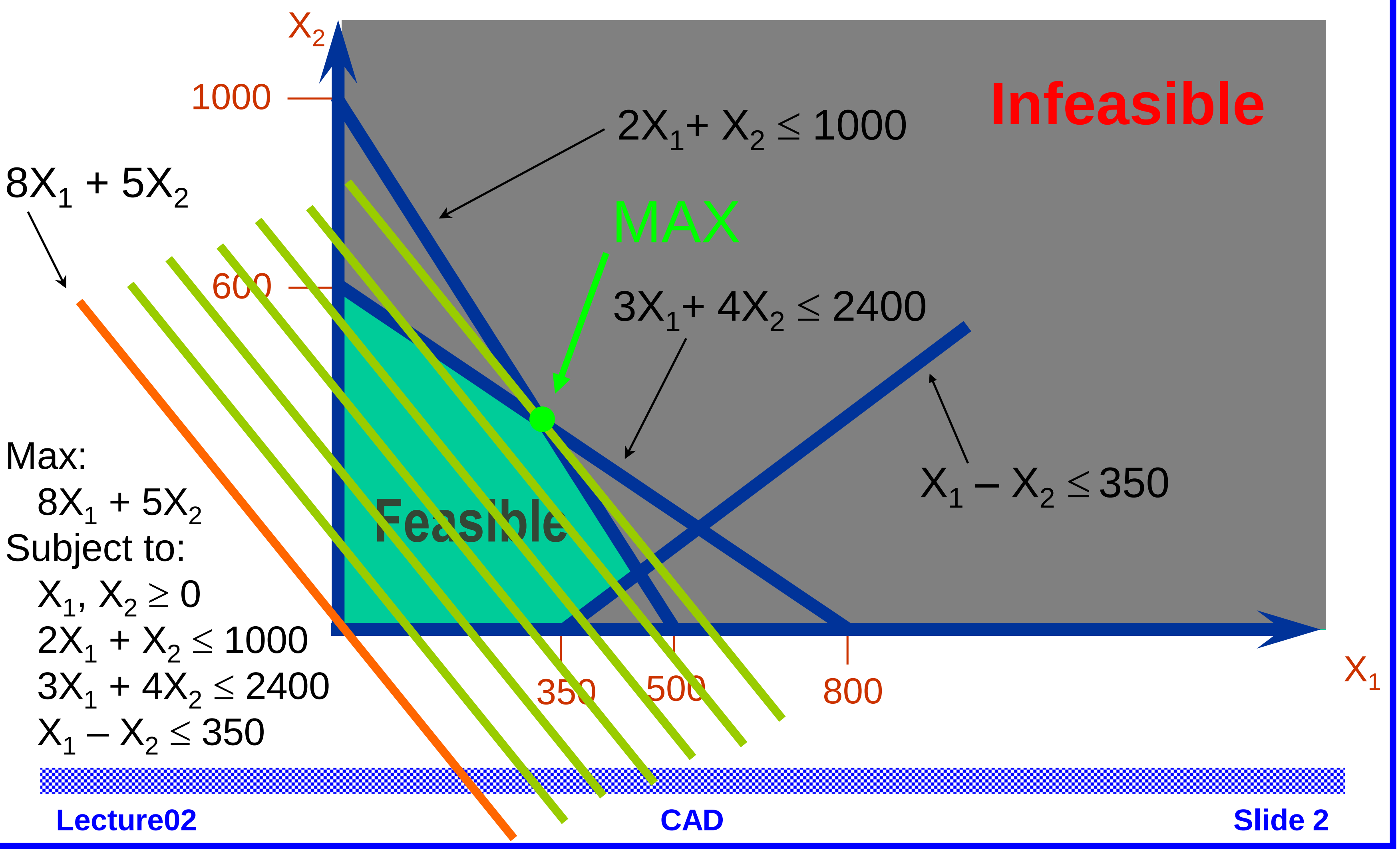
CAD for VLSI

Modeling

Linear Programming Example



Max:

$8X_1 + 5X_2$

Subject to:

$X_1, X_2 \geq 0$

$2X_1 + X_2 \leq 1000$

$3X_1 + 4X_2 \leq 2400$

$X_1 - X_2 \leq 350$

Mixed Integer Linear Programming

- A mathematical programming such that:
  - The objective is a linear function
  - All constraints are linear functions
  - Some variables are real numbers and some are integers, i.e., "mixed integer"
- It is almost like a linear programming, except that some variables are integers

NP-C problem

Group Assignment Problem

- You are going to assign 4 persons into 2 groups
  - Everyone associates with only one group
  - Every group has at least one person
  - Dislike persons are not in the same group
  - Like overcomes dislike
  - Try your best to balance the group
- Let's model this problem as an ILP/SAT problem!

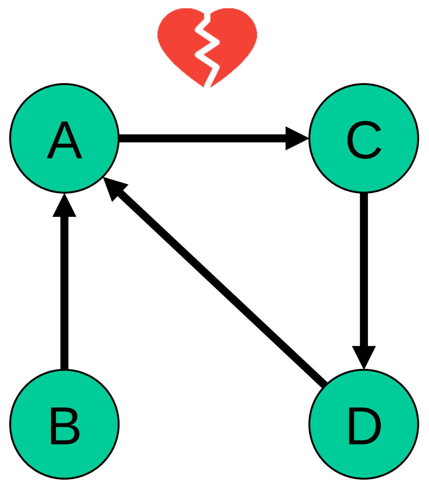
Example of Group Assignment

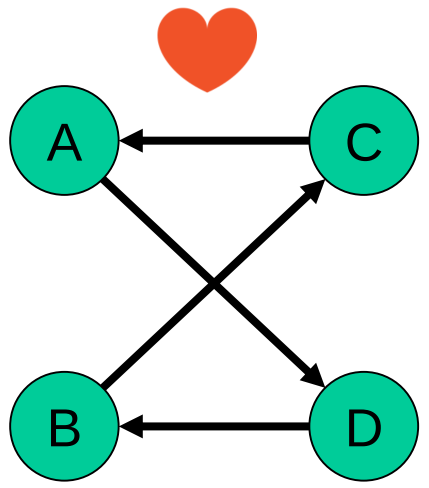
$$\begin{tikzcd}
A \arrow[r] & C \arrow[d] \\
B \arrow[u] & D \arrow[ul]
\end{tikzcd}$$

$$\begin{tikzcd}
A \arrow[dr] & C \arrow[l] \\
B \arrow[ur] & D \arrow[l]
\end{tikzcd}$$

ILP-based Formulation – 1

- Variable *a1* and *a2* are binary (0/1)
  - *a1* = 1: *A* in group 1
  - *a2* = 1: *A* in group 2
- Constraints:
  - Uniqueness constraint
  - Group constraint
  - Dislike constraint
- Objective:
  - Balanced group size

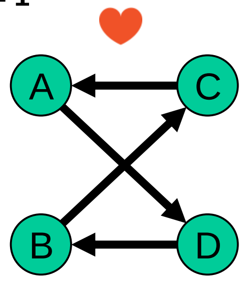


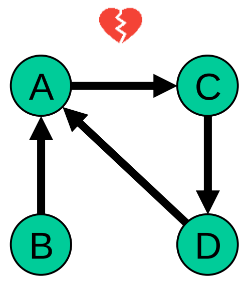


ILP-based Formulation – 2

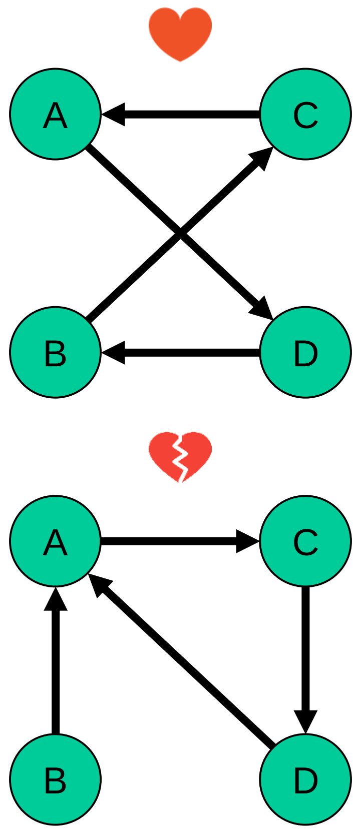
- ❑ Variable *a1* and *a2* are binary (0/1)
  - – *a1* = 1: *A* in group 1
  - – *a2* = 1: *A* in group 2
- ❑ Constraints:
  - – Uniqueness constraint
  - – Group constraint
  - – Dislike constraint
- ❑ Objective:
  - – Balanced group size

```
Min
a1 + b1 + c1 + d1 - a2 - b2 - c2 - d2
Subject To
a1 + b1 + c1 + d1 - a2 - b2 - c2 - d2 >= 0
a1 + a2 = 1
b1 + b2 = 1
c1 + c2 = 1
d1 + d2 = 1
a1 + b1 + c1 + d1 >= 1
a2 + b2 + c2 + d2 >= 1
b1 + a1 - c1 <= 1
b2 + a2 - c2 <= 1
a1 + c1 - d1 <= 1
a2 + c2 - d2 <= 1
c1 + d1 - a1 <= 1
c2 + d2 - a2 <= 1
d1 + a1 - b1 <= 1
d2 + a2 - b2 <= 1
Binary
a1
b1
c1
d1
a2
b2
c2
d2
End
```

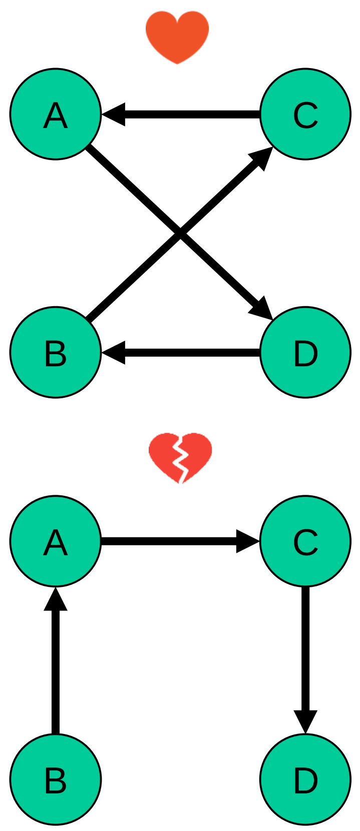




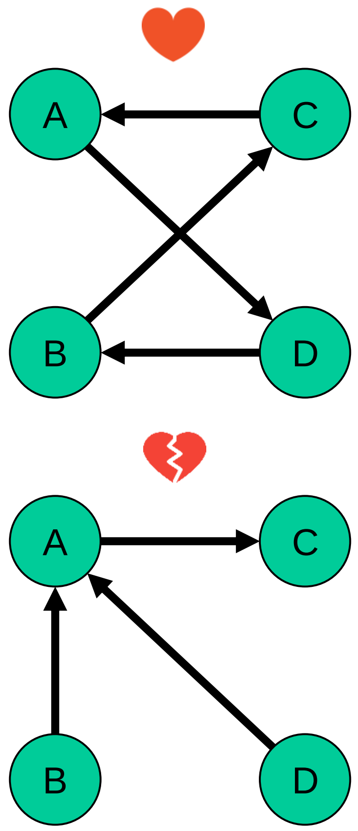
ILP-based Formulation – 3



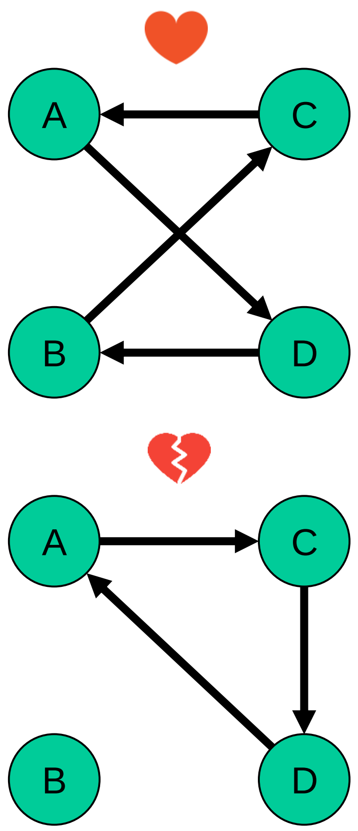
No solution



{A, D} {B, C}



{B, C, D} {A}



{A, B, D} {C}

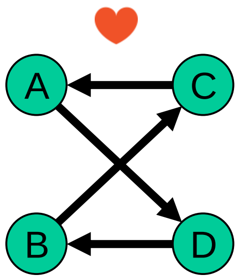
SAT-based Formulation

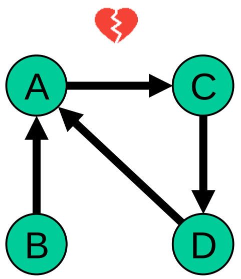
- Given 4 persons and 2 groups
  - *a1* = T: *A* in group 1
  - *a2* = T: *A* in group 2
- Constraints:
  - Uniqueness constraint
  - Group constraint
  - Dislike constraint
- Objective:
  - Any feasible solution is OK.

$a_1 \rightarrow 1$
$b_1 \rightarrow 2$
$c_1 \rightarrow 3$
$d_1 \rightarrow 4$

$a_2 \rightarrow 5$
$b_2 \rightarrow 6$
$c_2 \rightarrow 7$
$d_2 \rightarrow 8$

```
p cnf 8 18
c uniqueness constraint
1 5 0
-1 -5 0
2 6 0
-2 -6 0
3 7 0
-3 -7 0
4 8 0
-4 -8 0
c group constraint
1 2 3 4 0
5 6 7 8 0
c dislike constraint
-2 -1 3 0
-6 -5 7 0
-1 -3 4 0
-5 -7 8 0
-3 -4 1 0
-7 -8 5 0
-4 -1 2 0
-8 -5 6 0
```





Circuit Models

- Circuit model
  - Abstraction
  - Representation of relevant features only

- Unambiguously transferring the design information
  - Humans to humans
  - Humans to CAD tools

Model Classifications

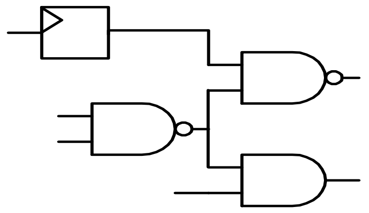
- Abstraction levels
  - Architectural
  - Logic
  - Geometrical
- Views
  - Behavioral
  - Structural
  - Physical
- Media
  - Language
  - Diagram
  - Mathematical model

Abstraction Levels

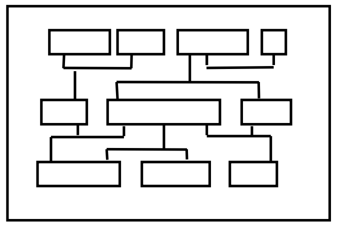
Architectual level

...
PC = PC + 1;
FETCH(PC);
DECODE(INST);
...

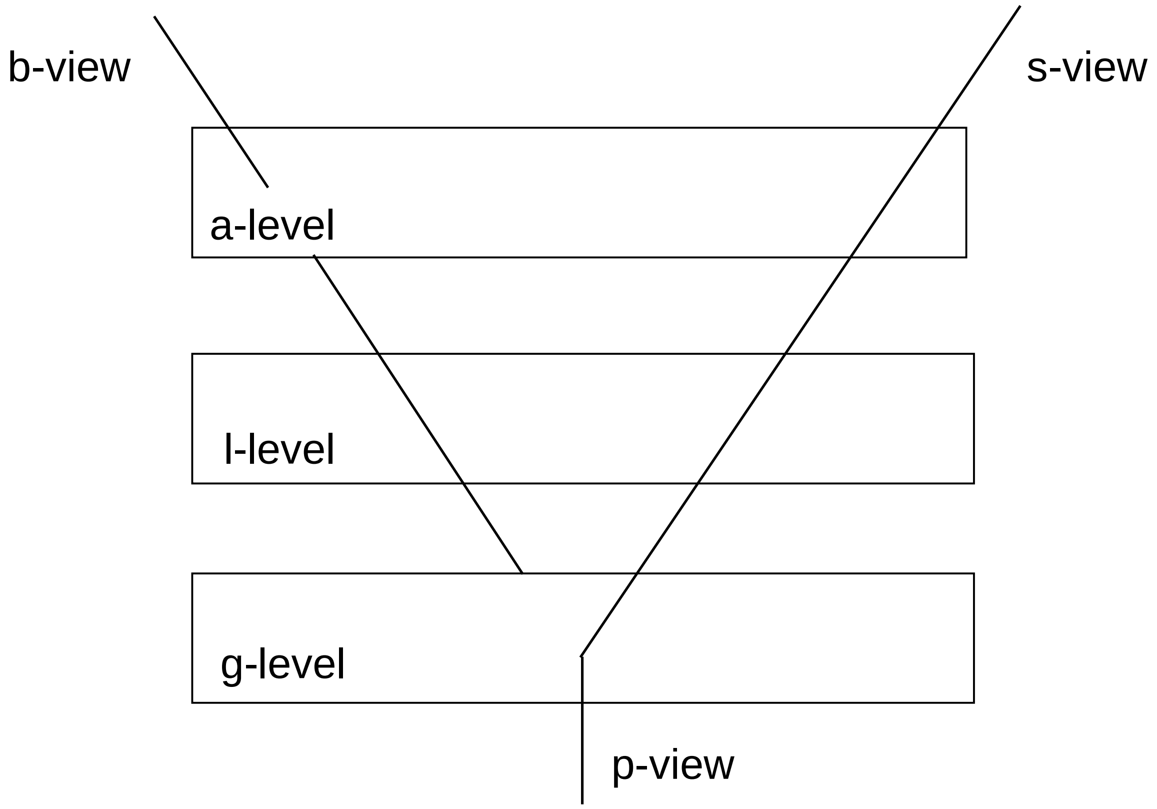
Logic level



Geometrical level

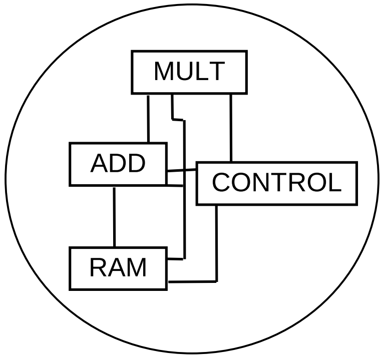


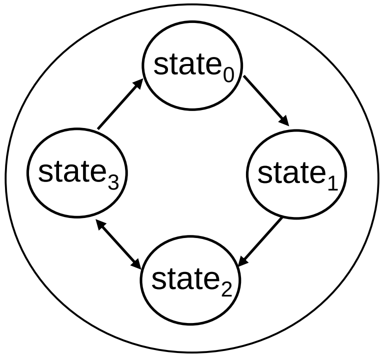
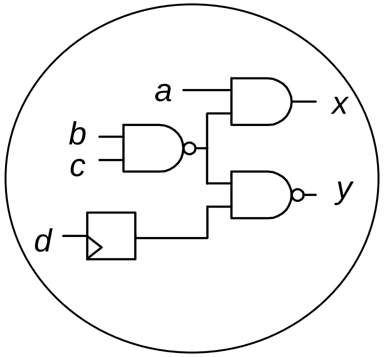
Views and Abstraction Levels



Levels of Abstractions and Corresponding Views

Behavioral View	Structural View	Views / Levels

...
PC = PC + 1;
FETCH(PC);
DECODE(INST);
...
		Architectural Level

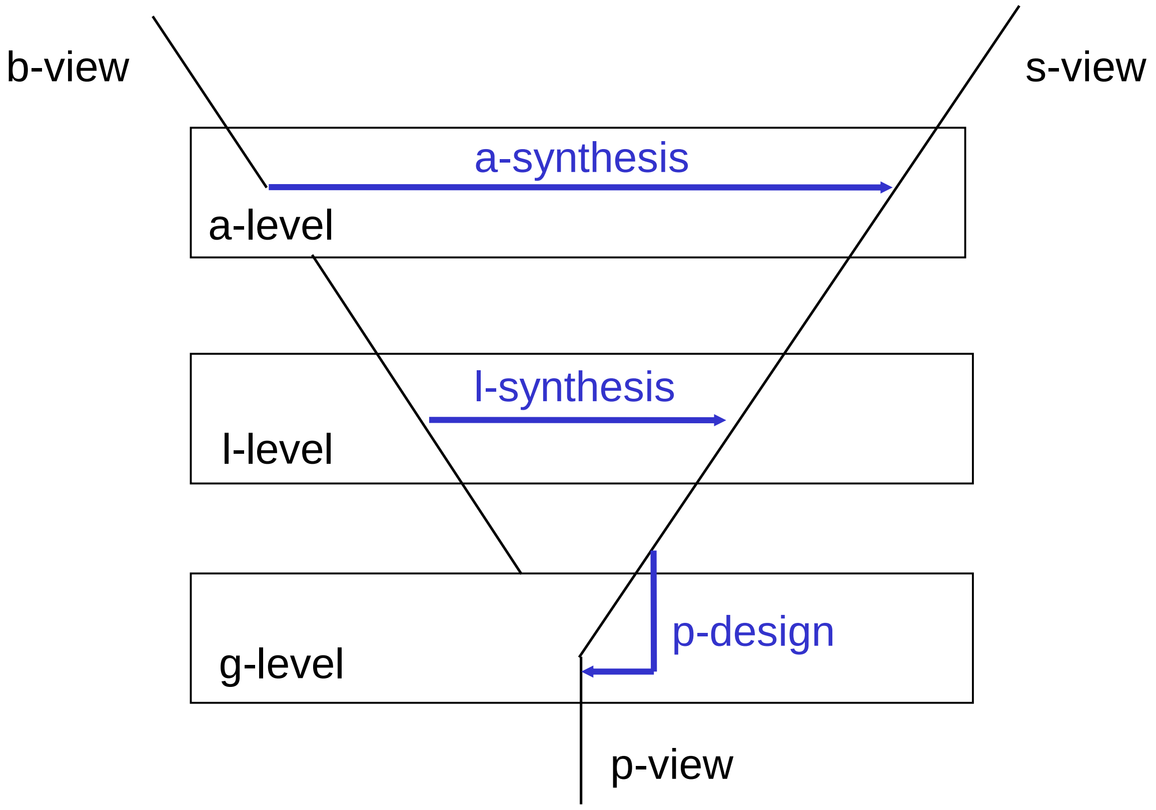
 state_0 state_1 state_2 state_3		Logic Level

Synthesis

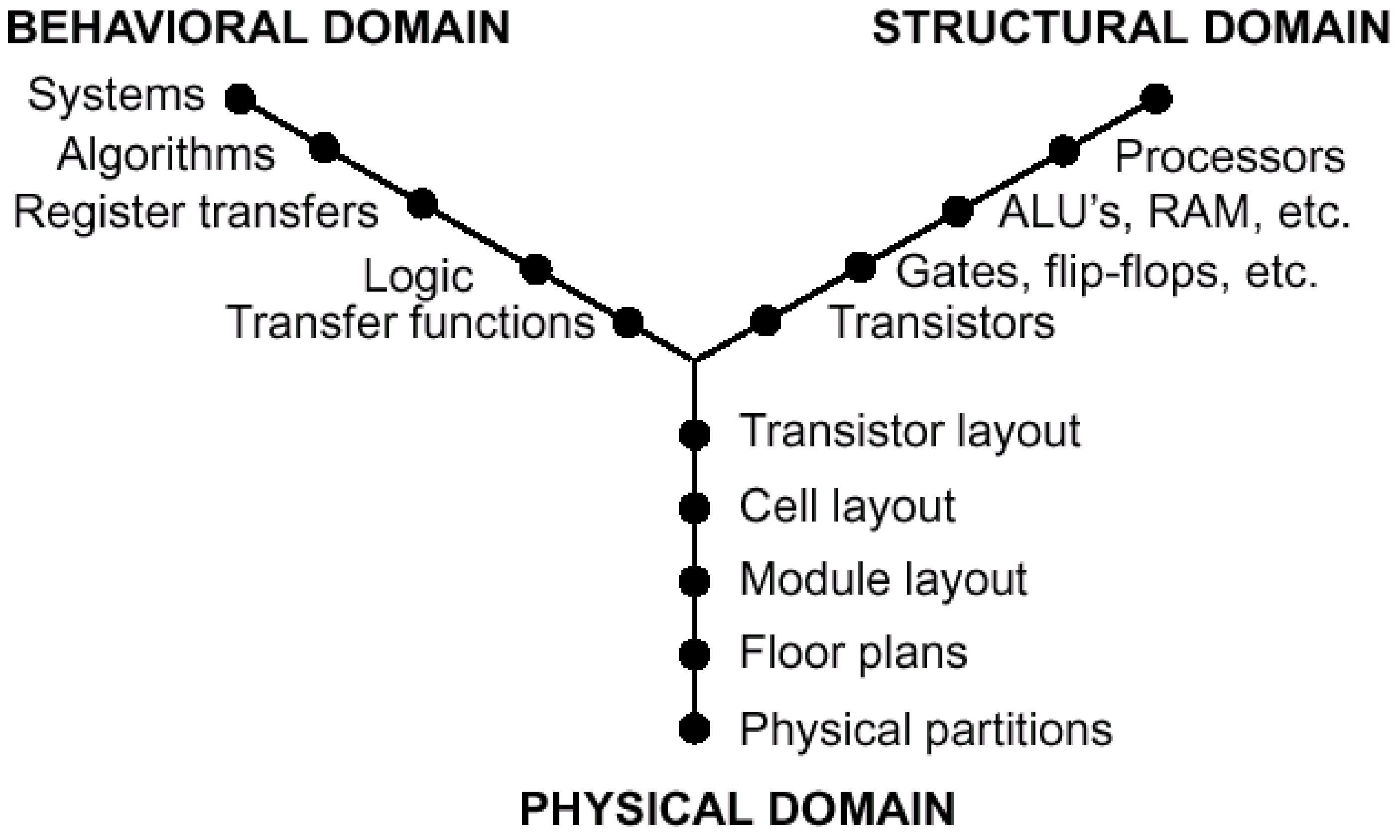
- Synthesis
  - A set of transformation between two views

- Synthesis tasks
  - Architectural-level synthesis
    - High-level synthesis, structural synthesis
  - Logic-level synthesis
    - Gate-level structure, library binding (mapping)
  - Geometrical-level synthesis
    - Physical design

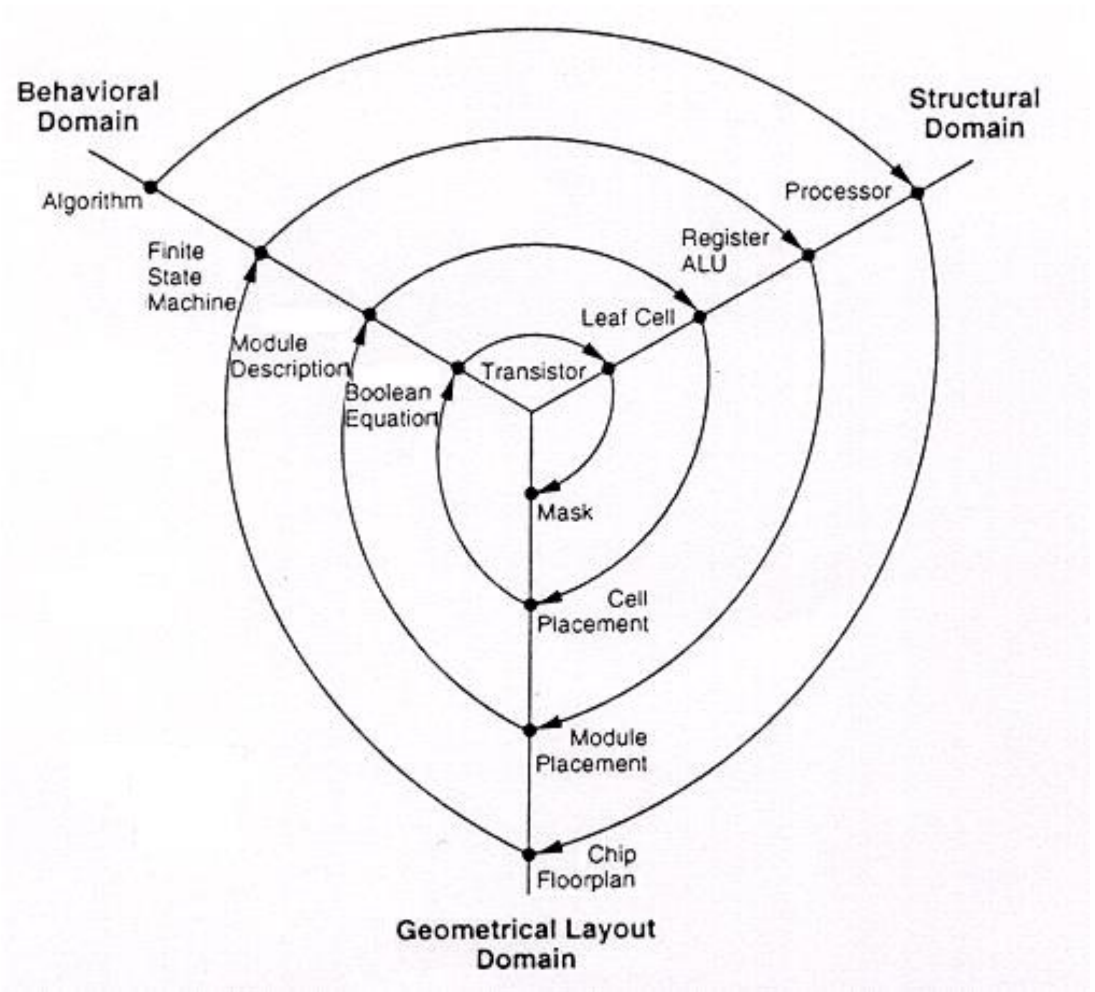
Synthesis Tasks



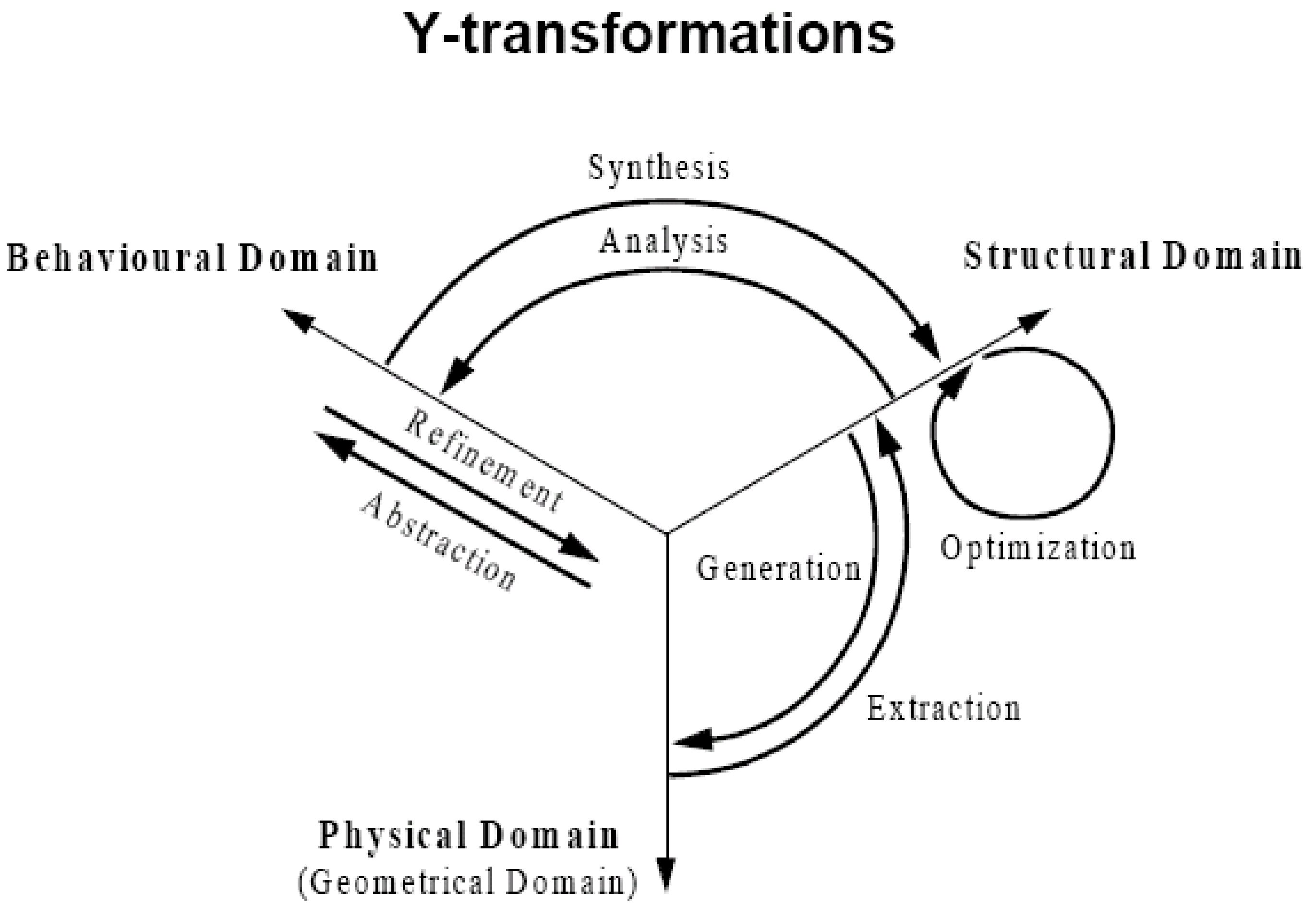
Gajski's Y-chart



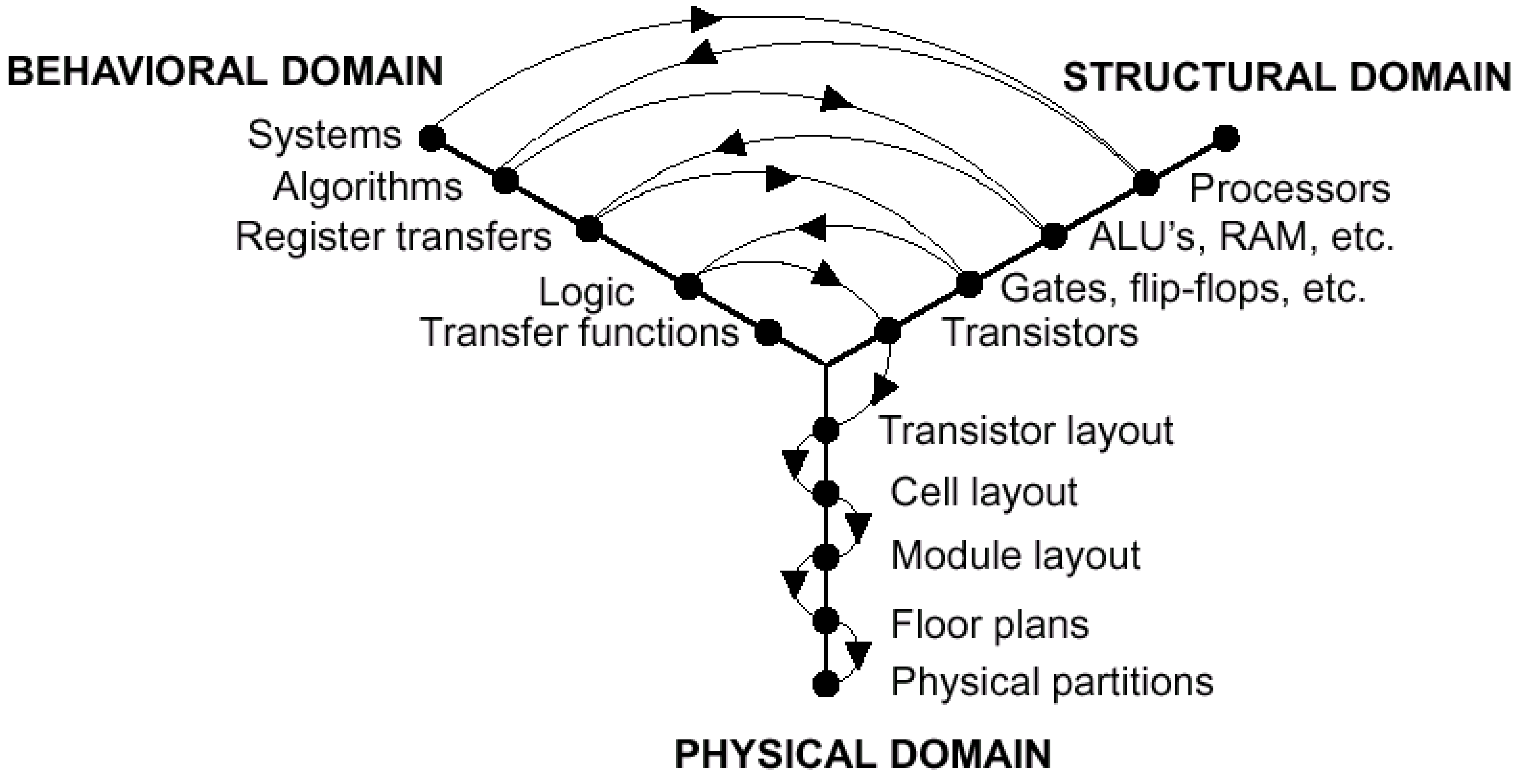
Y-chart Domain Mapping



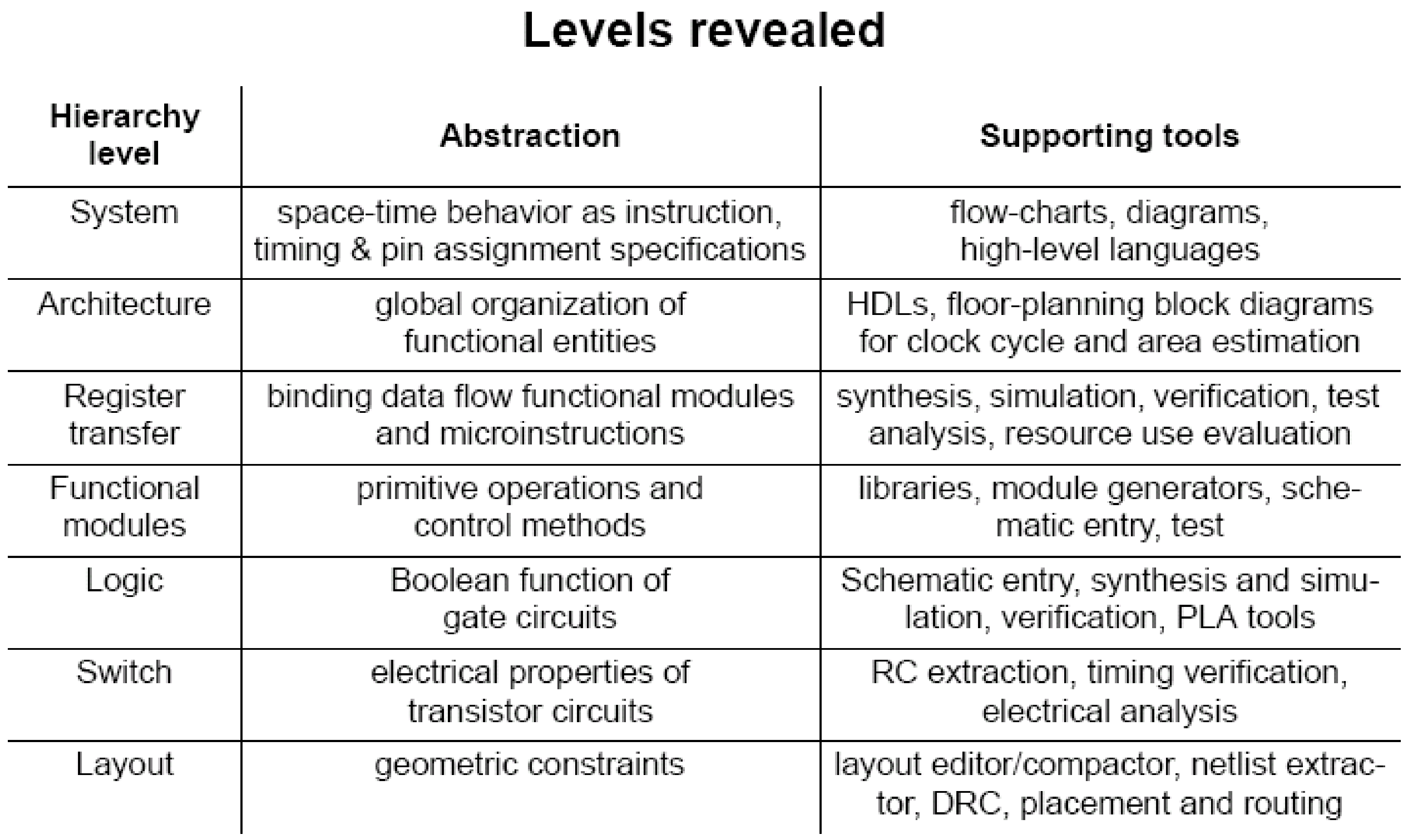
Y-transformations



Top-down Design



Levels Revealed



Levels revealed

Hierarchy level	Abstraction	Supporting tools
System	space-time behavior as instruction, timing & pin assignment specifications	flow-charts, diagrams, high-level languages
Architecture	global organization of functional entities	HDLs, floor-planning block diagrams for clock cycle and area estimation
Register transfer	binding data flow functional modules and microinstructions	synthesis, simulation, verification, test analysis, resource use evaluation
Functional modules	primitive operations and control methods	libraries, module generators, sche-matic entry, test
Logic	Boolean function of gate circuits	Schematic entry, synthesis and simu-lation, verification, PLA tools
Switch	electrical properties of transistor circuits	RC extraction, timing verification, electrical analysis
Layout	geometric constraints	layout editor/compactor, netlist extrac-tor, DRC, placement and routing

Synthesis Levels and Tasks

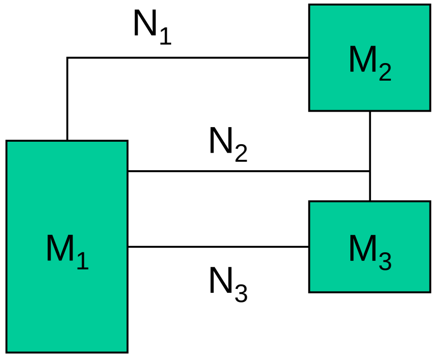
- System level synthesis
  - Clustering
  - Communication synthesis
- High-level synthesis
  - Resource or timing constrained scheduling
  - Resource allocation
  - Binding
- Register transfer level synthesis
  - Data-path synthesis
  - Controller synthesis
  - Logic-level synthesis
  - Logic minimization
  - Optimization, overhead removal
  - Library mapping
- Physical level synthesis
  - Placement
  - Routing

Abstract Models

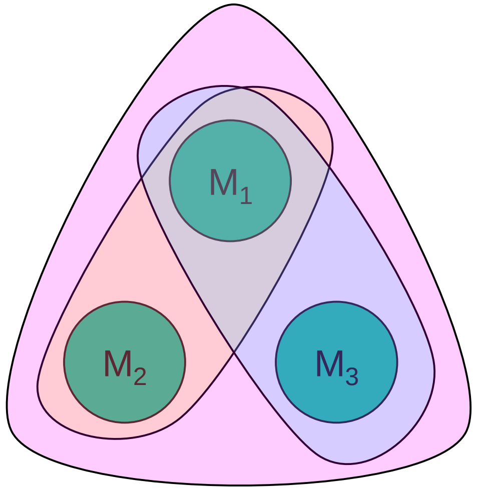
- Abstract models for different circuit views
  - Structure
  - Logic network
    - Combinational logic network
    - Sequential logic network
  - State diagram
  - Data-flow and Sequencing Graph

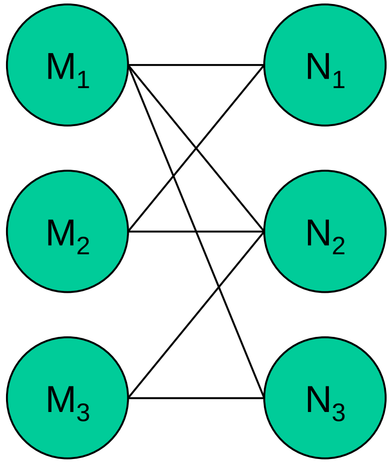
Abstract Models – Structure

- Module, net, pin
- Incidence matrix
- Hypergraph
- Bipartite graph



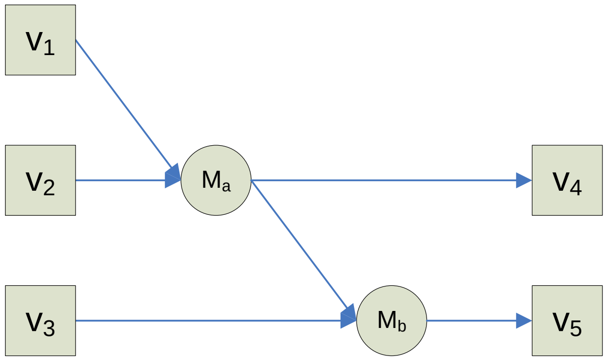
$$\begin{array}{c|ccc} & N_1 & N_2 & N_3 \\ \hline M_1 & 1 & 1 & 1 \\ M_2 & 1 & 1 & 0 \\ M_3 & 0 & 1 & 1 \end{array}$$



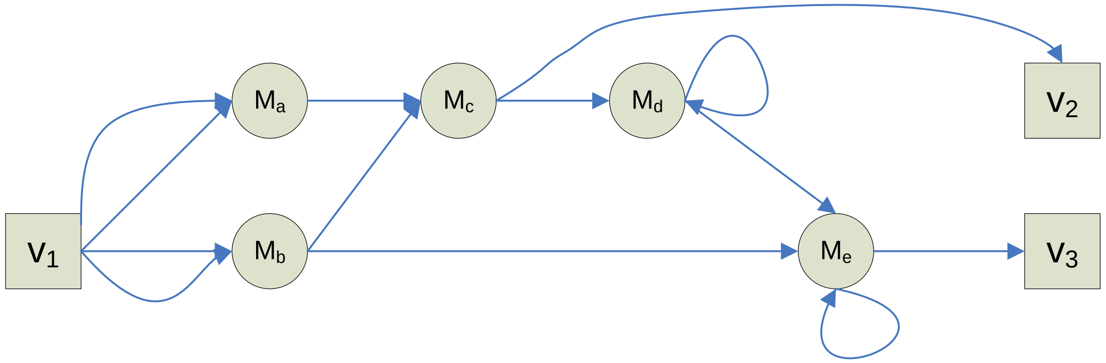


Abstract Models – Logic Network

- Combinational logic network



- Sequential logic network



Abstract Models – State Diagram

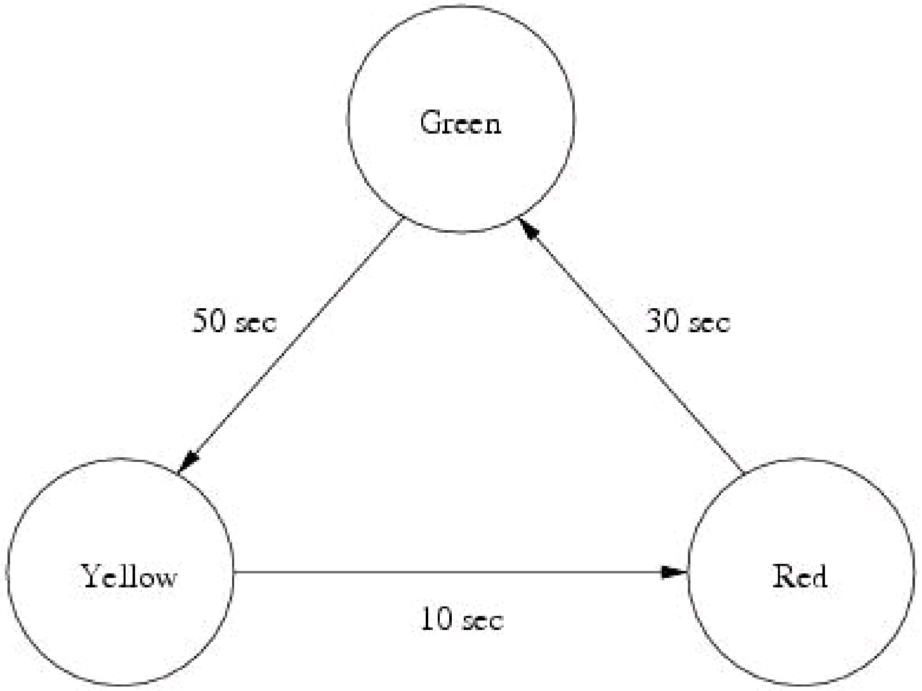
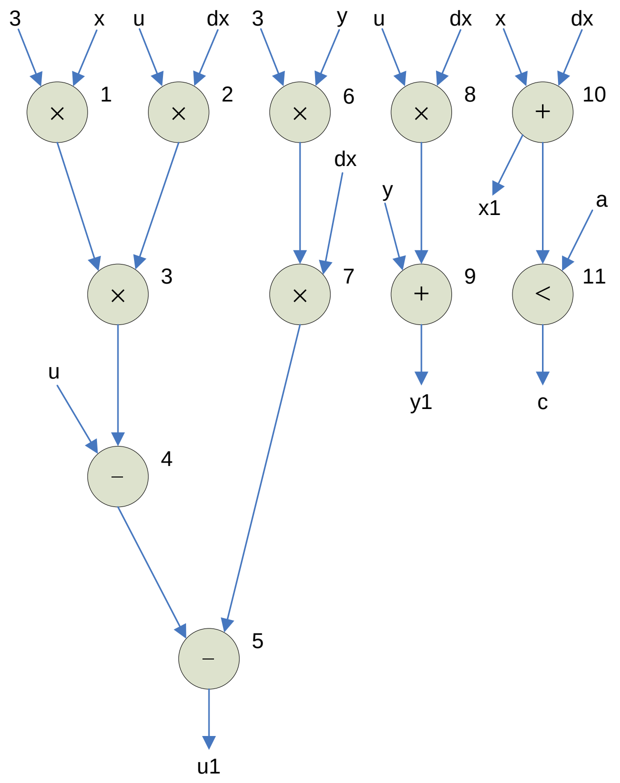
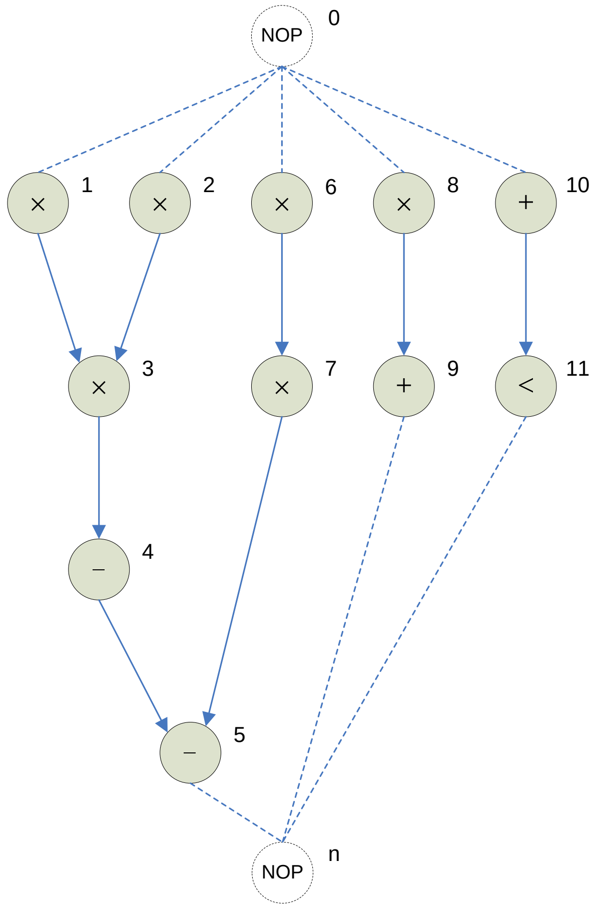


Figure 1: traffic light

Abstract Models – Data-flow and Sequencing Graph



Data-flow Graph



Sequencing Graph

Hardware Description Language

- ❑ Recent trend for circuit specification
  - – Hardware description language (HDL)
    - • Verilog, VHDL
  - – Similar to write a software program

- ❑ Concise HDL models are preferable for representing
  - – Flow
  - – State
  - – Logic diagram

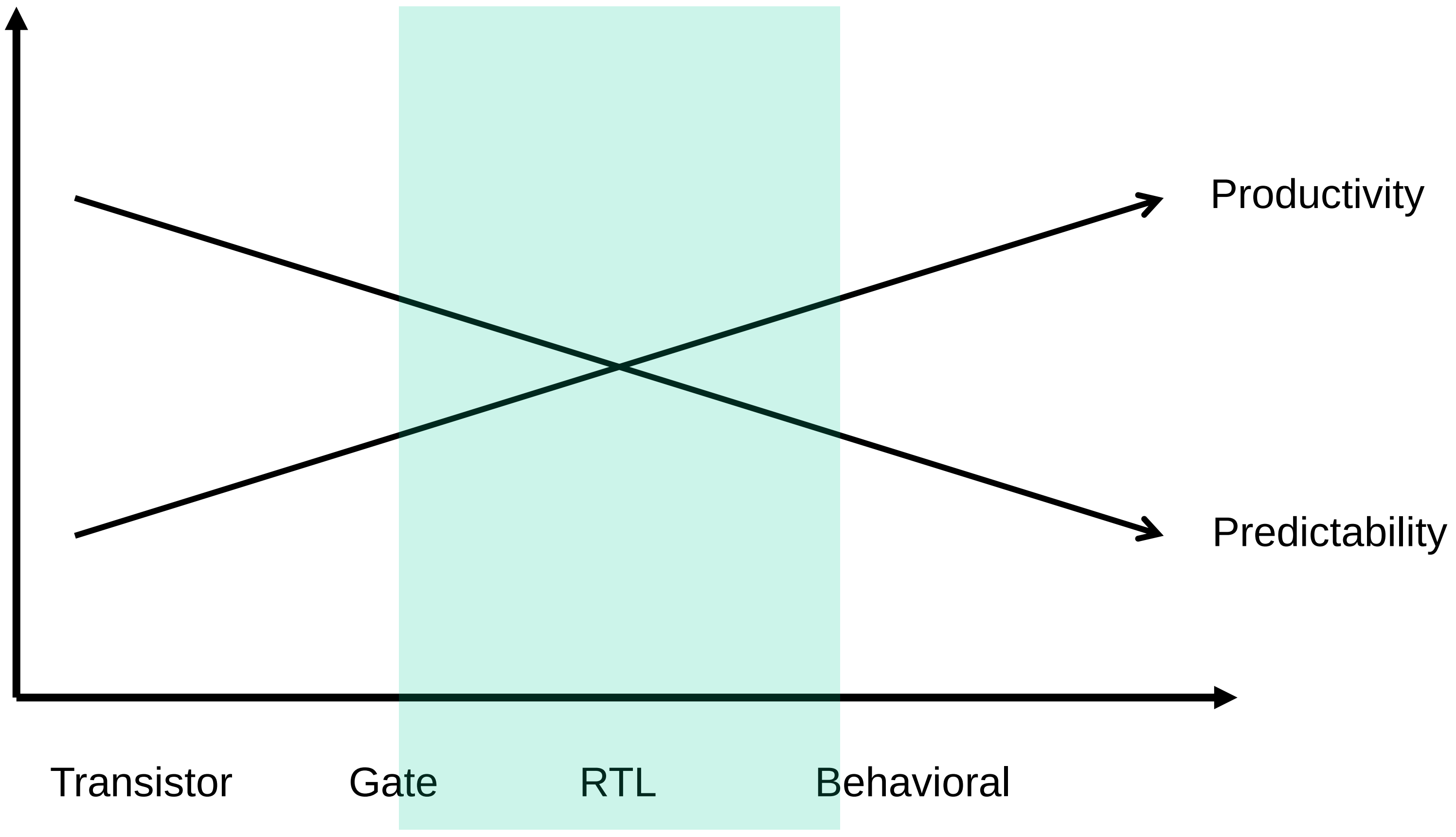
Hardware Description Language

- Structural description
  - Textual replacement for schematic
  - Hierarchical composition of modules from primitives
- Behavioral/functional description
  - Describe what module does, not how
  - Synthesis generates circuit for module

Miscellaneous HDLs

- Abel (circa 1983) - developed by Data-I/O
  - Targeted to programmable logic devices
  - Not good for much more than state machines
- ISP (circa 1977) - research project at CMU
  - Simulation, but no synthesis
- Verilog (circa 1985) - developed by Gateway (absorbed by Cadence)
  - Similar to Pascal and C
  - Delays is only interaction with simulator
  - Fairly efficient and easy to write
  - IEEE standard
- VHDL (circa 1987) - DoD sponsored standard
  - Similar to Ada (emphasis on re-use and maintainability)
  - Simulation semantics visible
  - Very general but verbose
  - IEEE standard

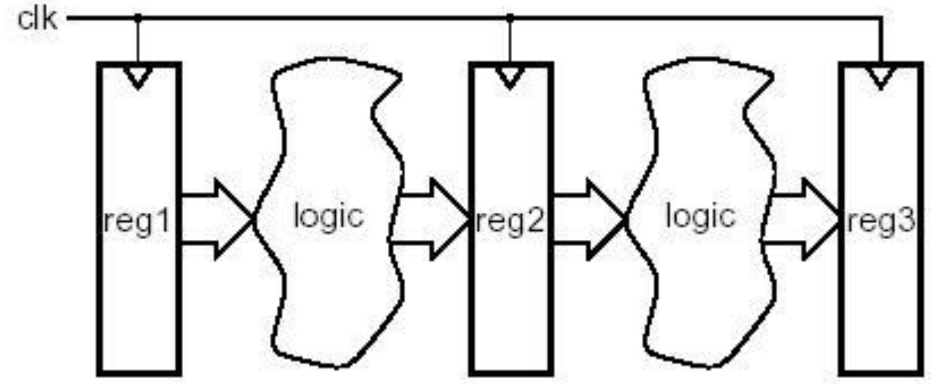
Productivity vs. Predictability



∴ RTL is commonly used for circuit design

Register Transfer Level (RTL)

- A digital system is specified at the register transfer level (RTL) when it is specified by:
  - The set of registers
  - The operations performed on the data stored
  - The control that supervises the sequence of operations



Verilog

- Supports structural and behavioral descriptions
- Structural description
  - Explicit structure of the circuit
  - Ex: each logic gate instantiated and connected to others
- Behavioral description
  - Program describes input/output behavior
  - Many structural implementations could have same behavior
  - Ex: different implementation of one Boolean function (full-adder)

Verilog Descriptions

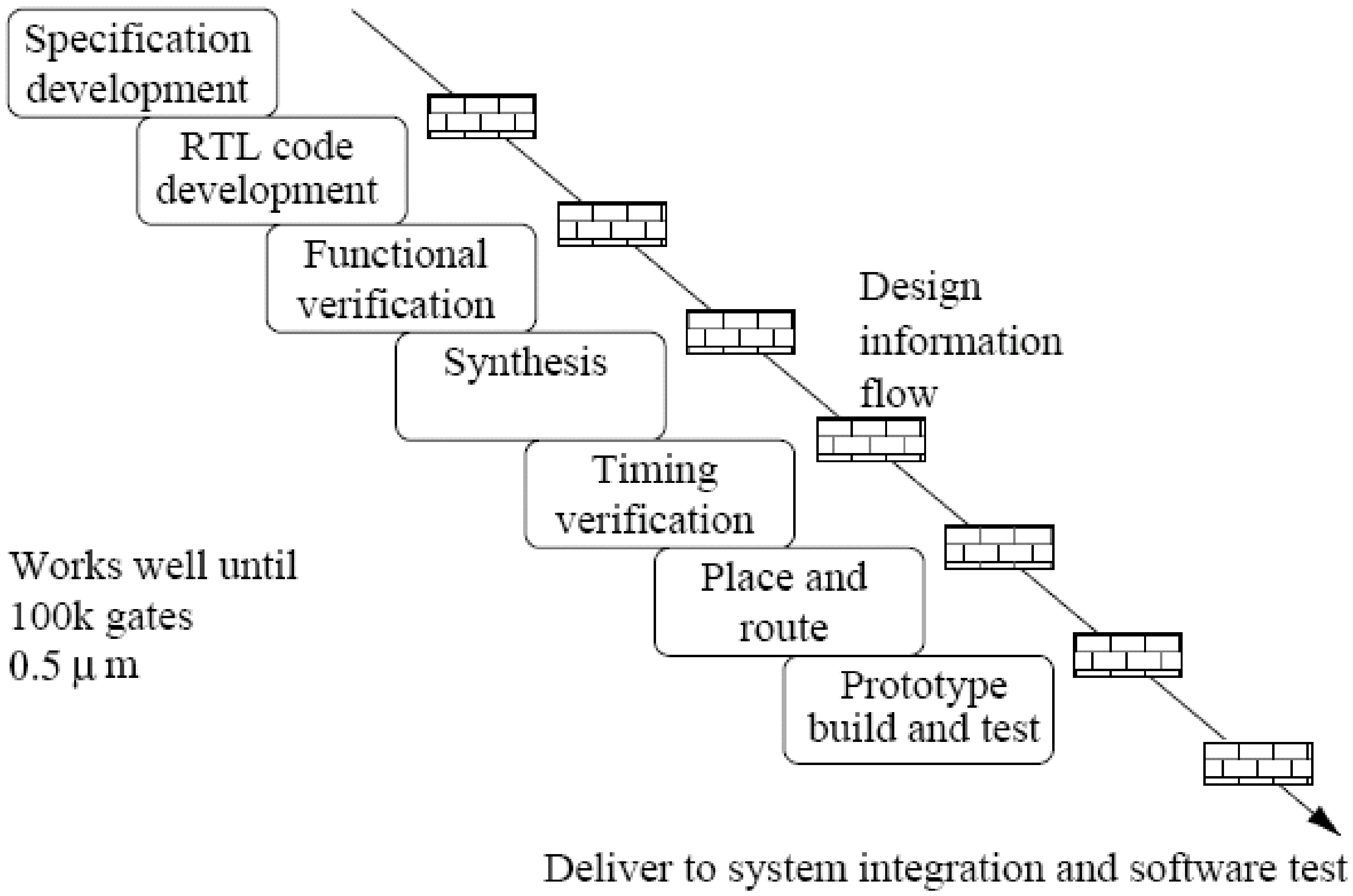
- ❑ Structural description

```
module HALF_ADDER (a, b, carry, sum);
   input   a, b;
   output   carry, sum;

   and
      G1 (carry, a, b);
   xor
      G2 (sum, a, b);

endmodule
```

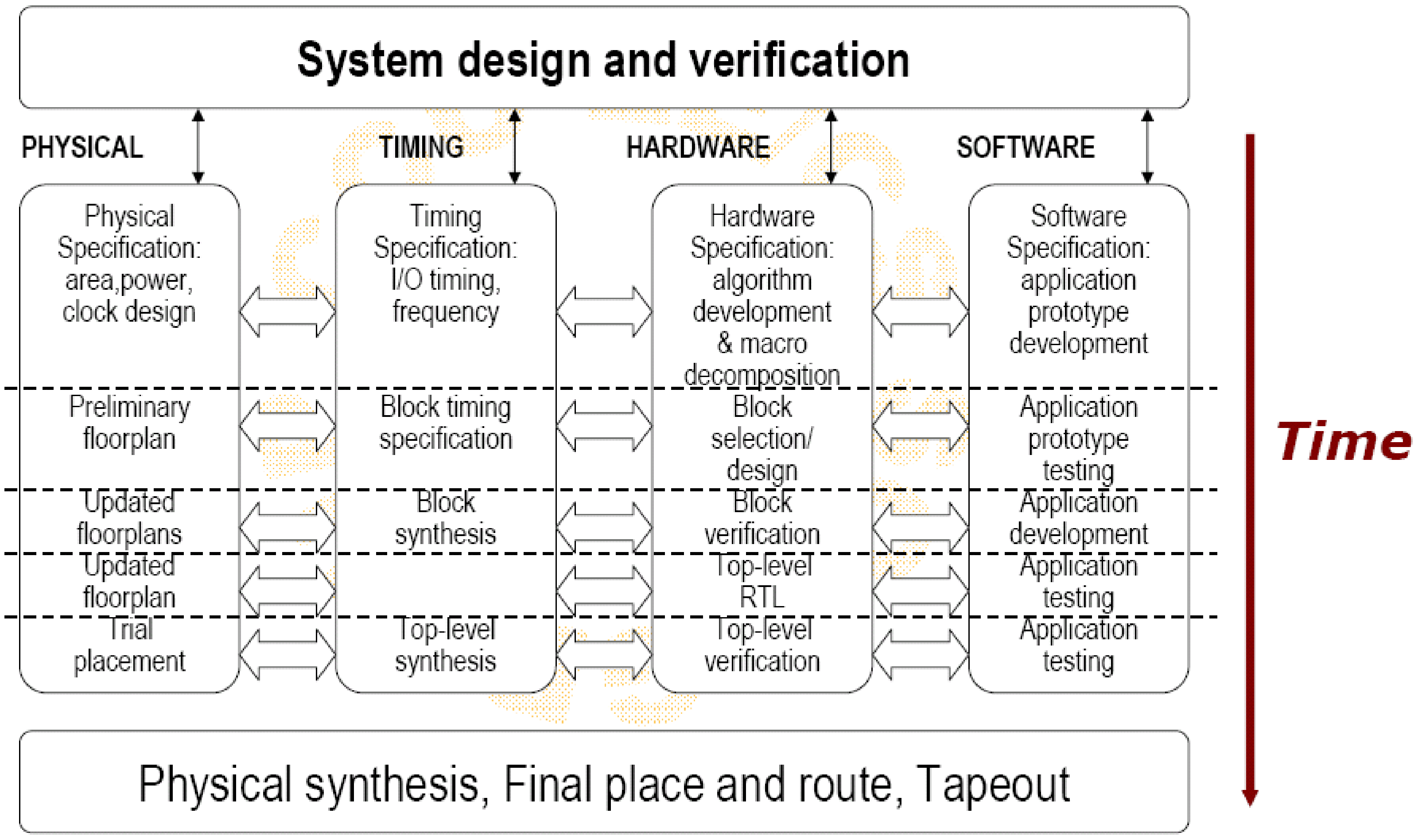
- ❑ Behavioral description

```
module HALF_ADDER (a, b, carry, sum);
   input   a, b;
   output   carry, sum;

   assign carry = a & b;
   assign sum = a ^ b;

endmodule
```

Traditional Waterfall Model



Spiral SOC Design Flow


System design and verification
PHYSICAL
TIMING
HARDWARE
SOFTWARE
Physical Specification: area,power, clock design
Timing Specification: I/O timing, frequency
Hardware Specification: algorithm development & macro decomposition
Software Specification: application prototype development
Preliminary floorplan
Block timing specification
Block selection/ design
Application prototype testing
Updated floorplans
Block synthesis
Block verification
Application development
Updated floorplan
Top-level RTL
Application testing
Trial placement
Top-level synthesis
Top-level verification
Application testing
Physical synthesis, Final place and route, Tapeout
Time

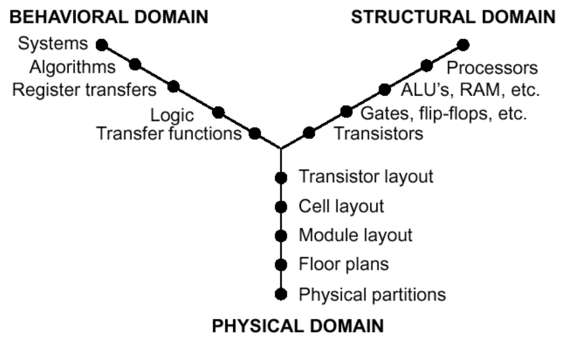
Spiral SOC Design Flow

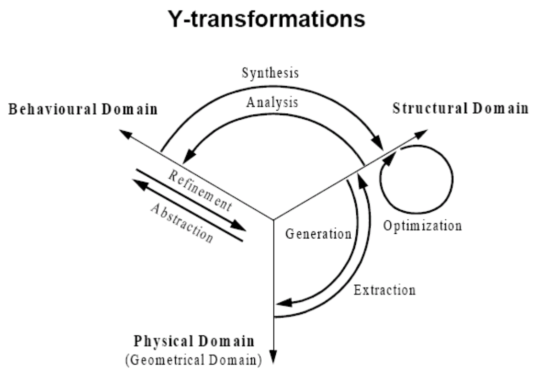
- Characteristics
  - Parallel, concurrent development of hardware and software
  - Parallel verification and synthesis of modules
  - Floorplan, placement, and routing are included in the synthesis process
  - Modules developed only if a pre-designed hard or soft macro is not available – reusability
  - Planned iteration throughout

- The engineer are addressing all aspects of hardware and software design concurrently: functionality, timing, physical design, and verification

Waterfall vs. Spiral

- Traditional ASIC development follows so called waterfall model
  - Project transitions from phase to phase in step
  - Never returning to the activities of the previous phase
  - “Tossing” project over the wall from one team to the next
- However…
  - Complexity increases
  - Geometry shrinks
  - Time-to-market pressure increases
- In the spiral model, the design teams work on multiple aspects of the design simultaneously, incrementally improving in each area as the design converges on completion





Levels revealed

Hierarchy level	Abstraction	Supporting tools
System	space-time behavior as instruction, timing & pin assignment specifications	flow-charts, diagrams, high-level languages
Architecture	global organization of functional entities	HDLs, floor-planning block diagrams for clock cycle and area estimation
Register transfer	binding data flow functional modules and microinstructions	synthesis, simulation, verification, test analysis, resource use evaluation
Functional modules	primitive operations and control methods	libraries, module generators, sche-matic entry, test
Logic	Boolean function of gate circuits	Schematic entry, synthesis and simu-lation, verification, PLA tools
Switch	electrical properties of transistor circuits	RC extraction, timing verification, electrical analysis
Layout	geometric constraints	layout editor/compactor, netlist extrac-tor, DRC, placement and routing