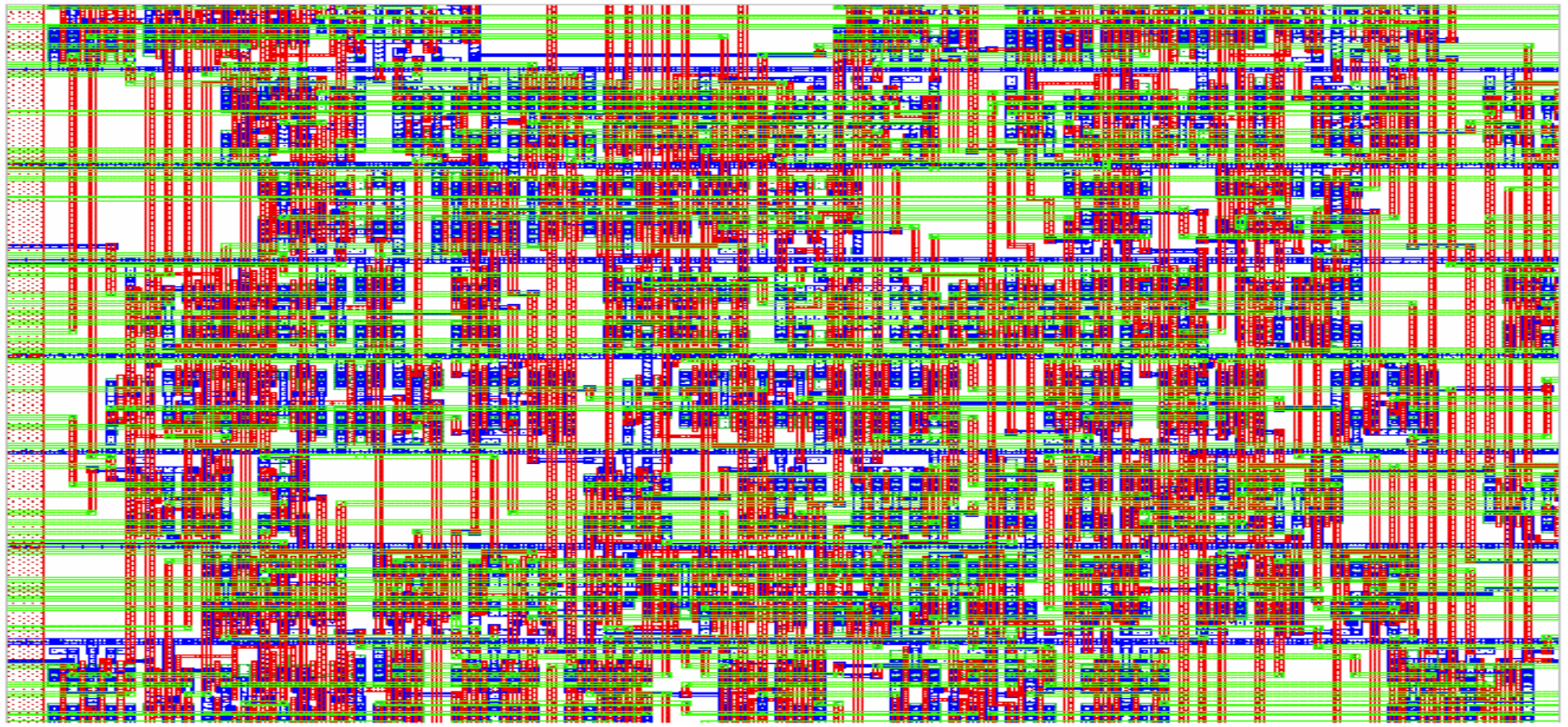


Standard Cell Layout

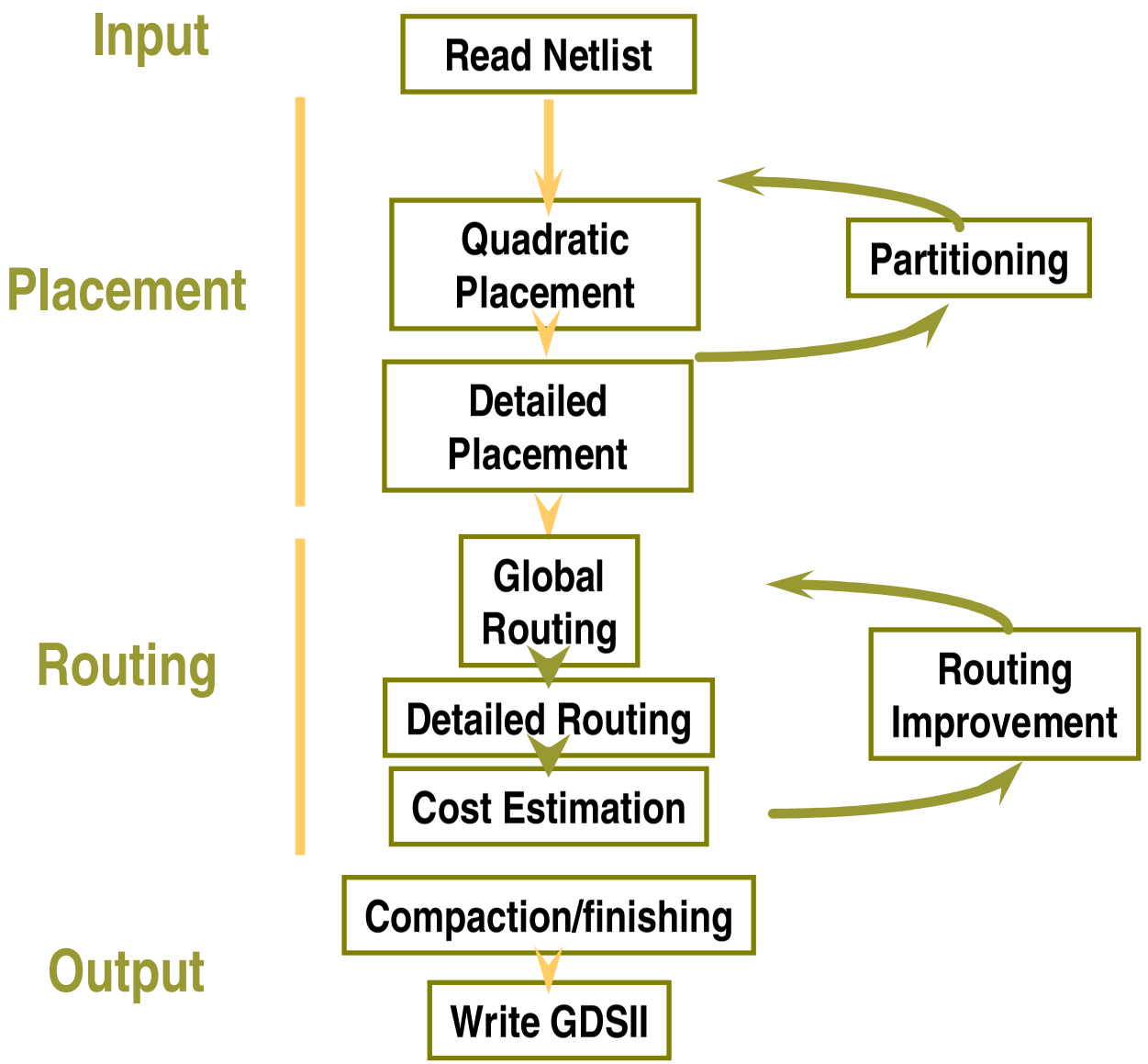


Channel less Cells

- Arbitrary routing over cells



Physical Design: Overall Flow



Gordian Placement Flow

J. Kleinhaus, G. Sigl, F. Johannes, K. Antreich,
GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization,
IEEE Trans. on CAD, March, 1991, pp. 356 - 365

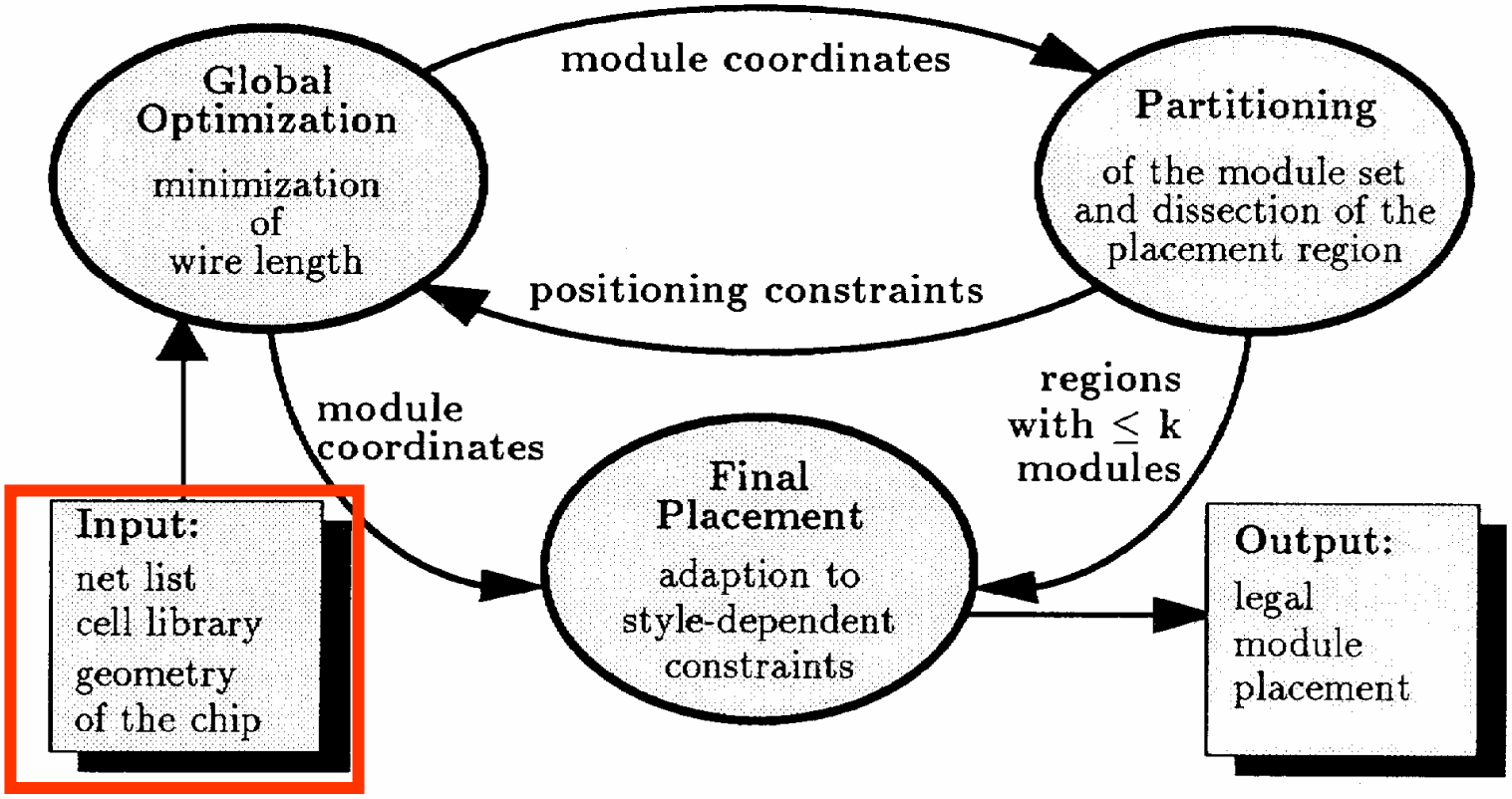
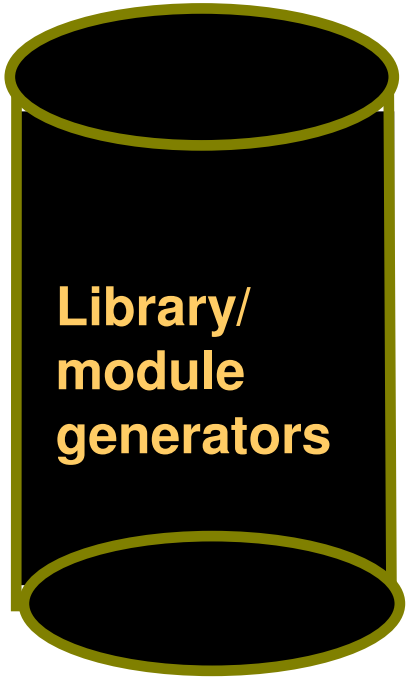
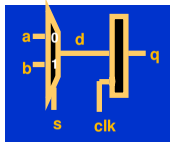


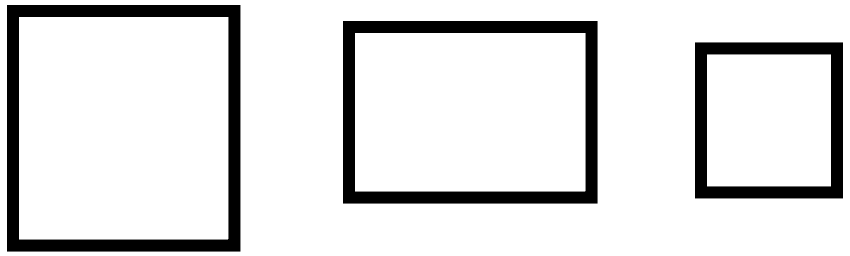
Fig. 1. Data flow in the placement procedure GORDIAN.

Library, Netlist, and Aspect Ratio



Netlist - >100K cells from library





Size and aspect ratio of core die

Setting up Global Optimization

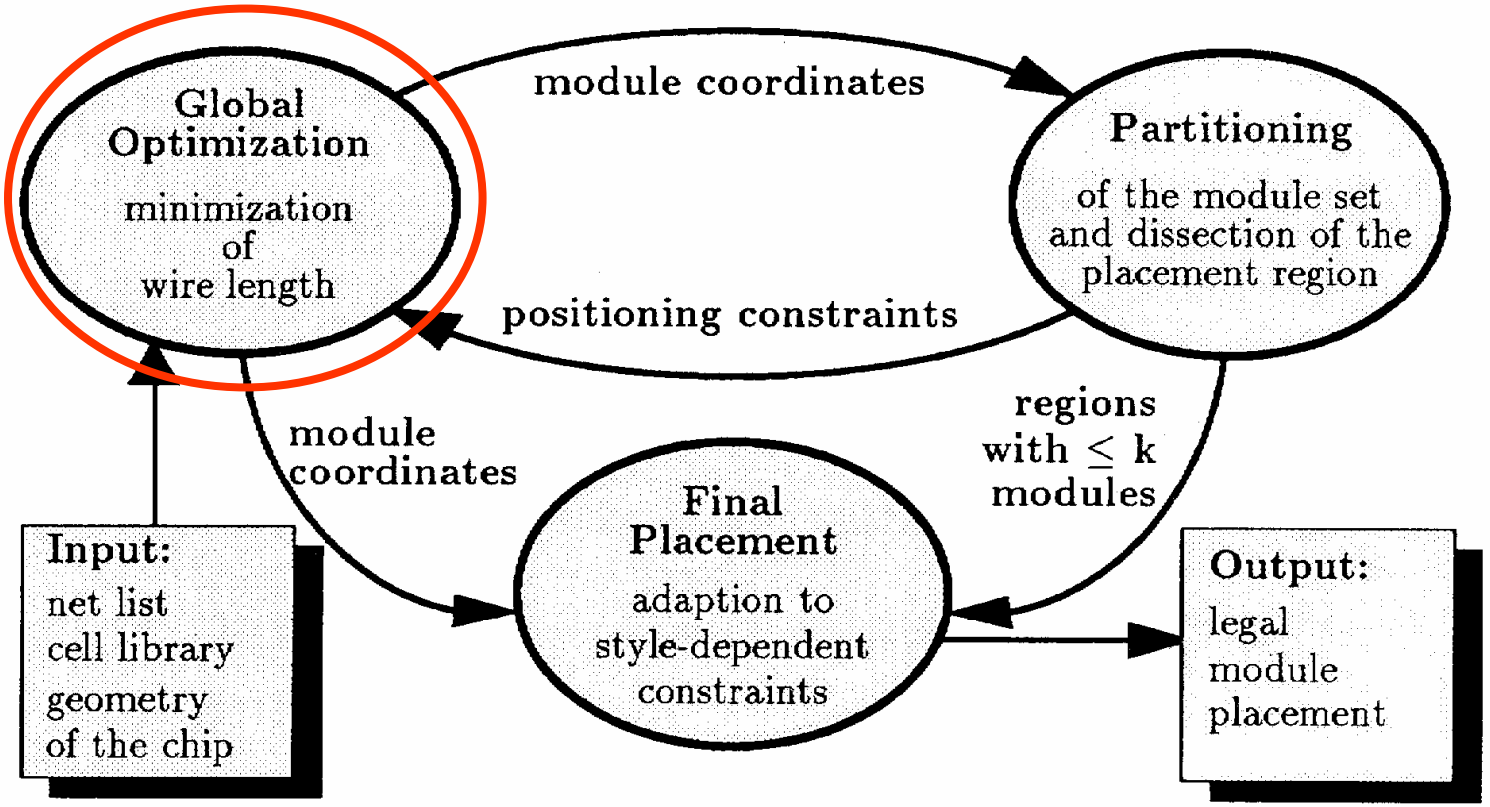
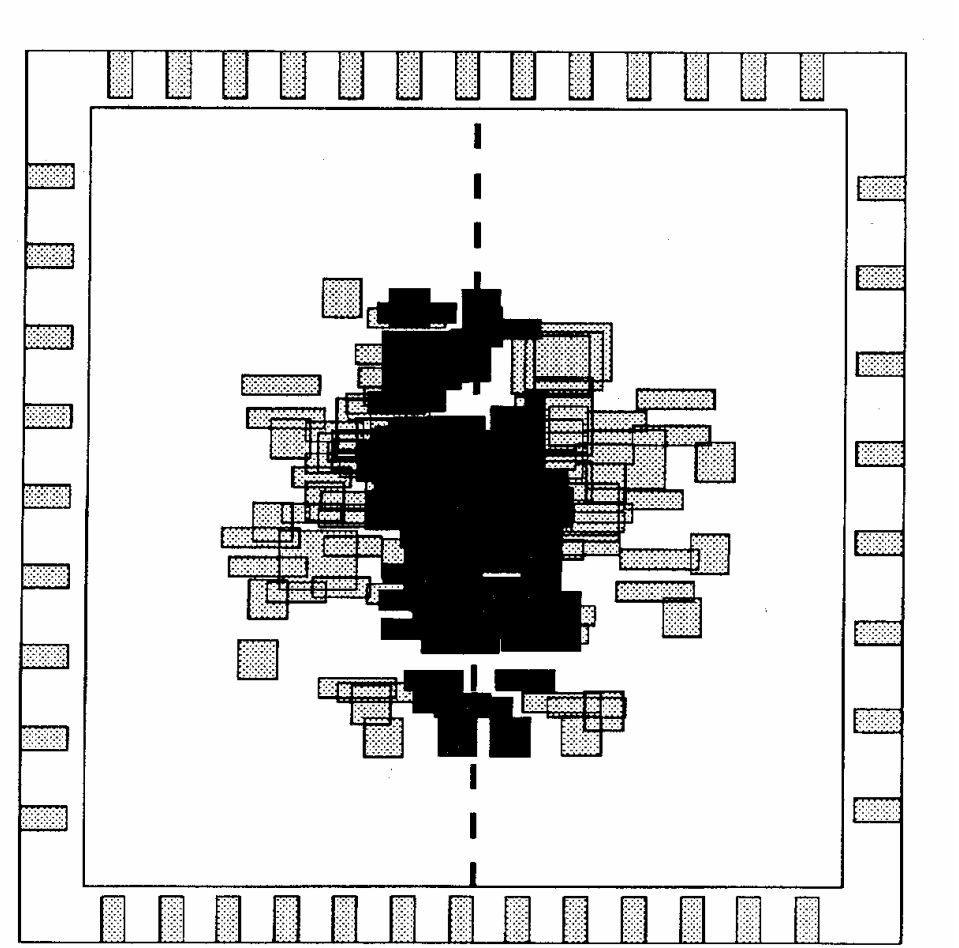


Fig. 1. Data flow in the placement procedure GORDIAN.

Resulting Layout



Partitioning

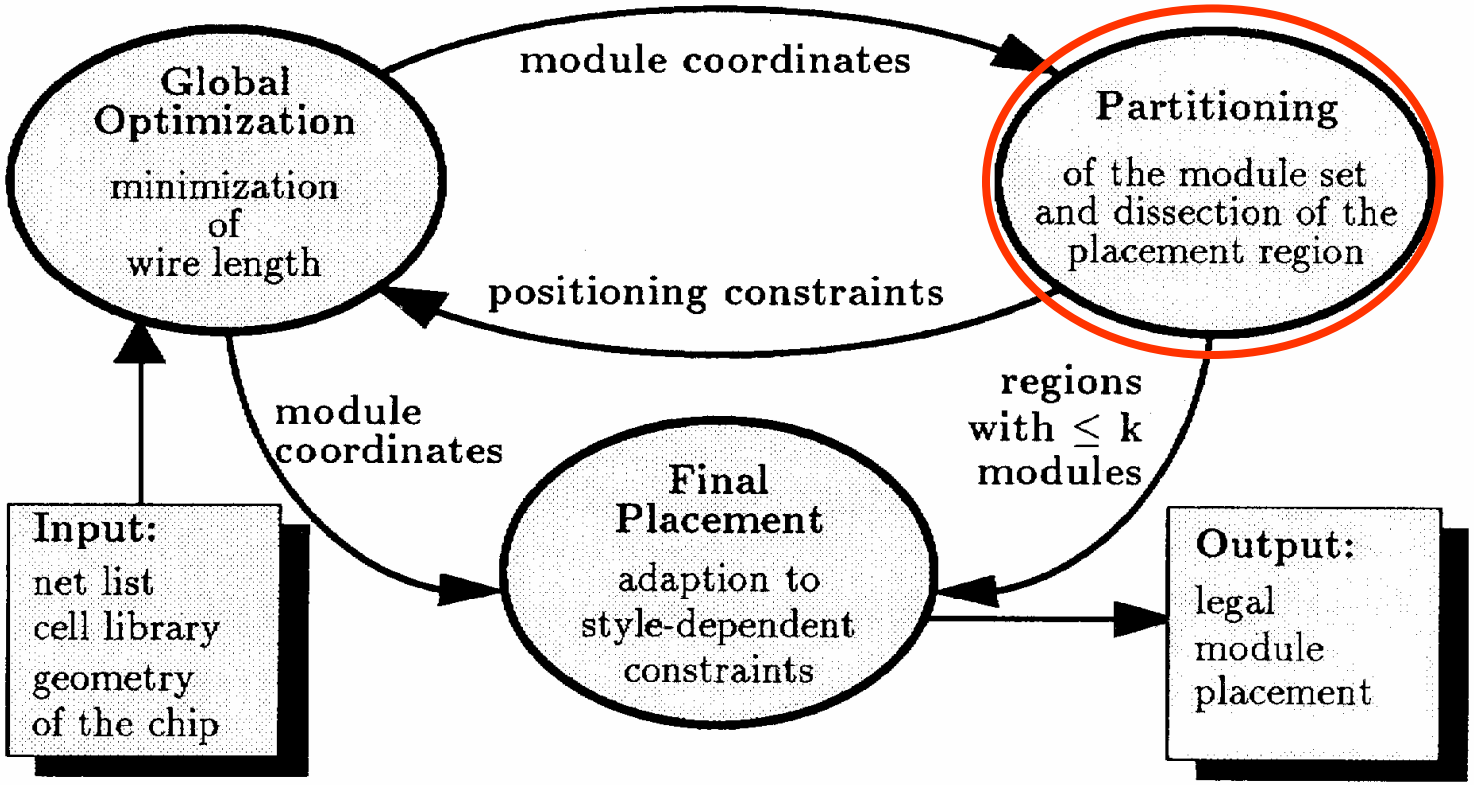
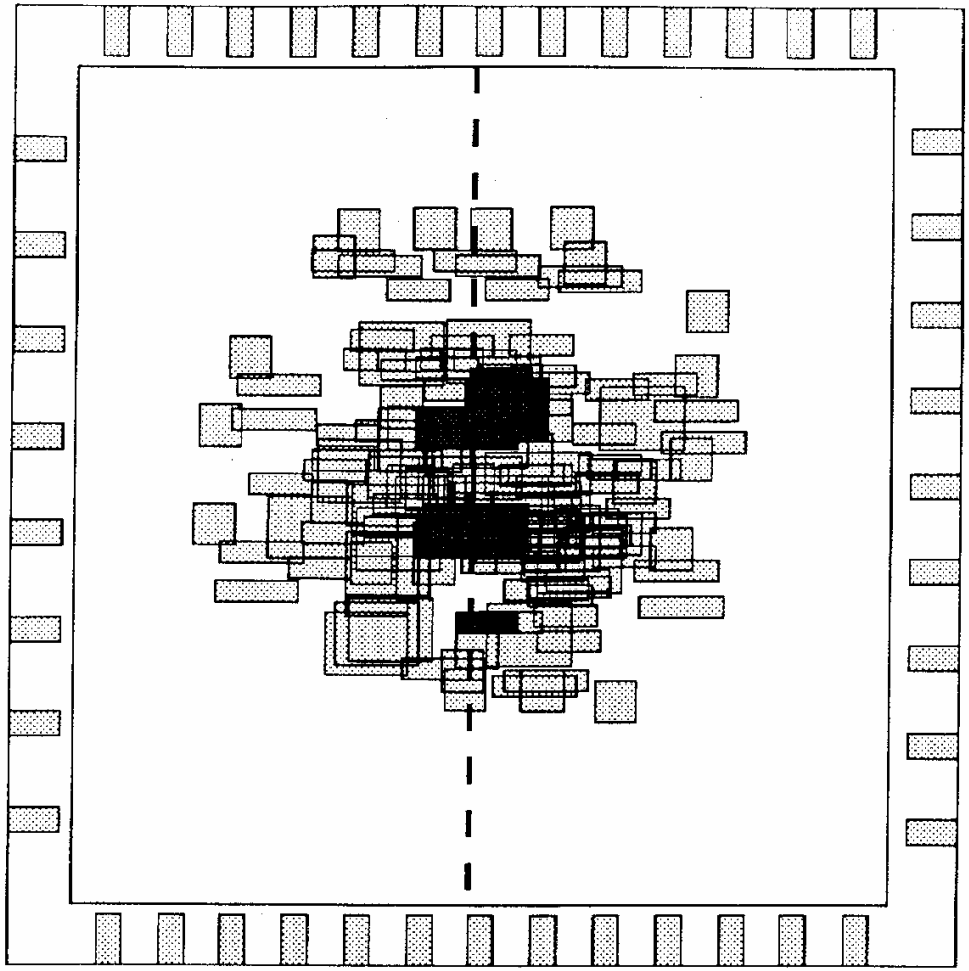


Fig. 1. Data flow in the placement procedure GORDIAN.

Layout after Min-cut



B.W. Kernighan, S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", Bell Syst. tech Journal, vol 49, 1970

A linear-time heuristic for improving network partitions, C. Fidduica, R. Mattheyses, Design Automation Conference, 1982, 175 – 181.

Final Placement

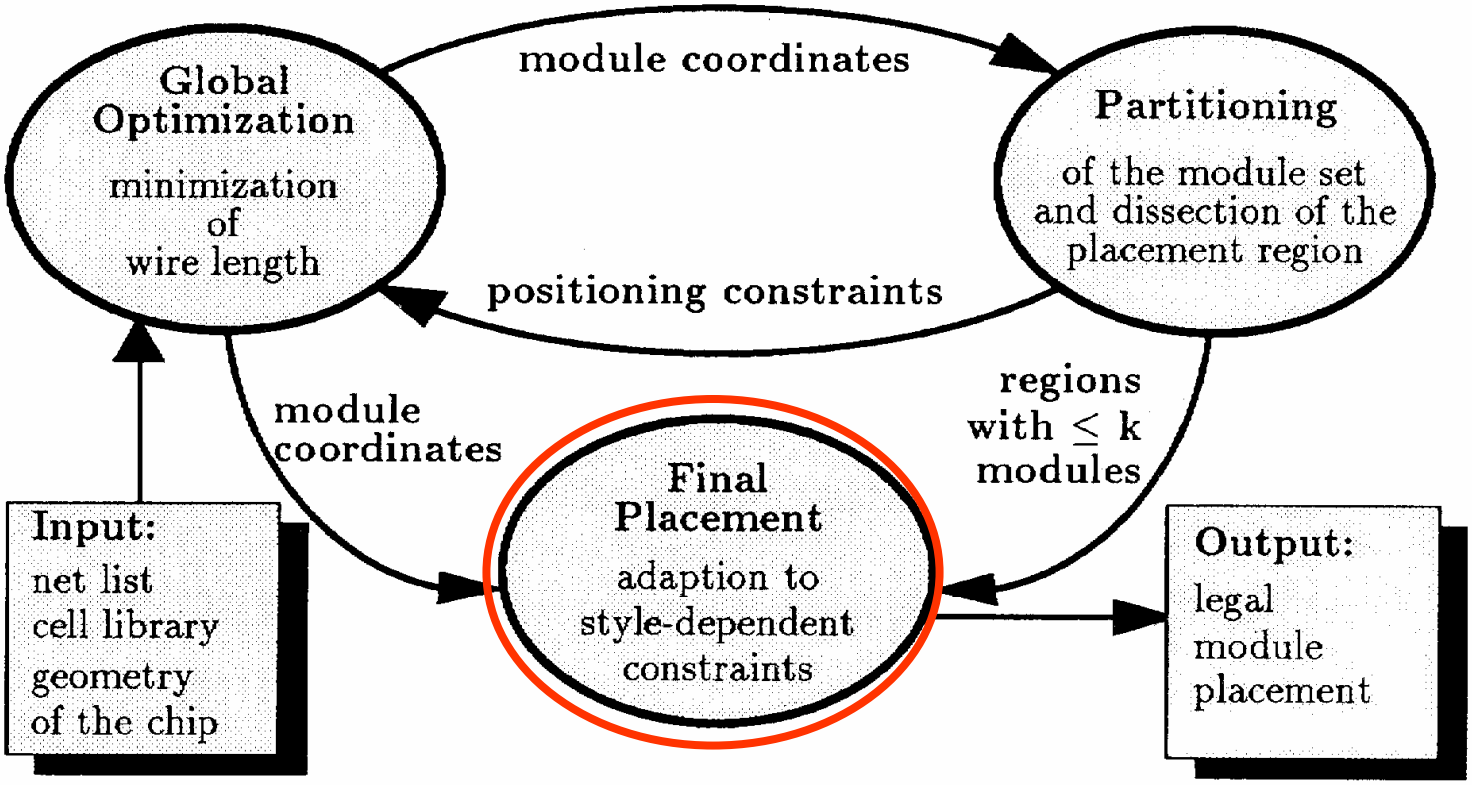
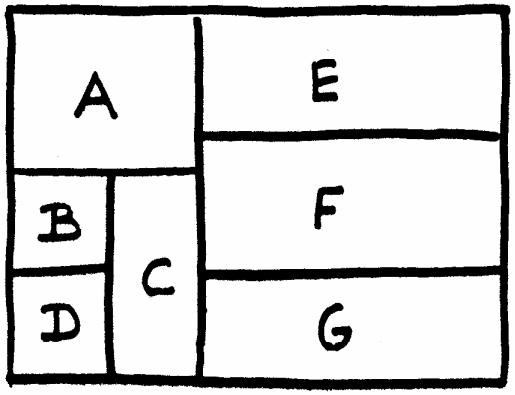
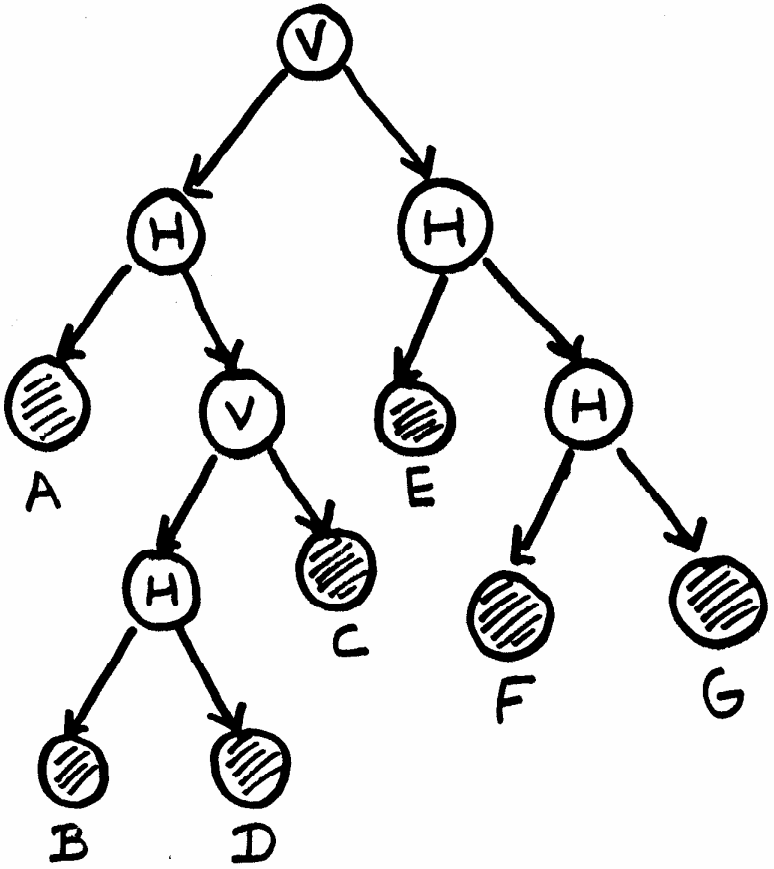


Fig. 1. Data flow in the placement procedure GORDIAN.

Slicing Tree



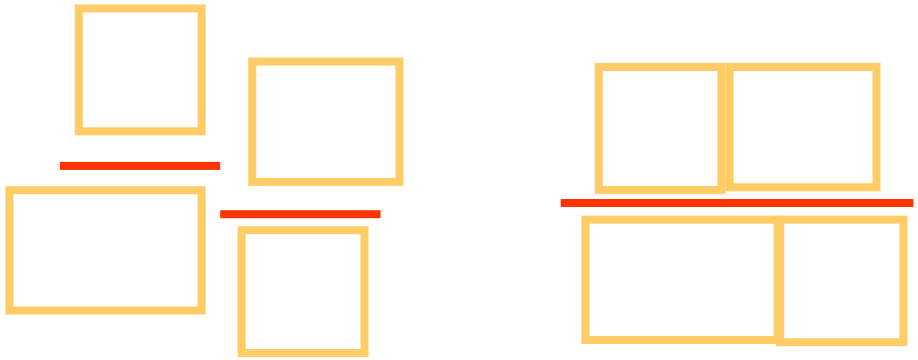


"Optimal slicing of plane point placements",
van Ginneken,L.P.P.P. Otten, R.H.J.M. ,
Proceedings of the European Design Automation Conference: 12-15 Mar 1990, 322-326.

Other details - slotting constraints

A. E. Dunlop, B. W. Kernighan,
A procedure for placement of standard-cell VLSI circuits, IEEE Trans. on CAD, Vol. CAD-4, Jan , 1985, pp. 92- 98

- **Slotting constraints**



Generating Final Placement

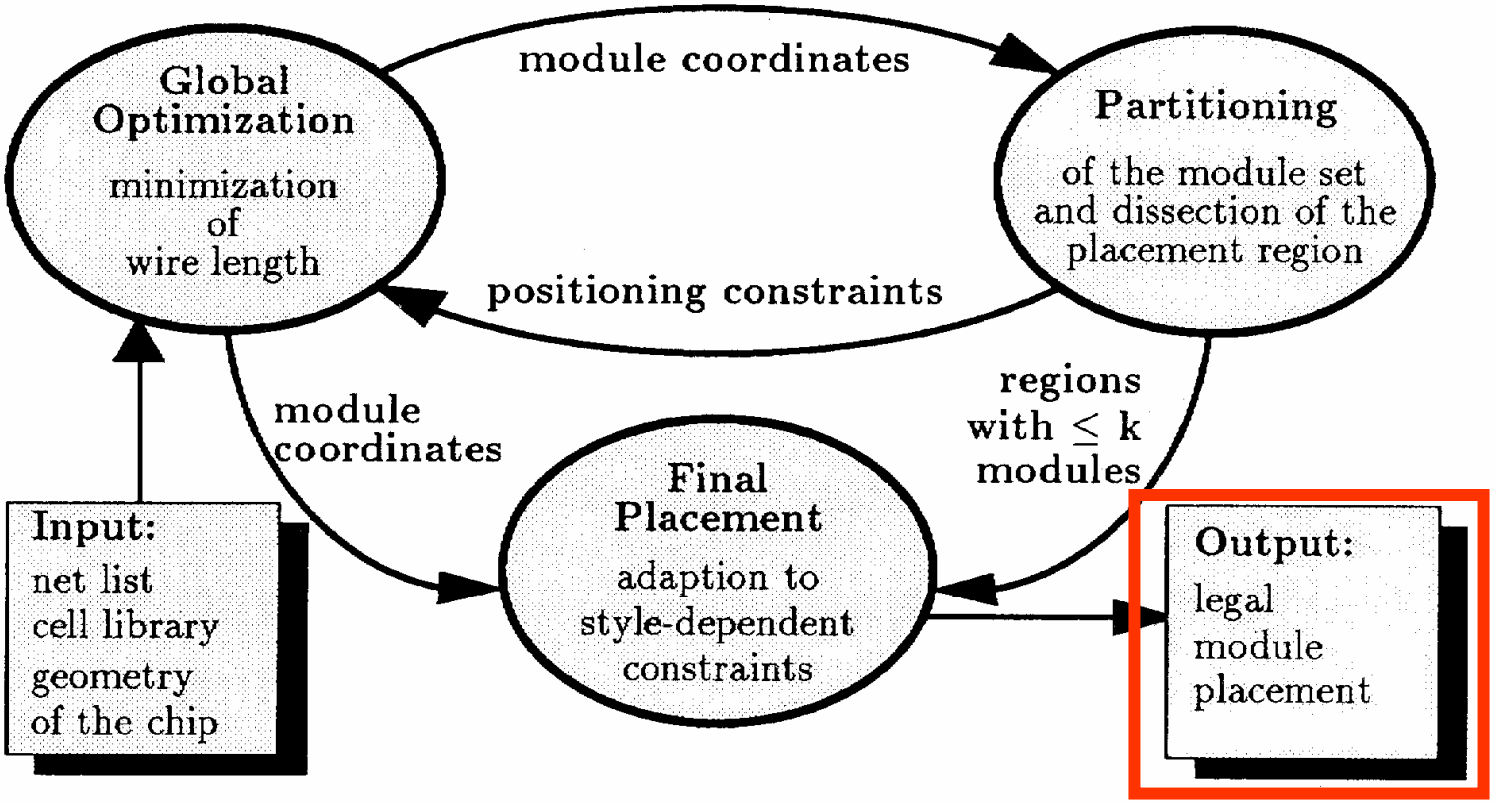
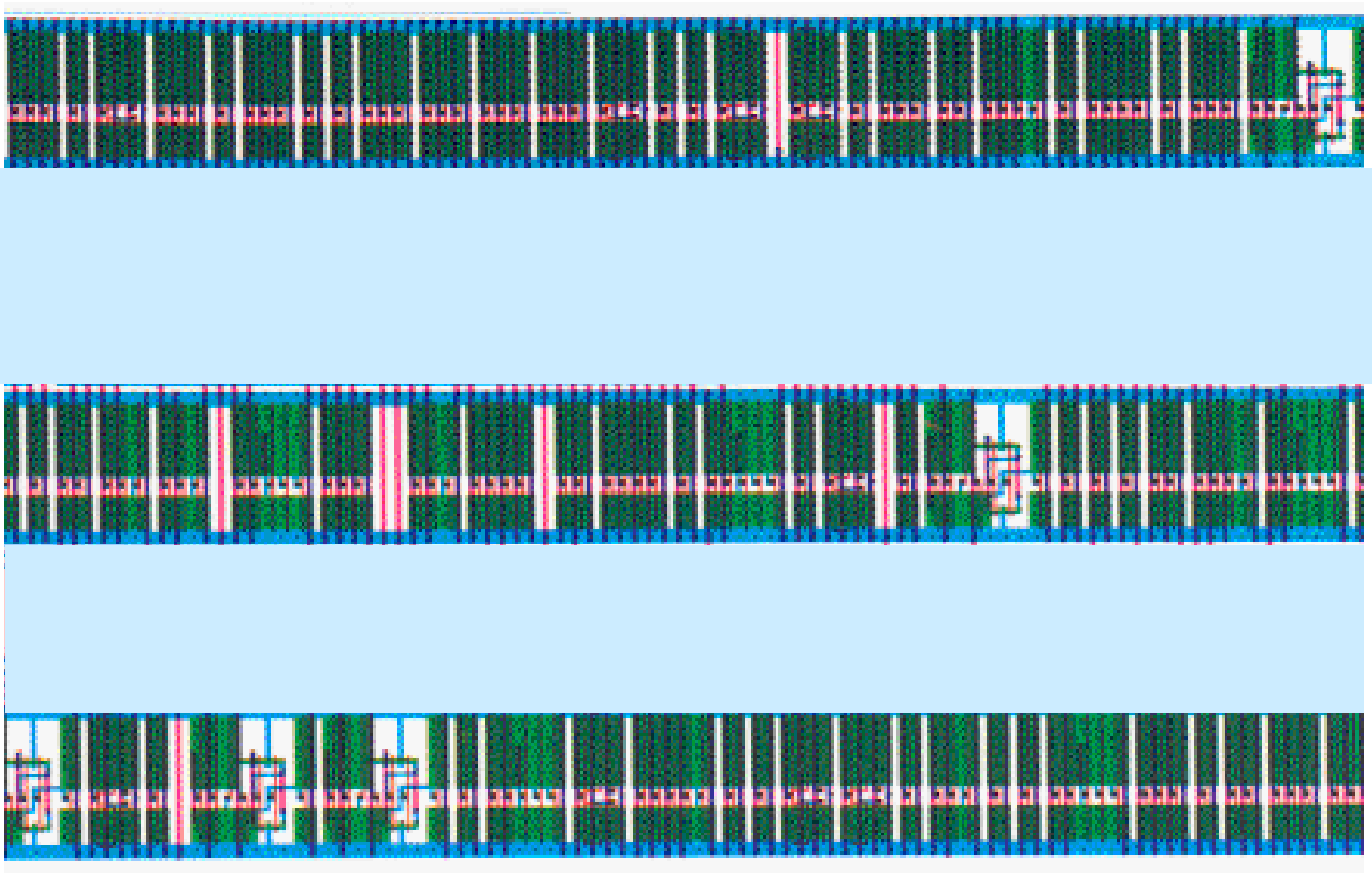
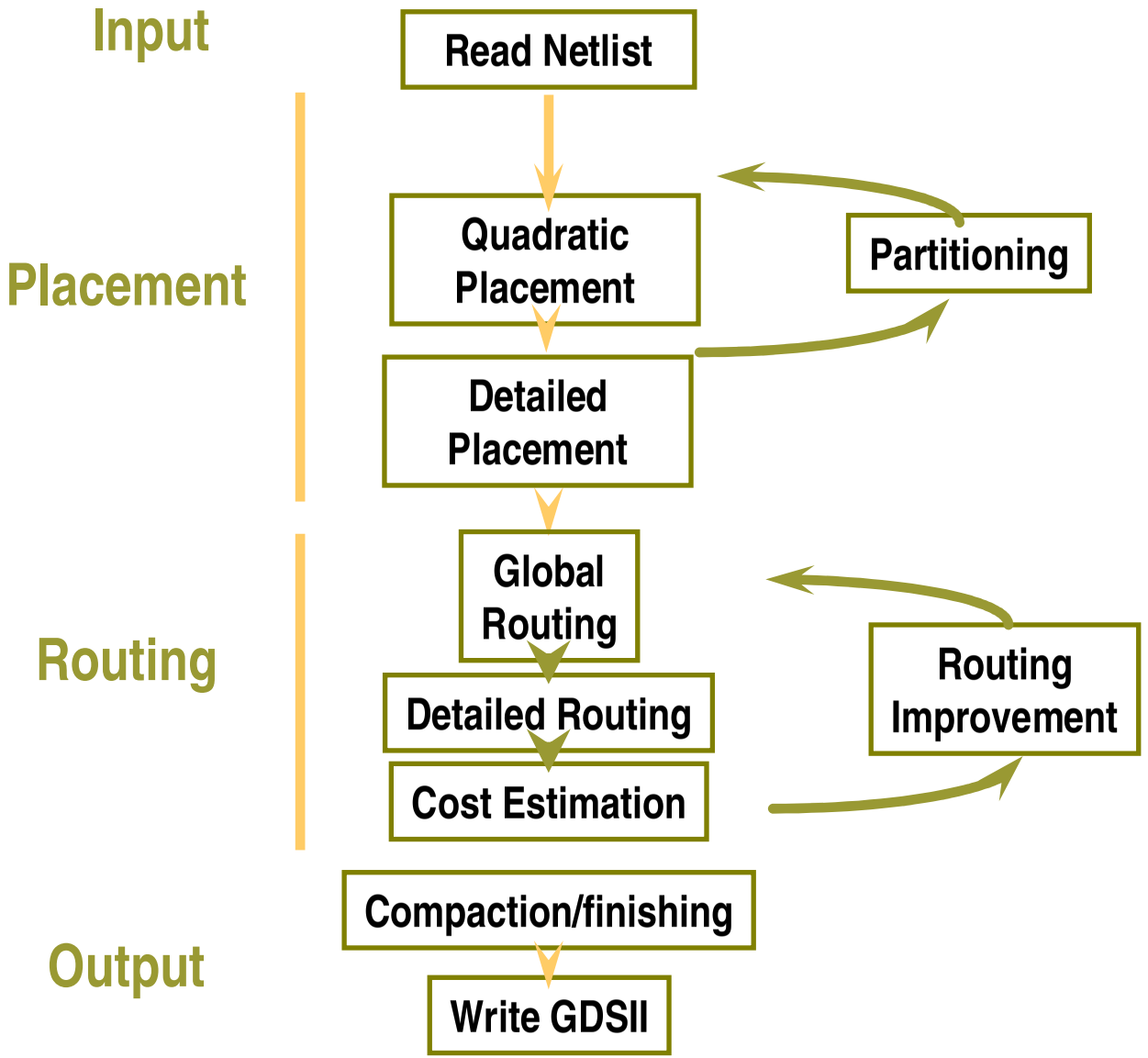


Fig. 1. Data flow in the placement procedure GORDIAN.

Standard Cell Layout



Physical Design: Overall Flow



Routing

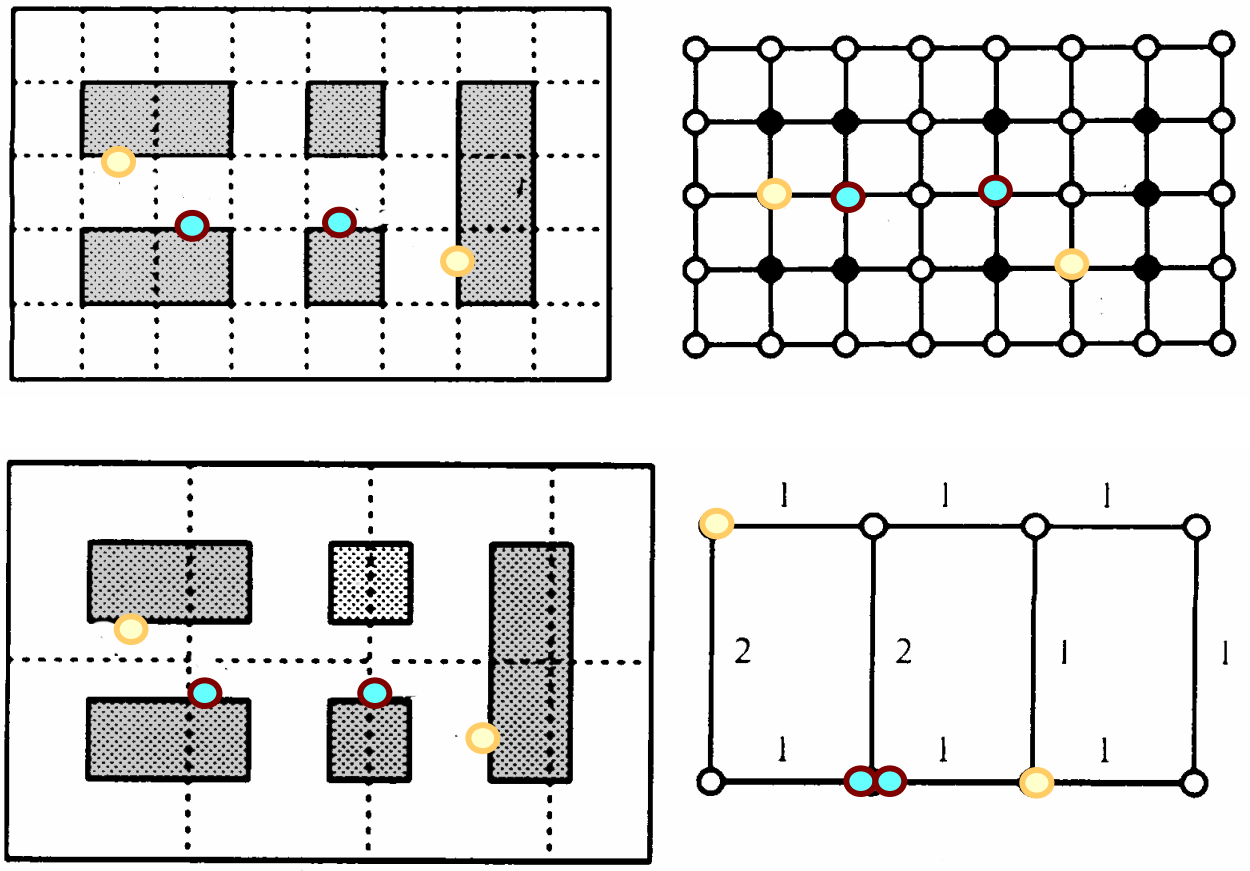
- To simplify routing problem, divide it into two phases
  - Global
  - Detailed

- Global routing
  - Define routing regions
  - Assign nets to regions

- Detailed (Channel) routing
  - Route nets within each region
  - Assign nets to pins

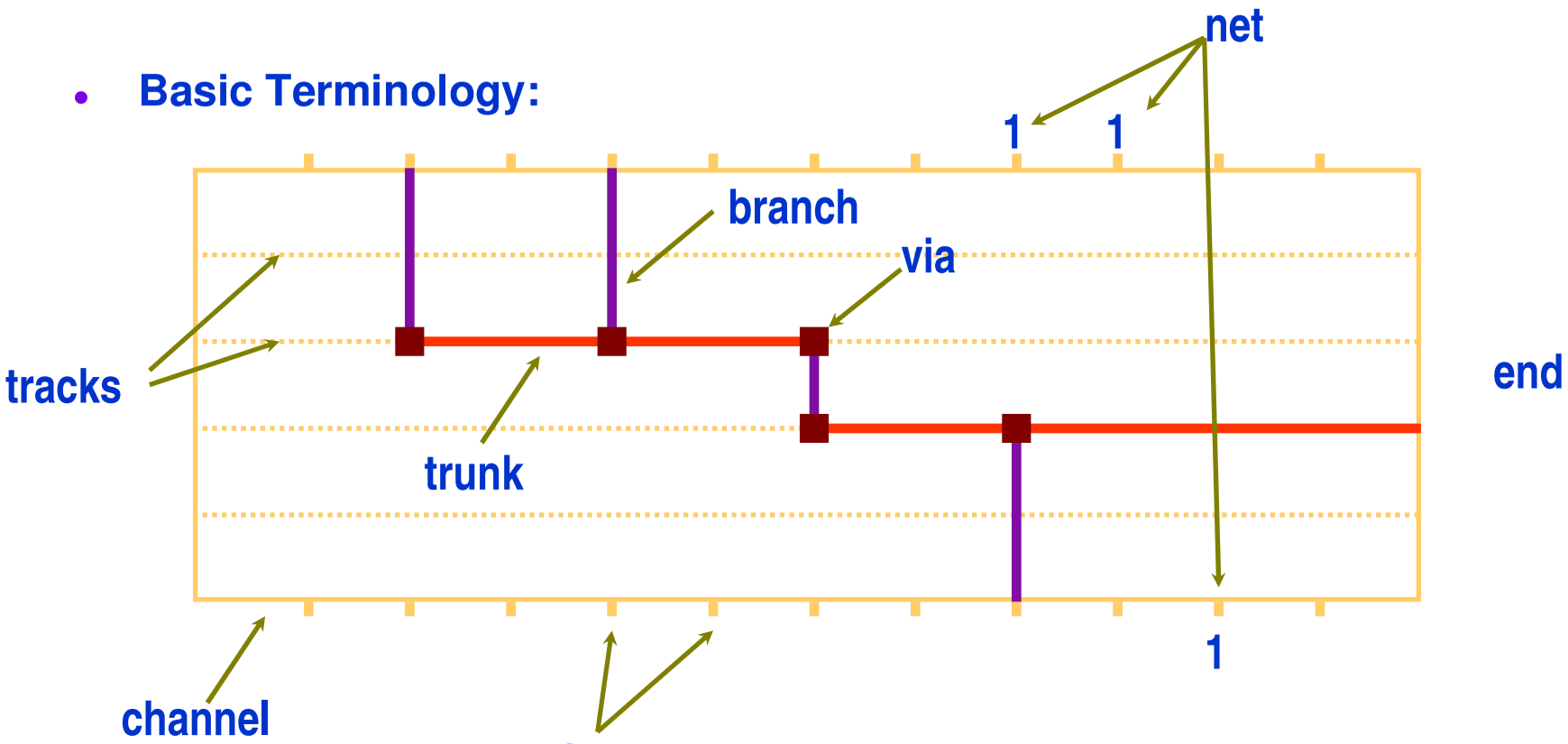
Global Routing

- Grid-Graph Model
- Checker-Board Graph (also use slicing structure)



Channel Routing

- Basic Terminology:



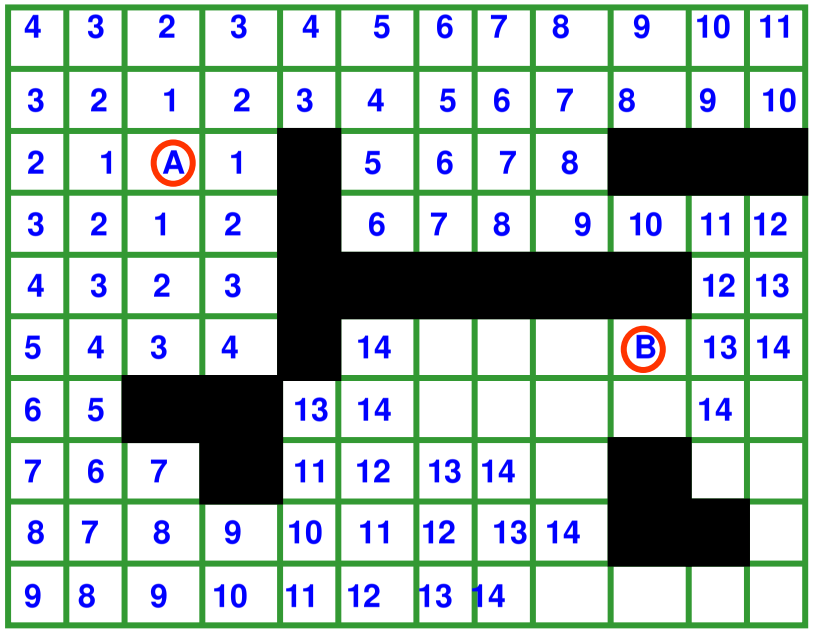
- Fixed pin positions on top and bottom edges
- Classical channel: no nets leave channel
- Three-sided channel possible

A "Greedy" Channel Router,
RL Rivest, CM Fiduccia,
Design Automation Conference, 1983

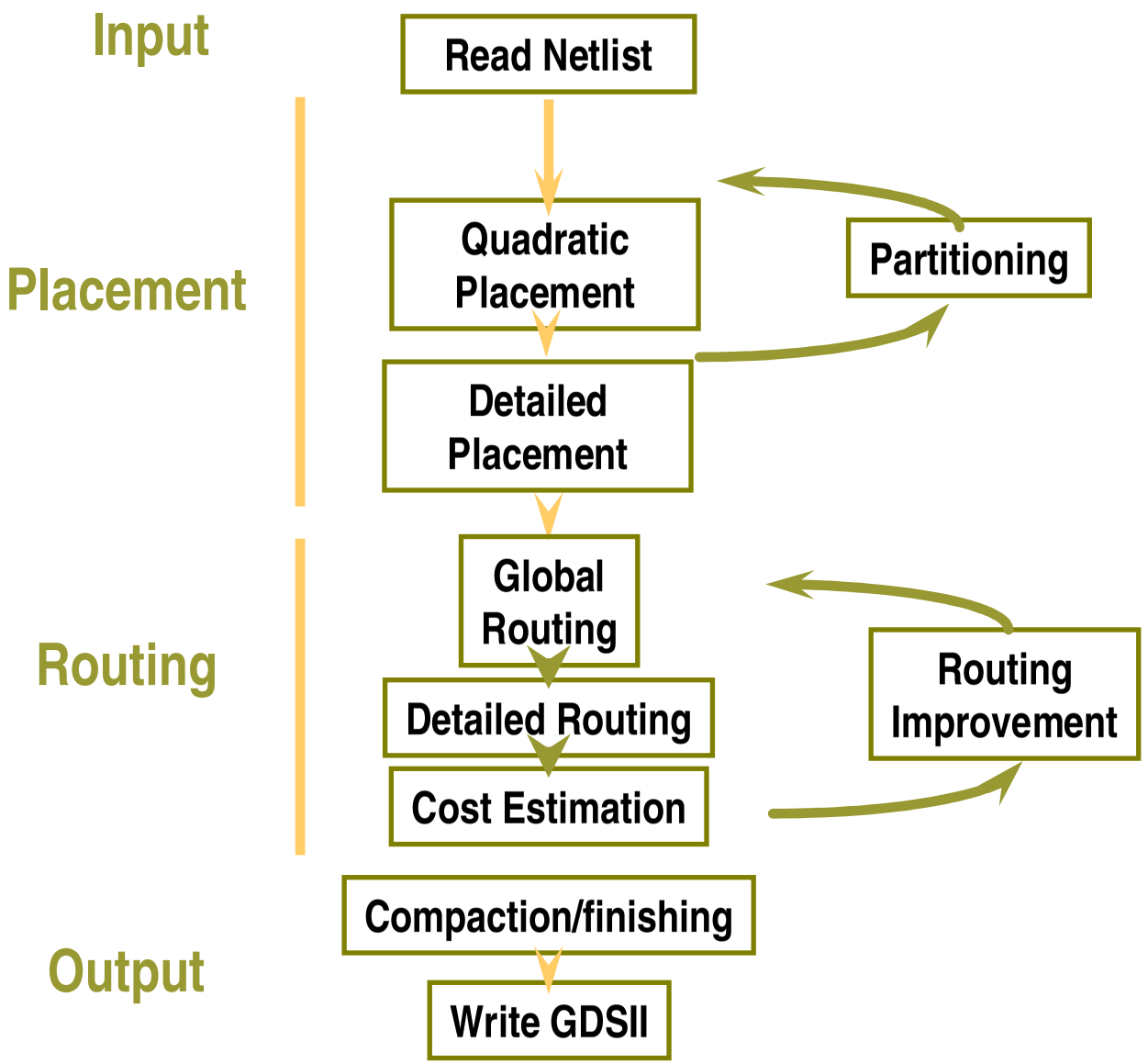
Detail Router 2: Maze Routing

Basic idea -- wave propagation method(Lee, 1961)

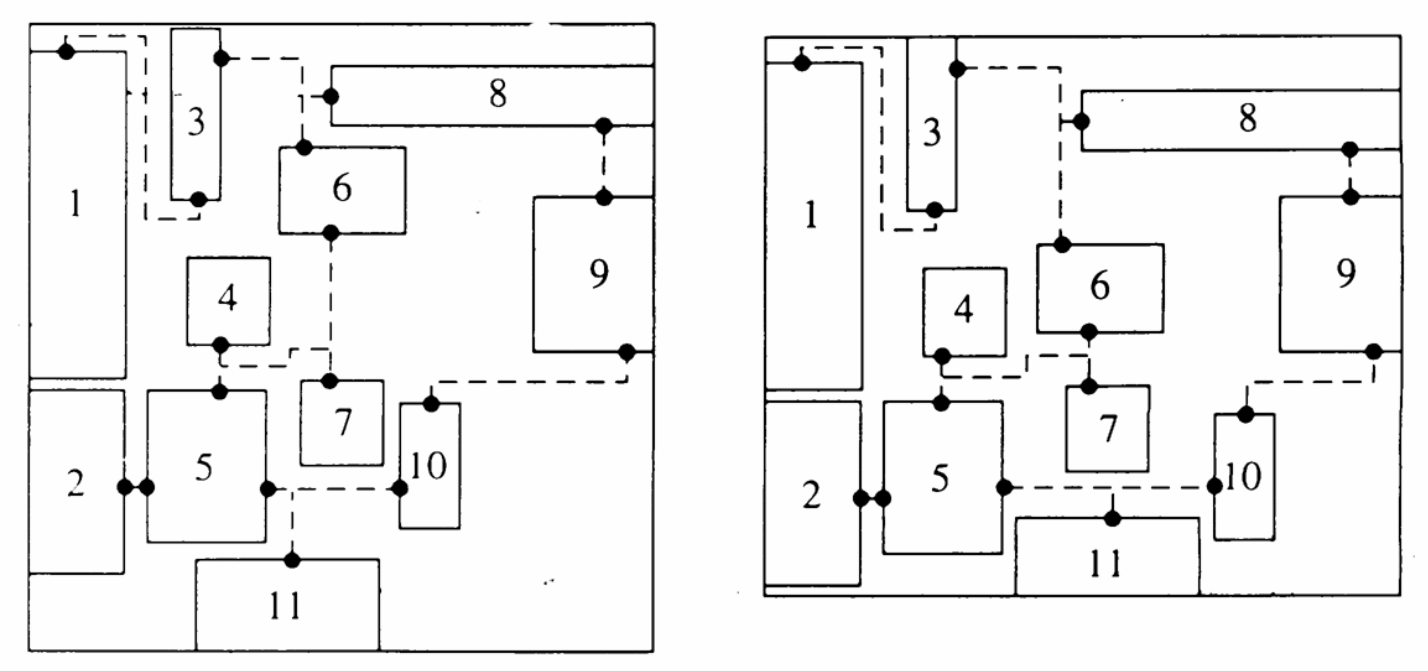
- **Breadth-first search**
- **Back-tracing after finding the shortest path**
- **guarantee to find the shortest path**



Physical Design: Overall Flow



Followed by Compaction

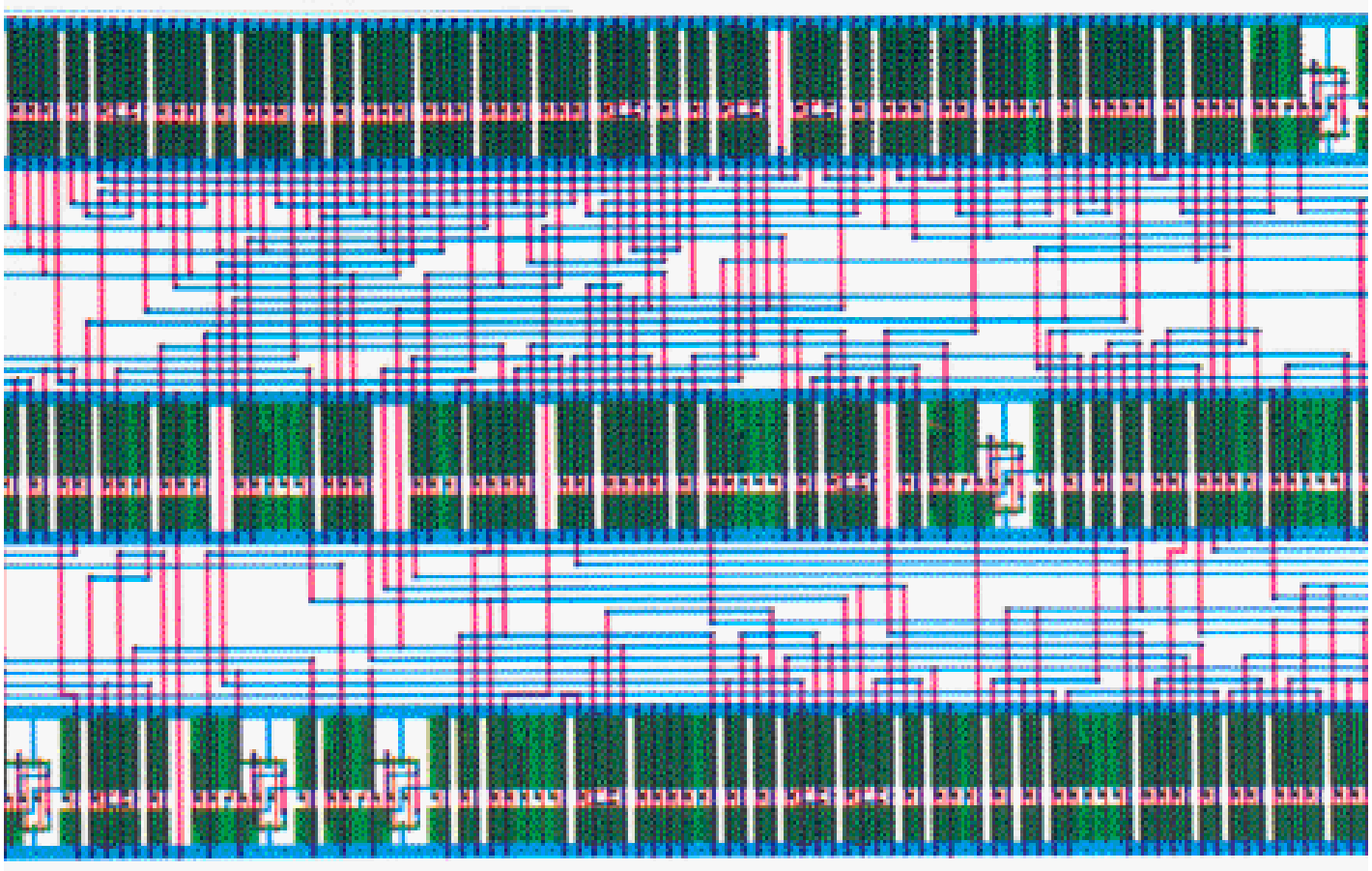


X then Y 1D Compaction Y then X 1D Compaction

Thomas Lengauer, "Combinatorial Algorithms for Integrated Circuit Layout", Chapter 10, "Compaction", pages 579-649, Wiley and Teubner, 1990

Algorithms and techniques for VLSI layout synthesis
DD Hill, D Shugard, J Fishburn, K Keutzer - 1989 - Boston: Kluwer Academic Publishers

Placed and Routed Standard Cells



Another Final Placement

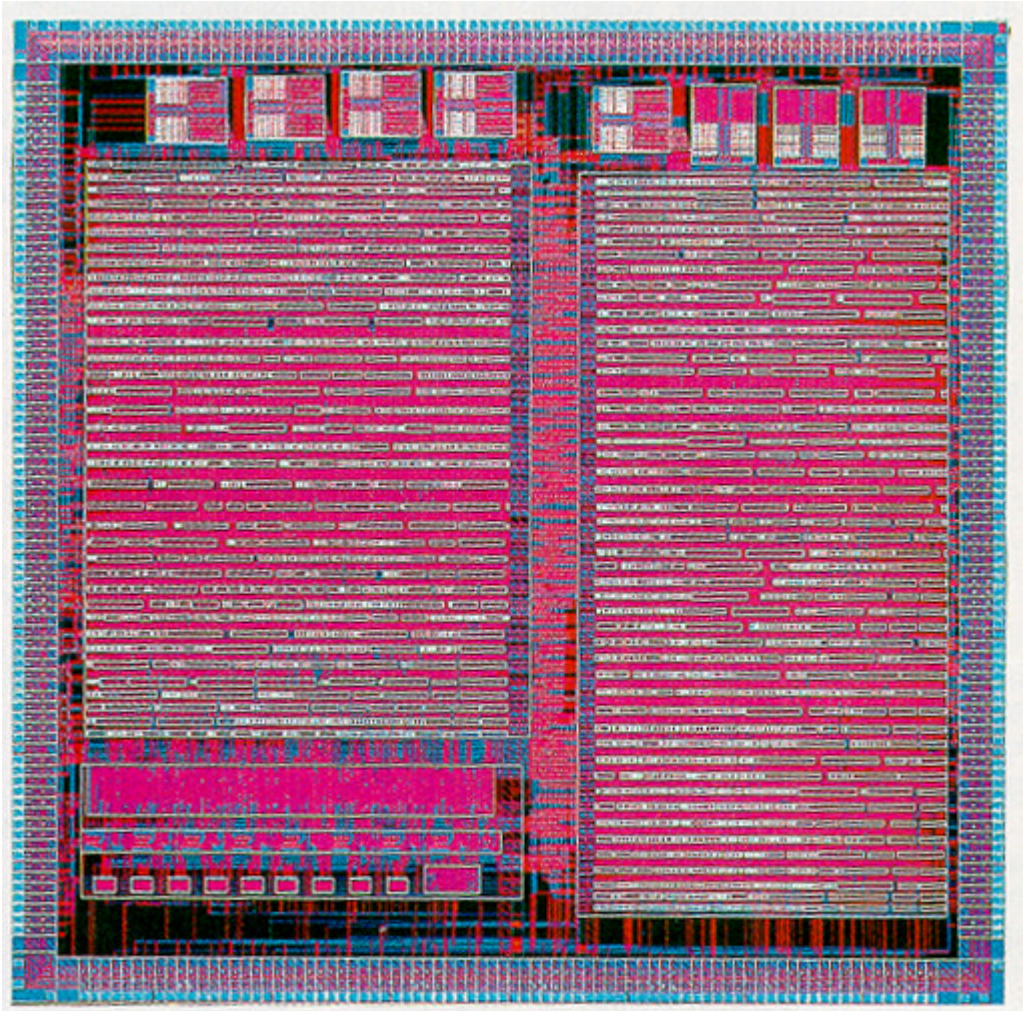
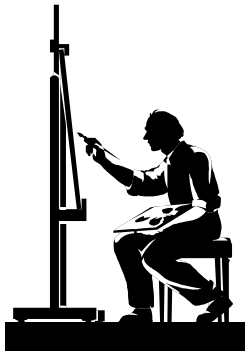


Fig. 10. Macrocell design with standard cell blocks *scb8* and *scb9*.

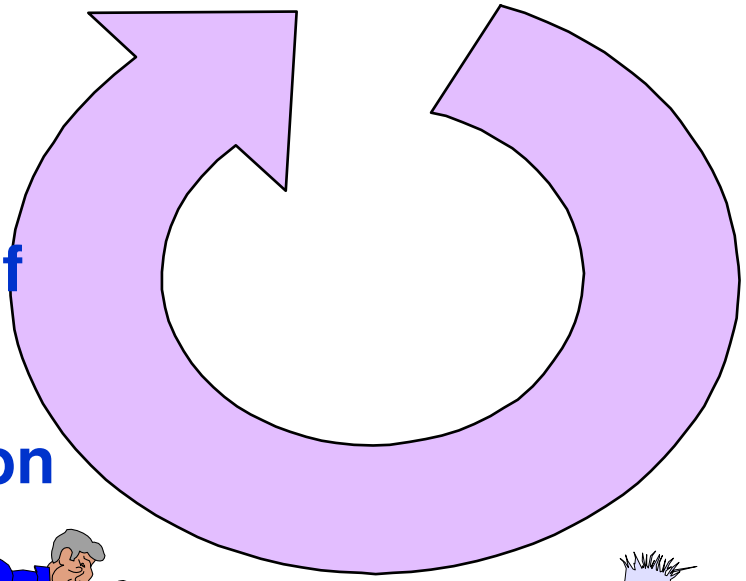
Re-visiting Verfication

- **Design** : specify and enter the design intent



Verify:

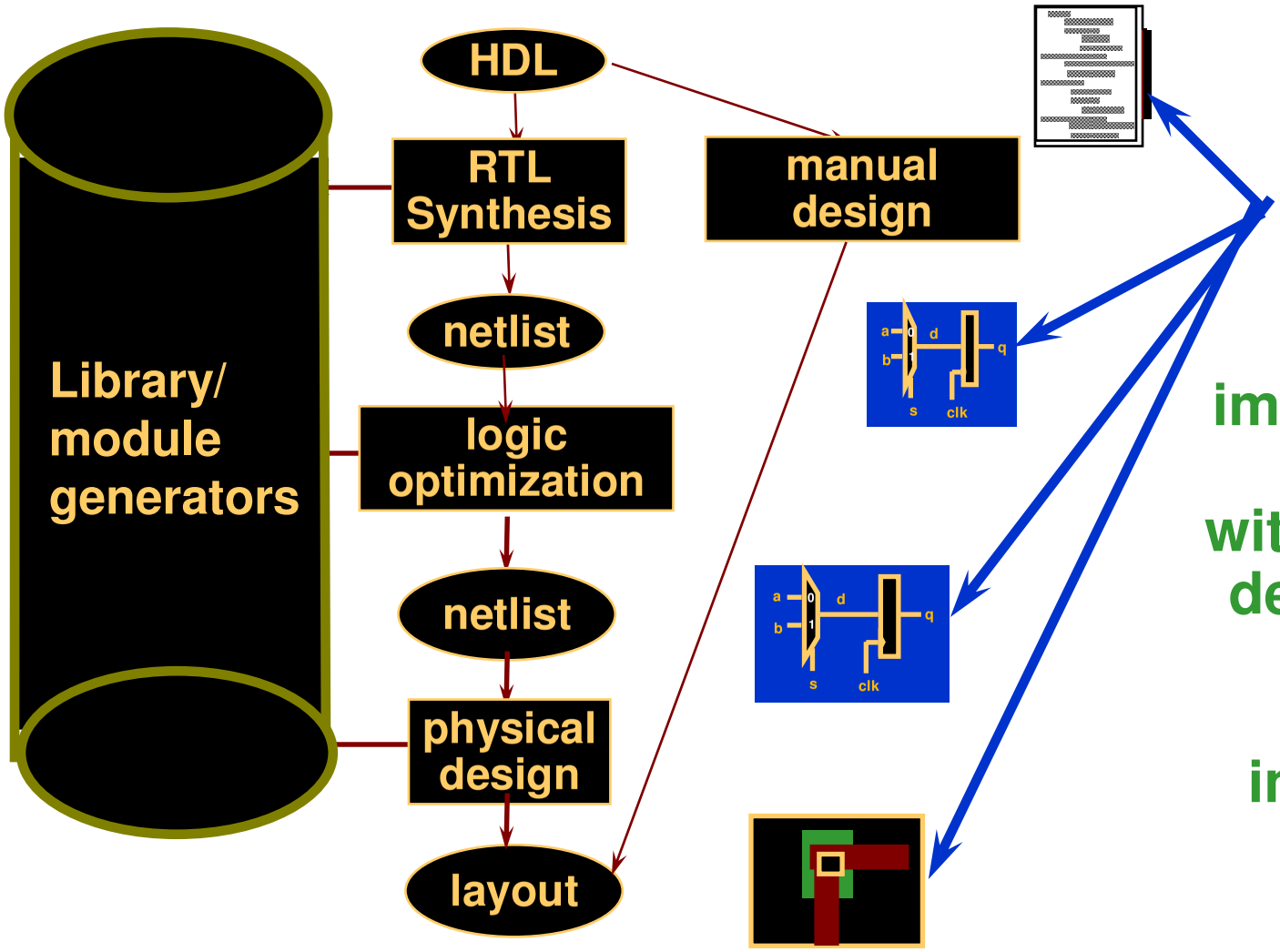
verify the correctness of design and implementation



Implement:

refine the design through all phases

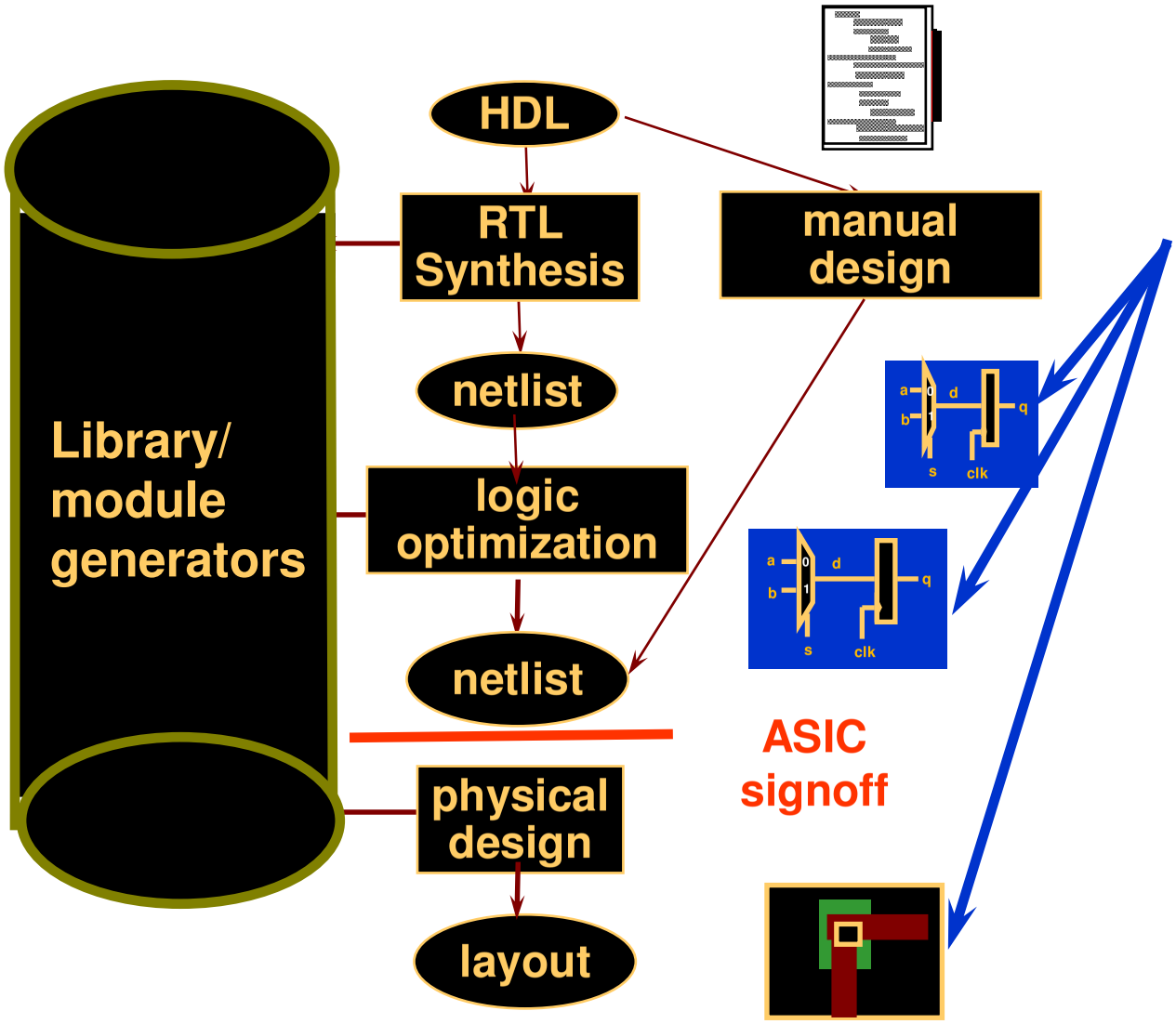
Implementation Verification



Is the implementation consistent with the original design intent?

Is what I implemented what I wanted?

Static Sign-off

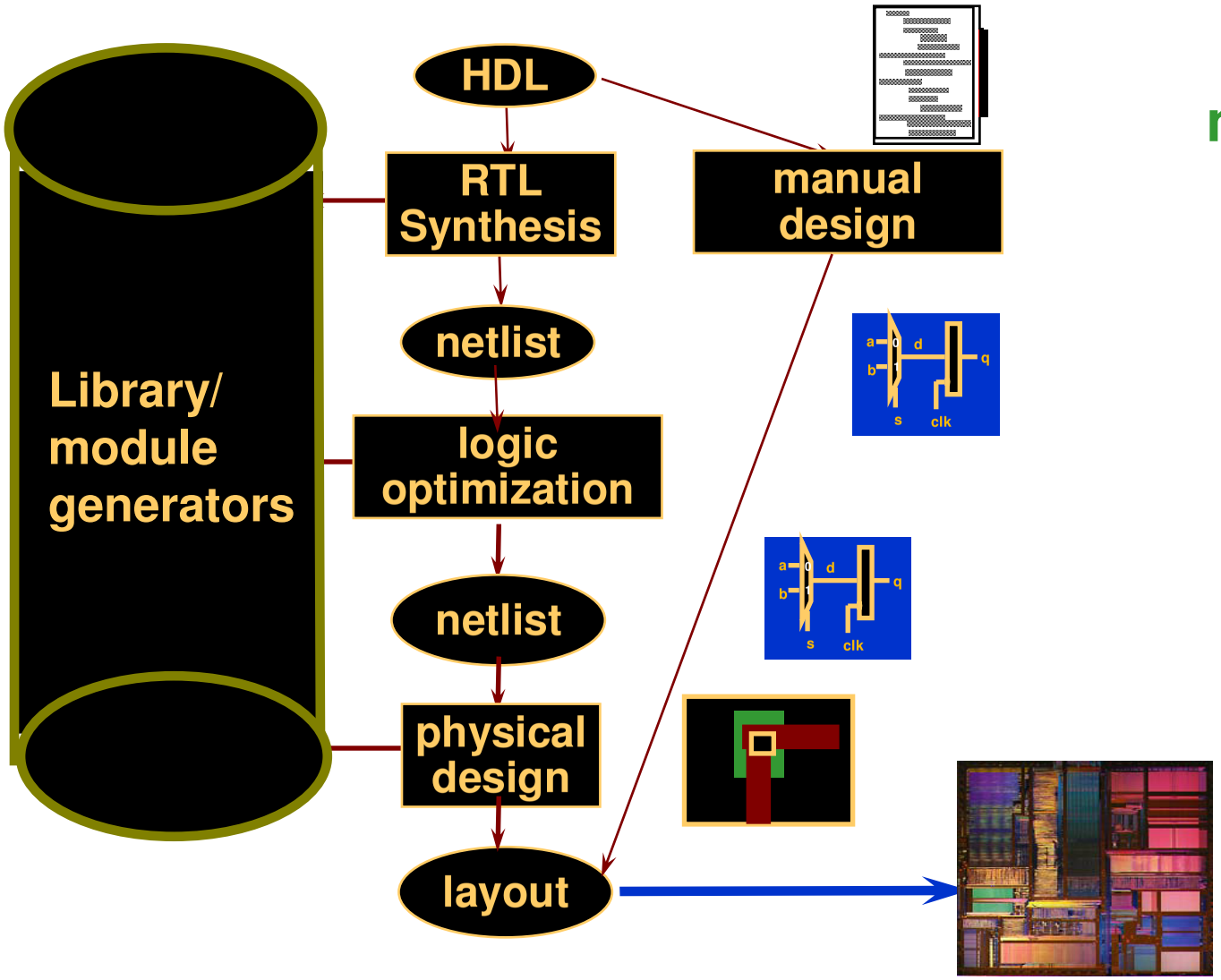


Use static analysis techniques to verify:

functionality:
- *formal equivalence-checking techniques*

and timing:
- use static timing analysis

Manufacture Verification (Test)

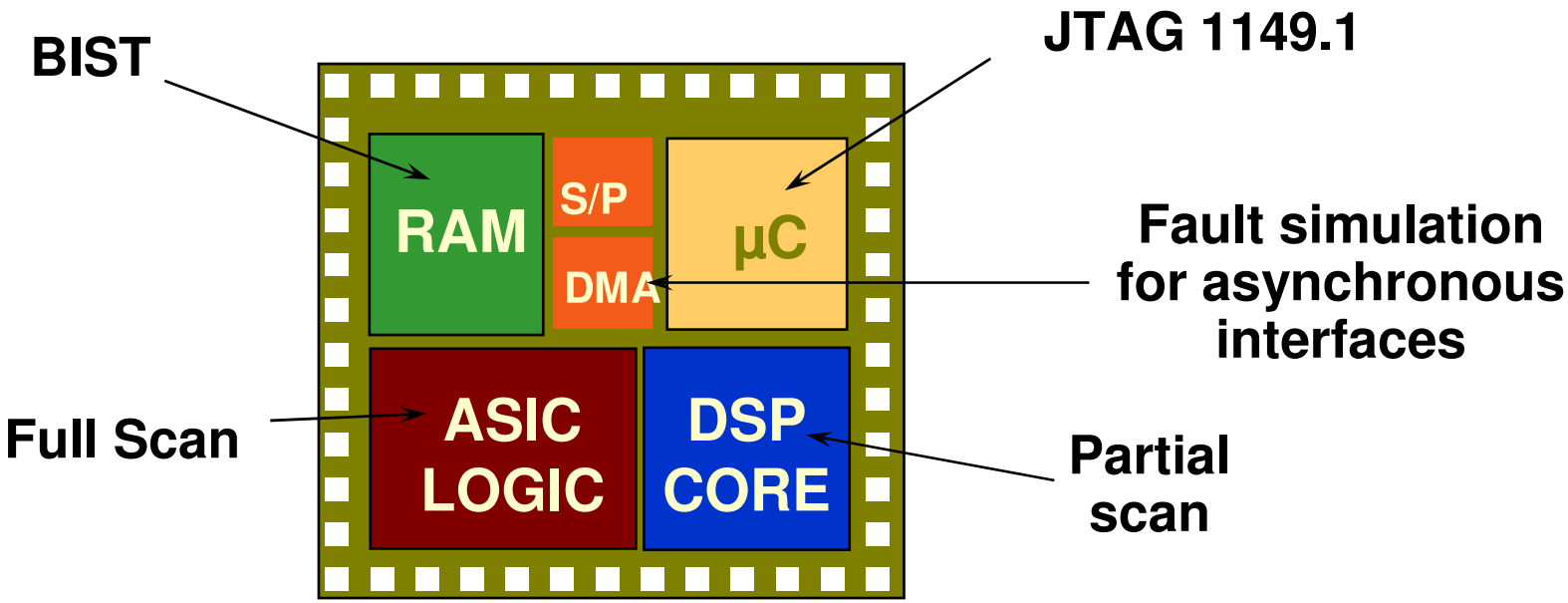


Is the manufactured circuit consistent with the implemented design?

Did they build what I wanted?

Test Synthesis

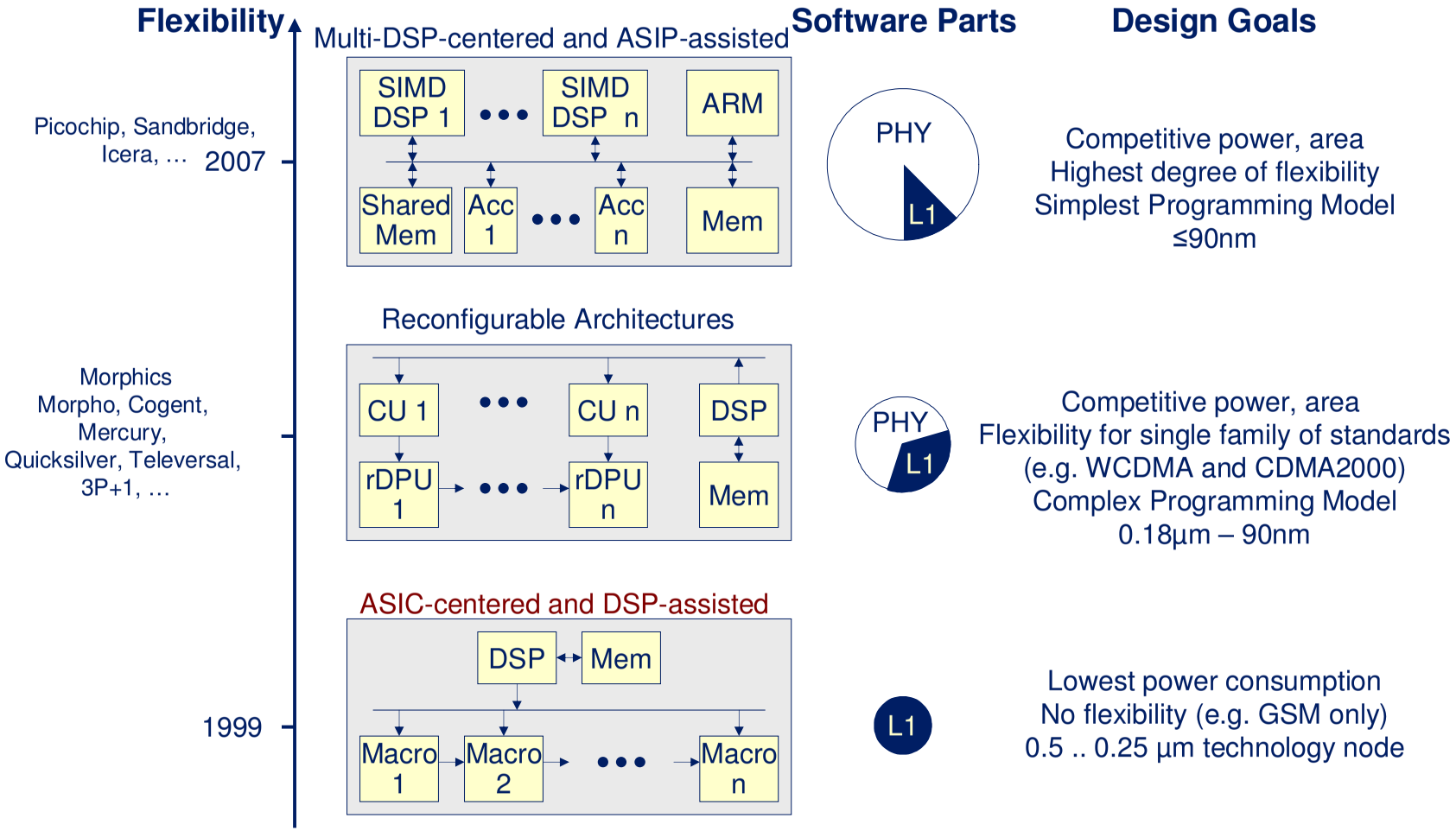
- Full-chip test requires:



Current Status of RTL Design Flow

- Current RTL design flow is able to produce
  - High speed microprocessors - e.g. Alpha, Pentium Pro > 10M gate-equivalents > 4GHz. (with intervention)
  - System-on-a-chip/Systems on silicon
    - Integration of micro-p, DSP, memory and ASIC on a single die > 10M gate-equivalents >500Hz.
  - Rapid turnaround "Structured" ASIC
    - ISSP-90 HIS from NEC
    - ~ 3M usable gates
    - ~10 Mb SRAM; 10G SerDes
  - RTL Design flow now used for FPGA's as well
    - RTL synthesis provided by independent vendors – e.g. Synplicity – as well as FPGA providers – e.g. Xilinx
    - Place and route provided by FPGA vendors – e.g. Xilinx

How close or far are these 3 solutions In terms of area, power, programmability ???



Acc: Accelerator
CU: Control Unit
DSP: Digital Signal Proc.

L1: Layer-1 Control
Mem: Memory
PHY: Physical Layer

rDPU: Reconf. Data Processing Unit
SIMD: Single Instruction Multiple Data