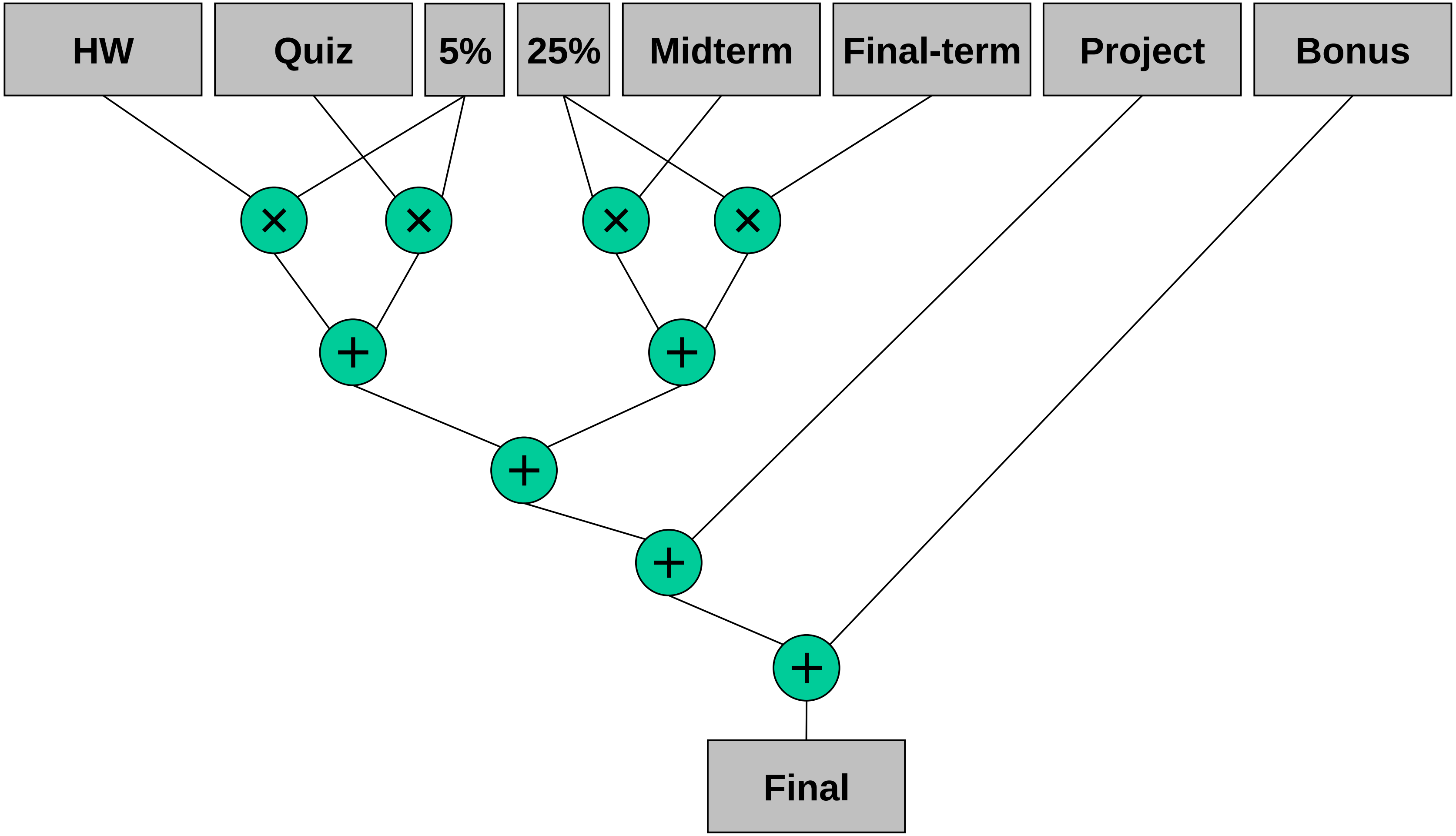
CAD for VLSI

Introduction

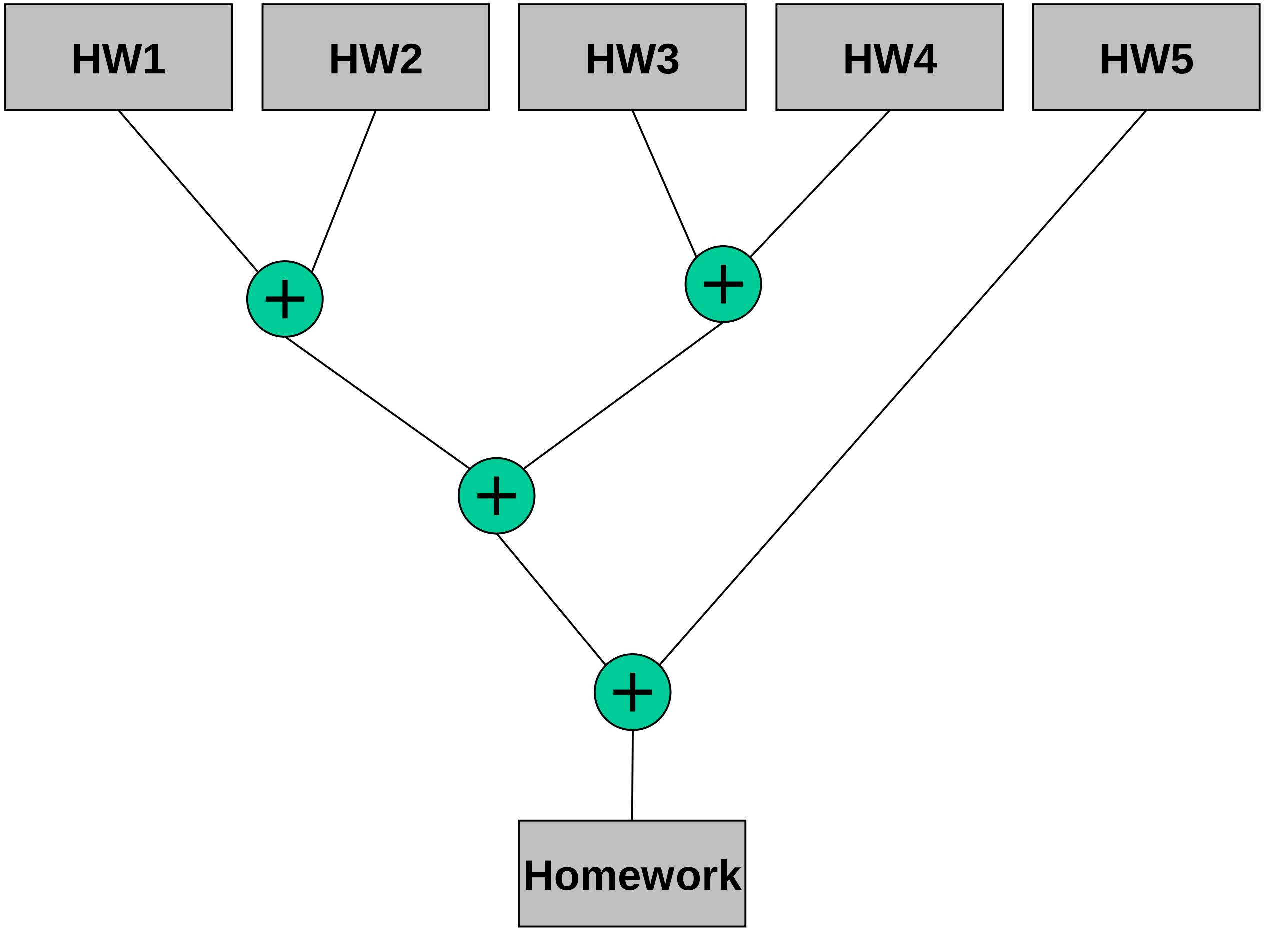
Let’s Design a Grading System

- ❑ 5 homework assignments (25%)
- ❑ 6 in-class quizzes (ignore lowest score) (25%)
- ❑ 1 midterm examination (25%)
- ❑ 1 final-term examination (25%)
- ❑ 1 extra term project
- ❑ Some bonus – class participation

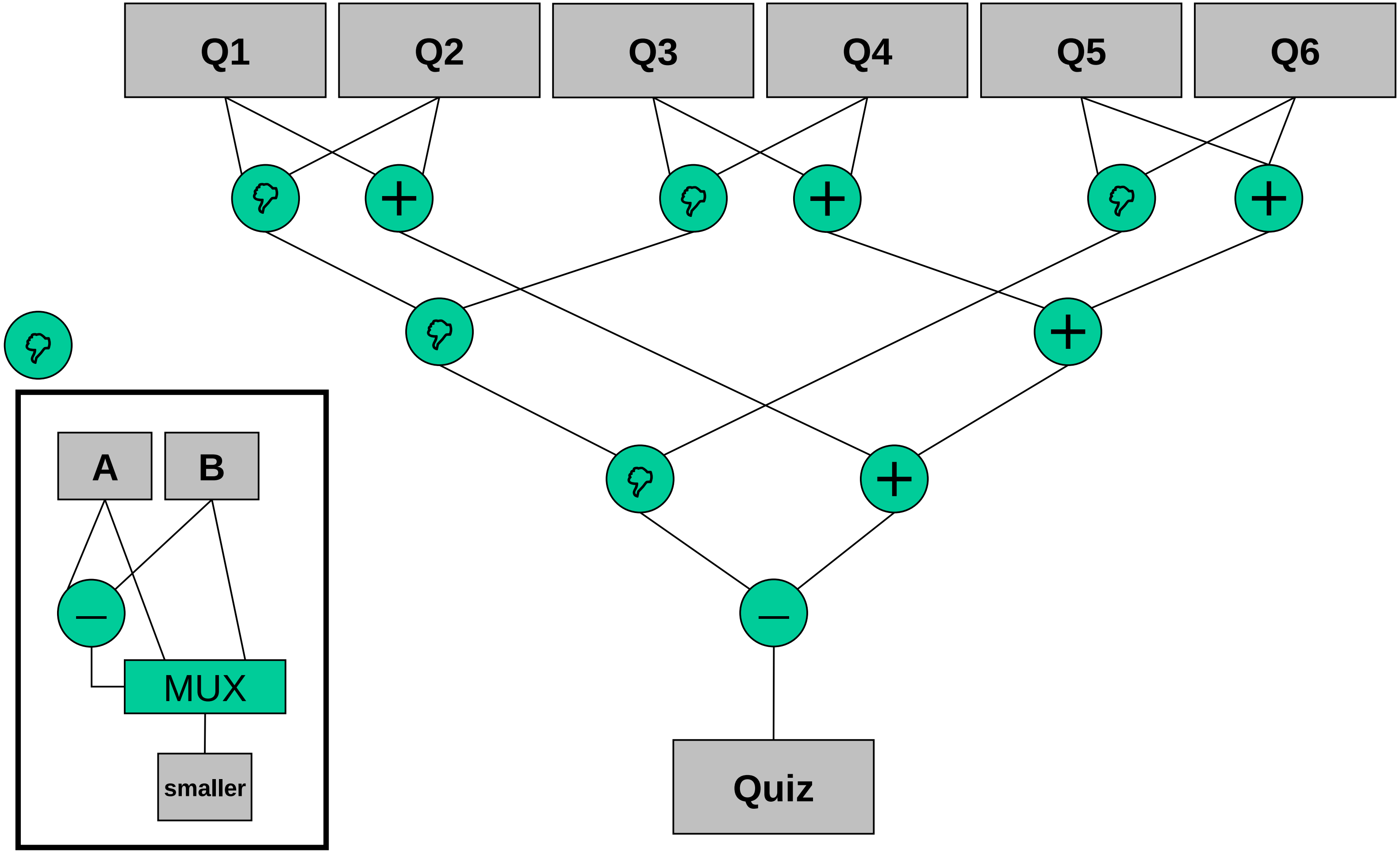
Overall Data Flow Graph (DFG)



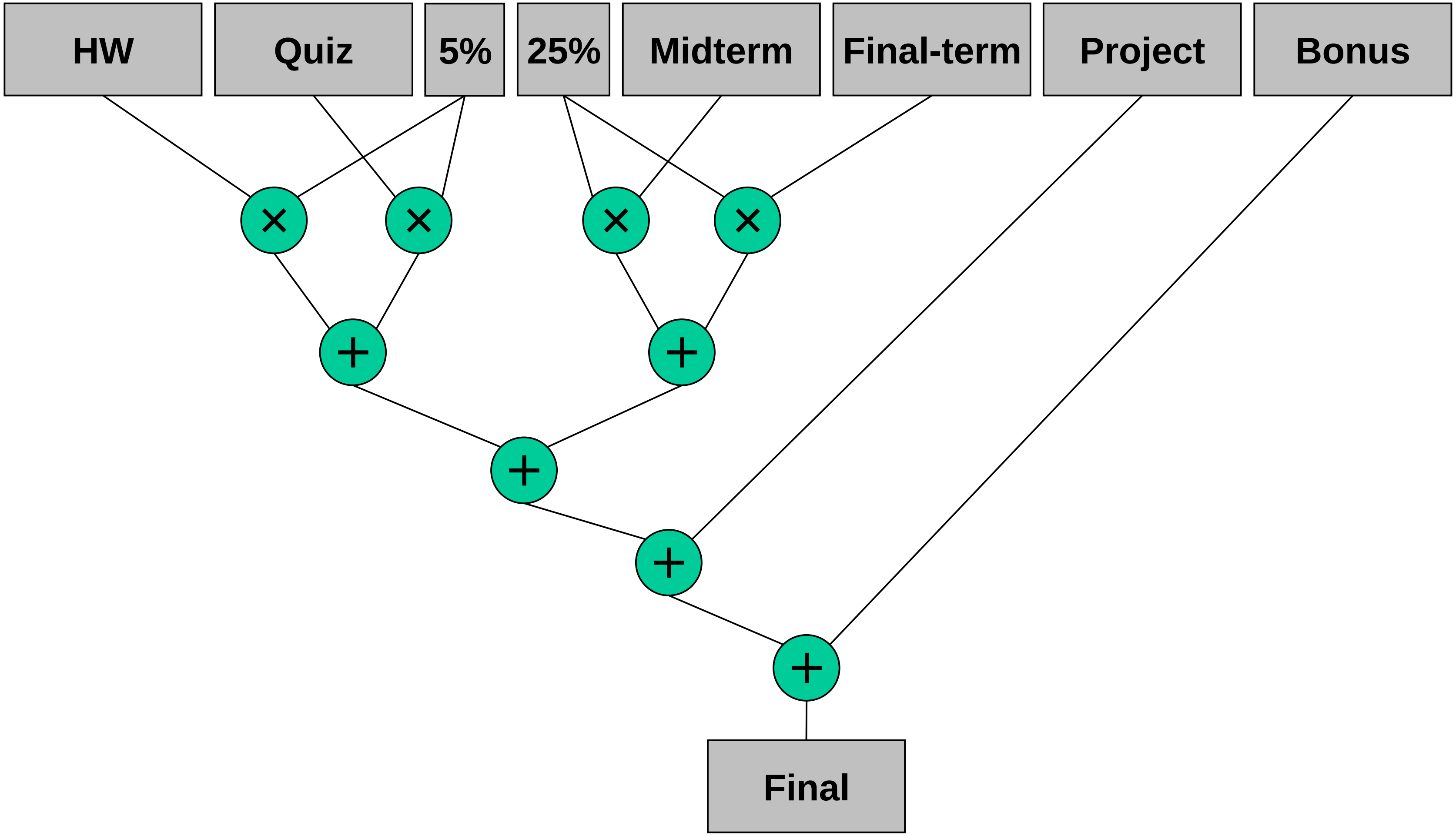
Homework DFG



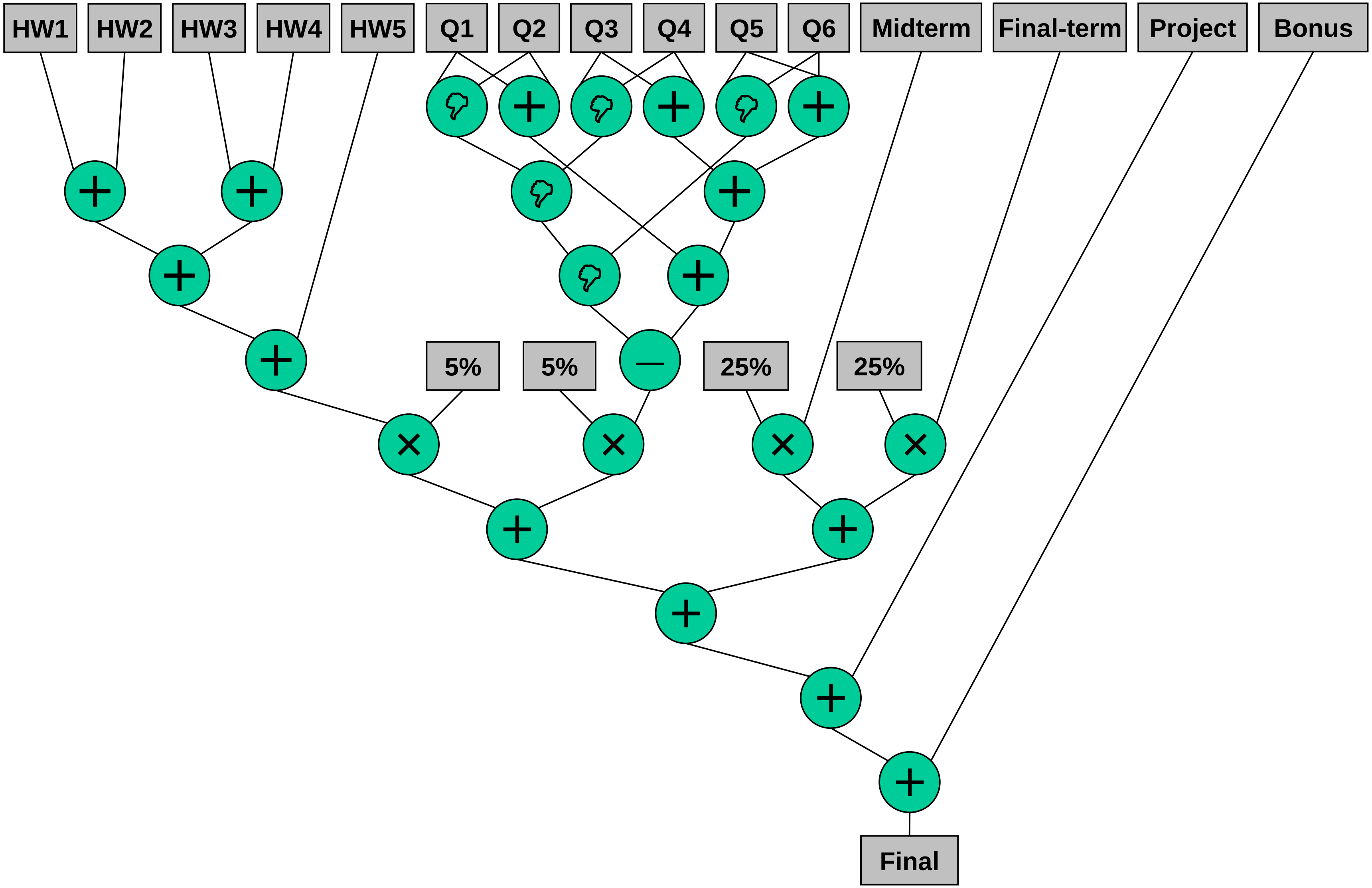
Quiz DFG



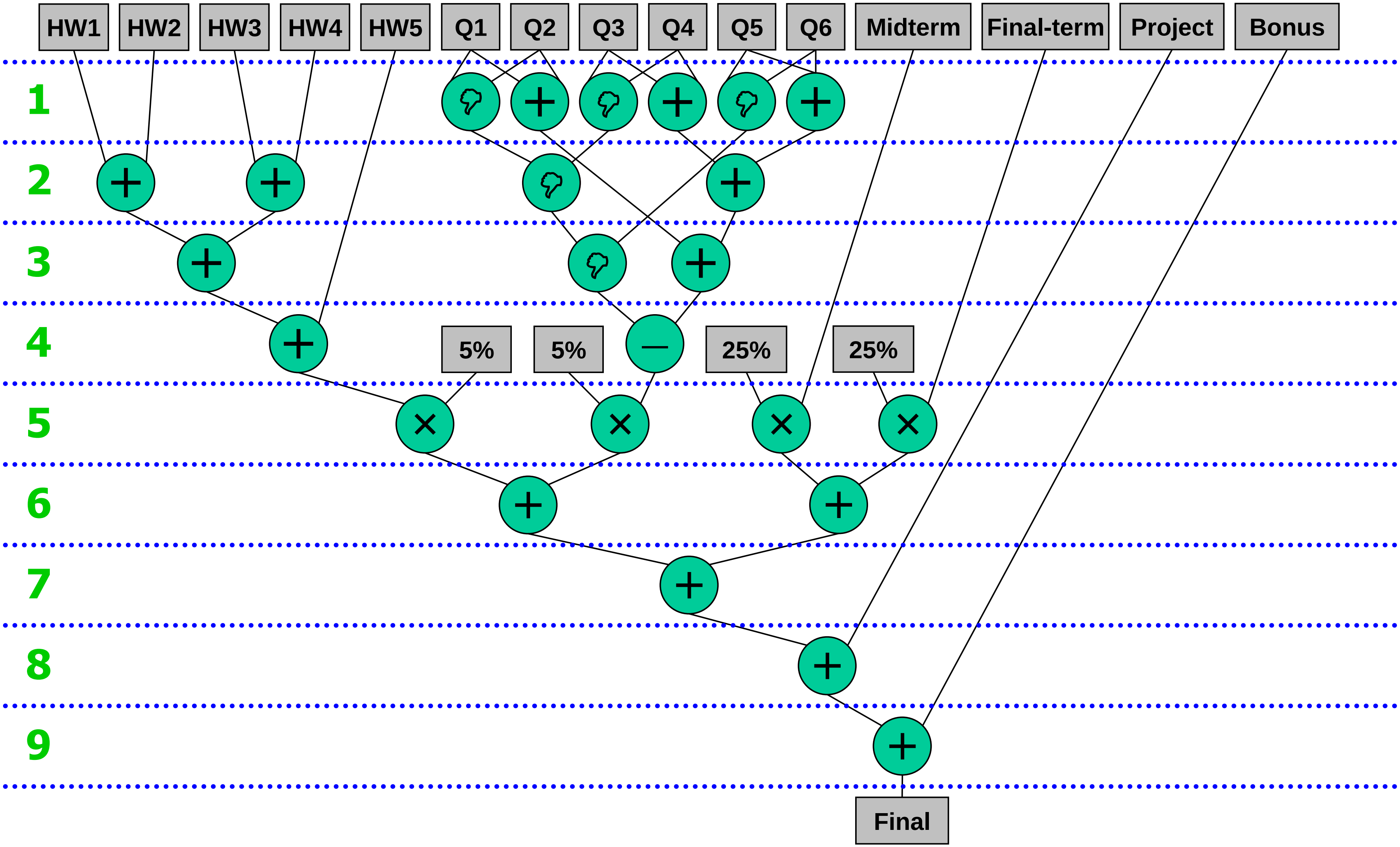
Overall DFG



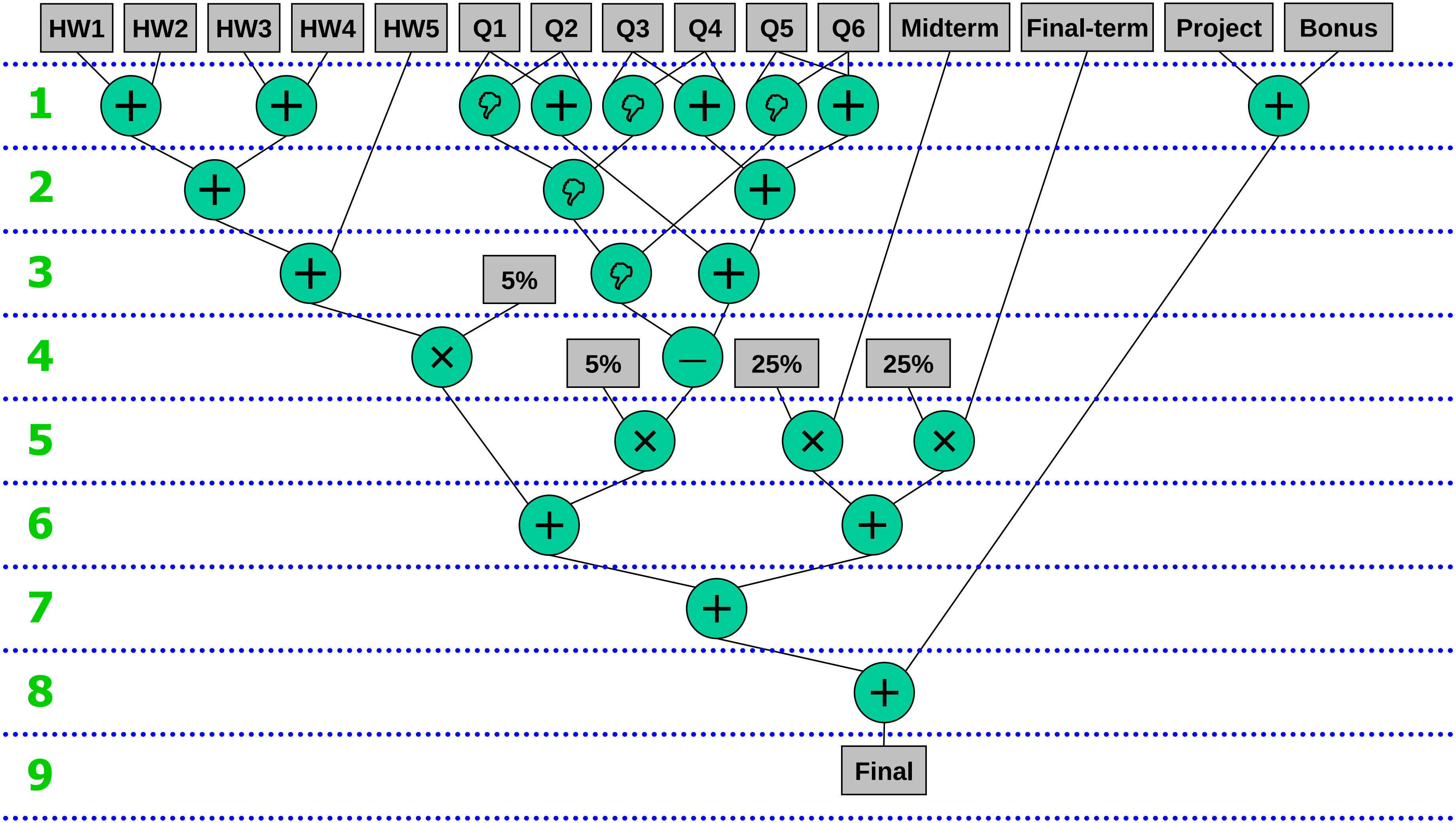
Overall DFG


HW1
HW2
HW3
HW4
HW5
Q1
Q2
Q3
Q4
Q5
Q6
Midterm
Final-term
Project
Bonus
5%
5%
25%
25%
Final

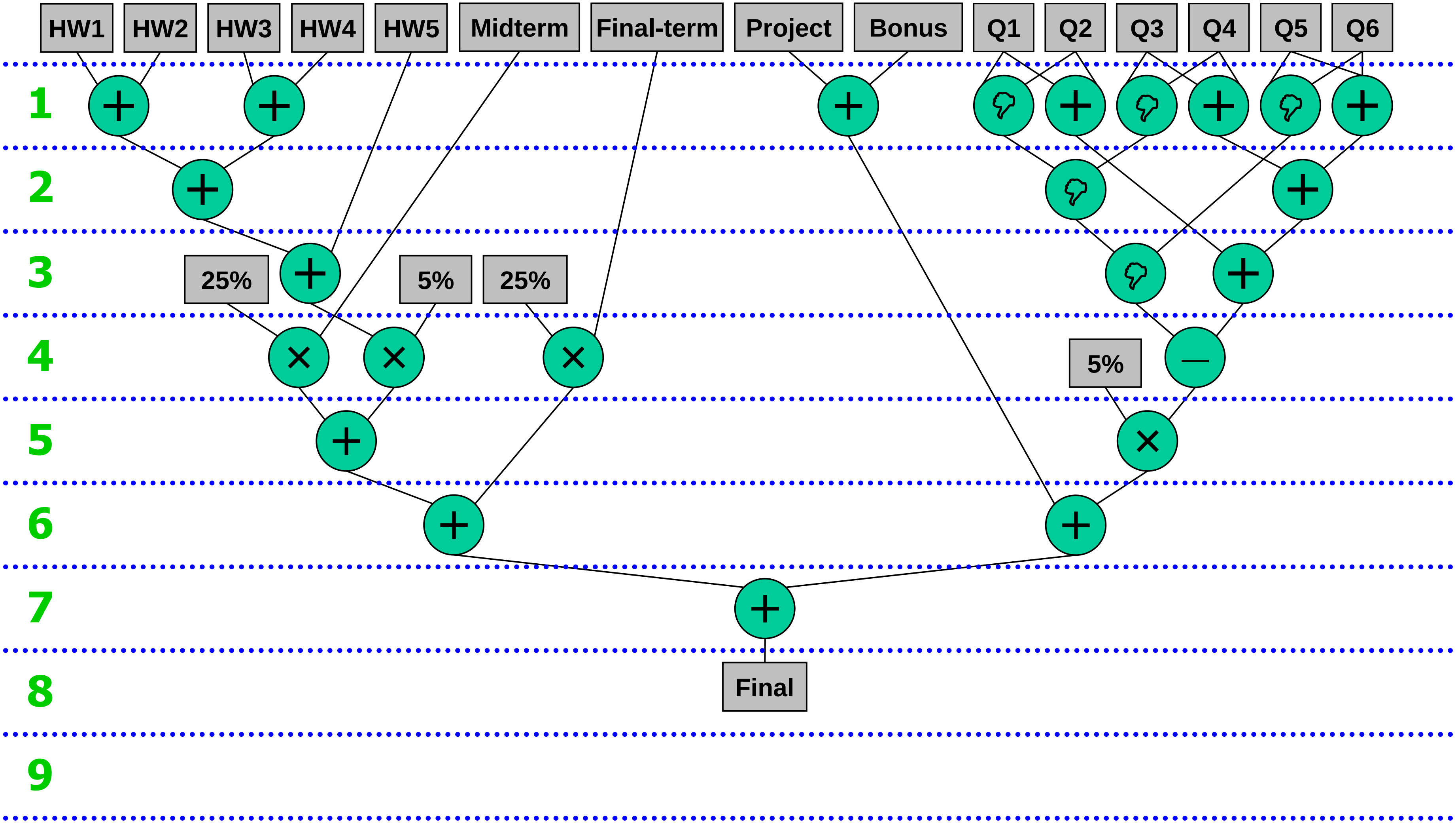
Control Steps



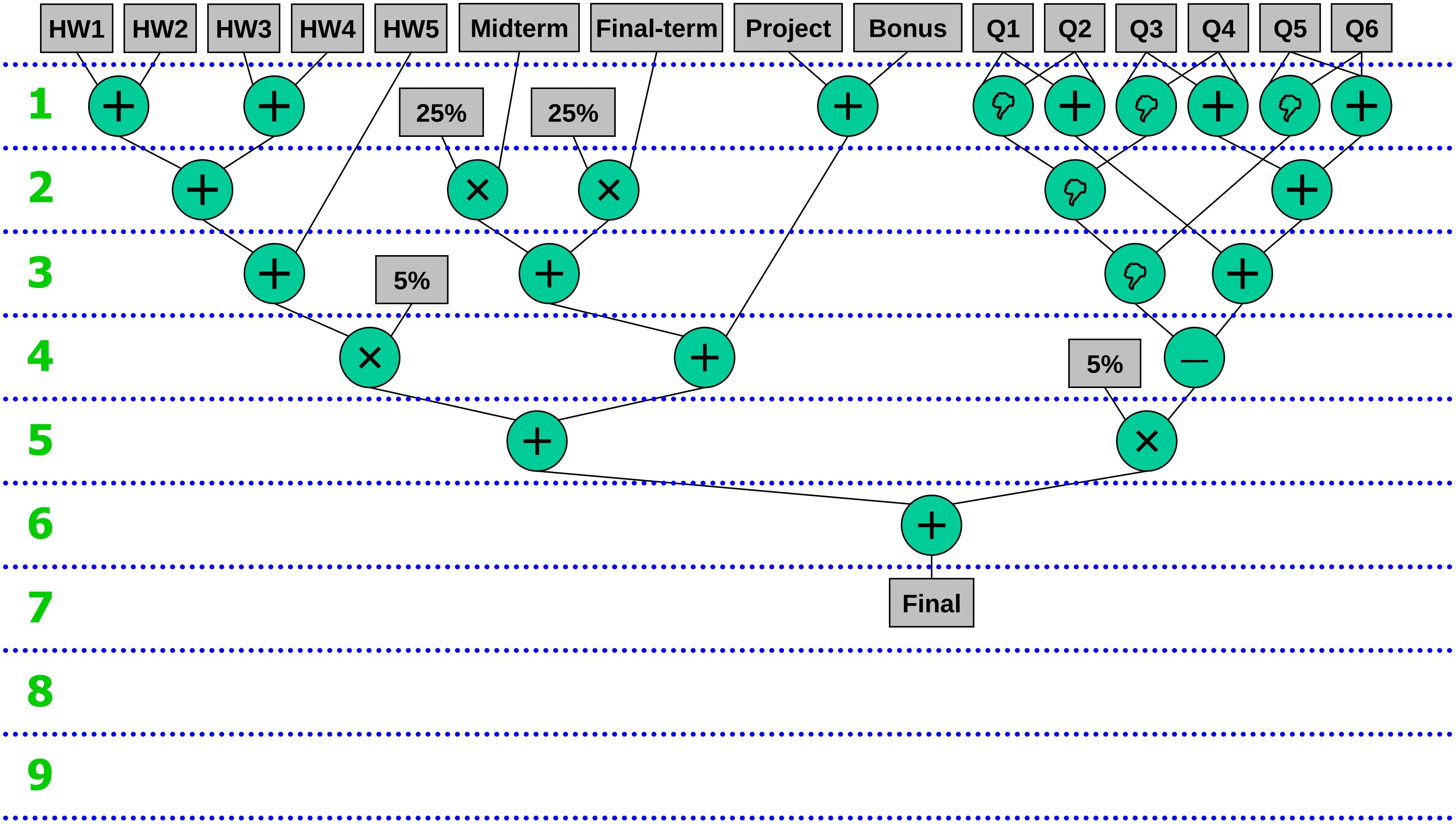
8 Control Steps



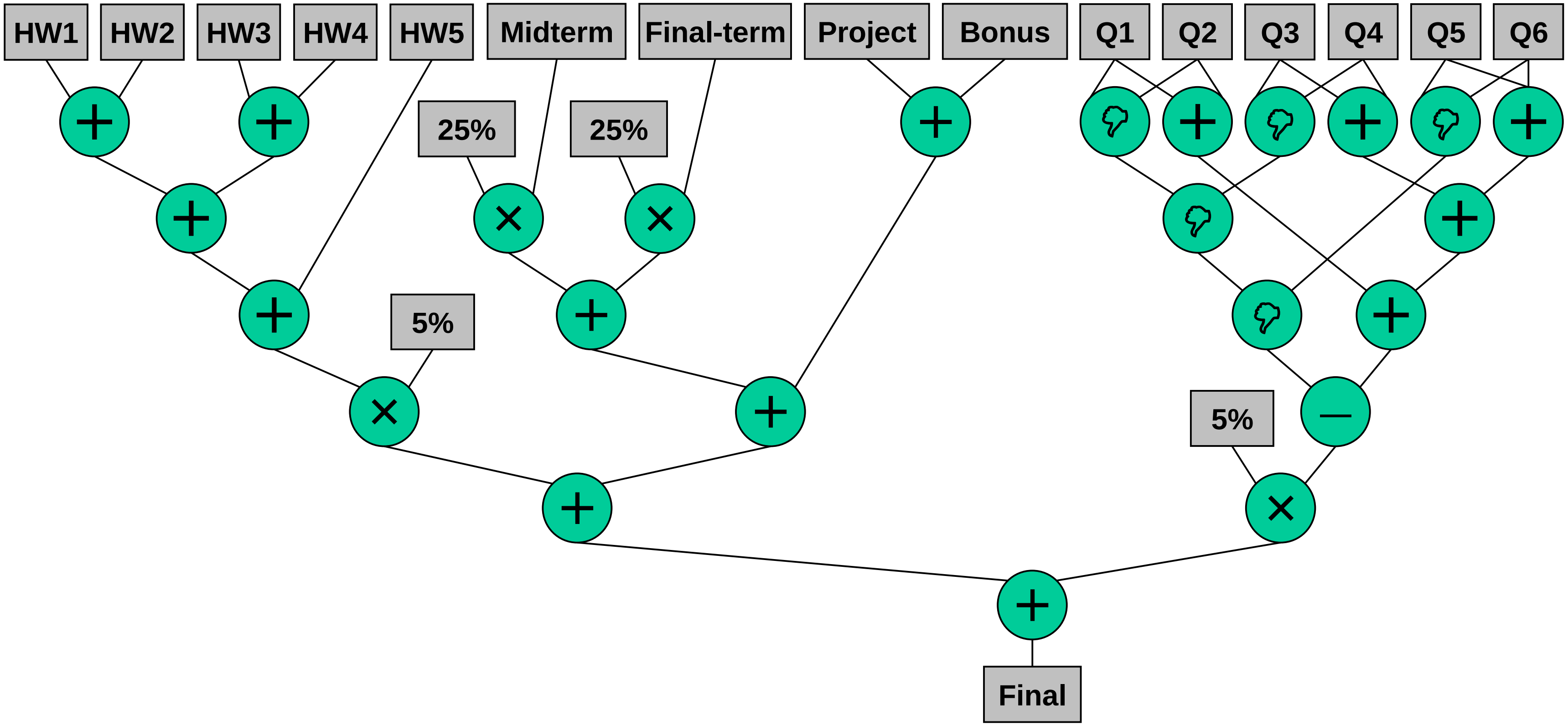
7 Control Steps



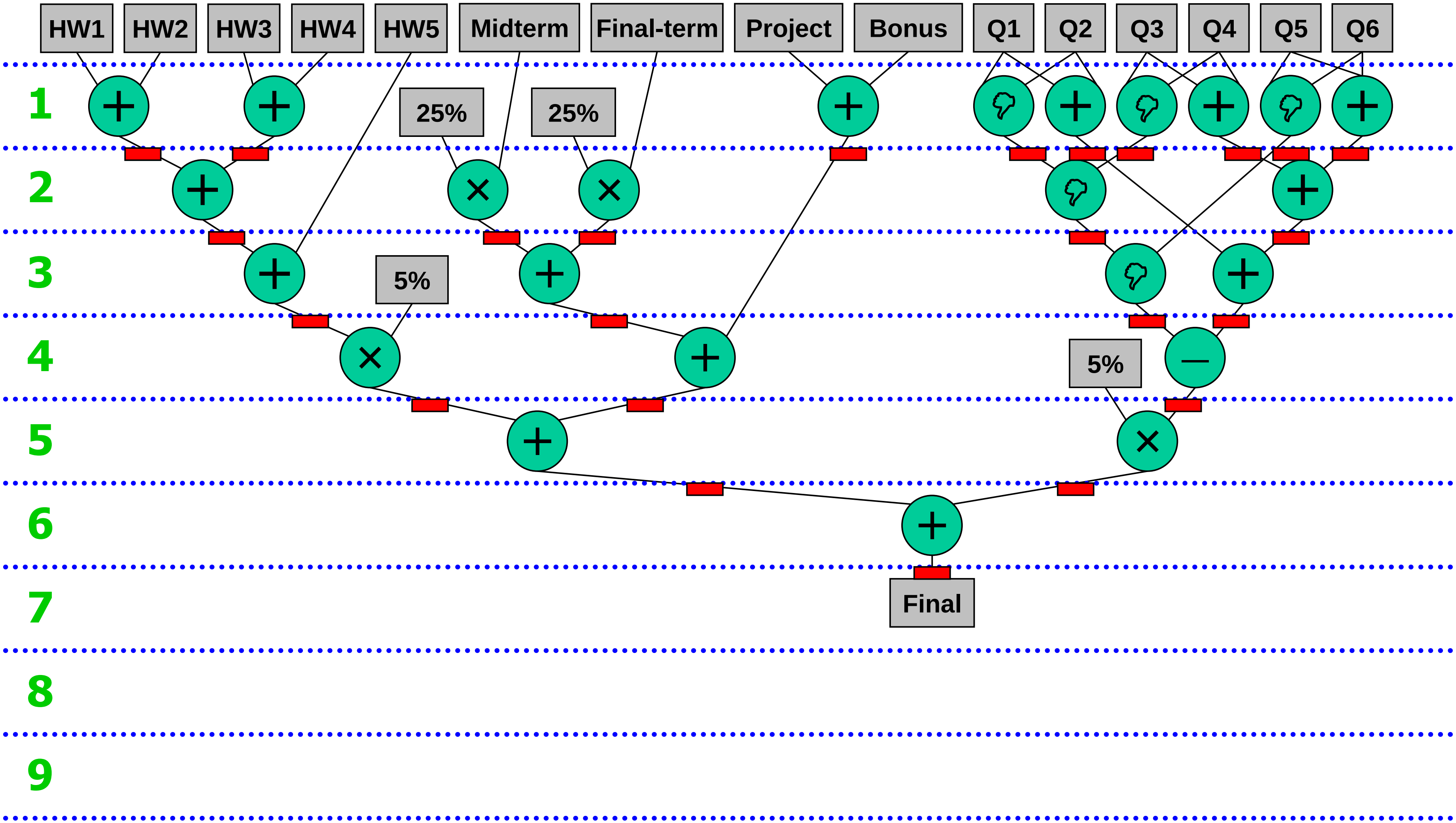
6 Control Steps



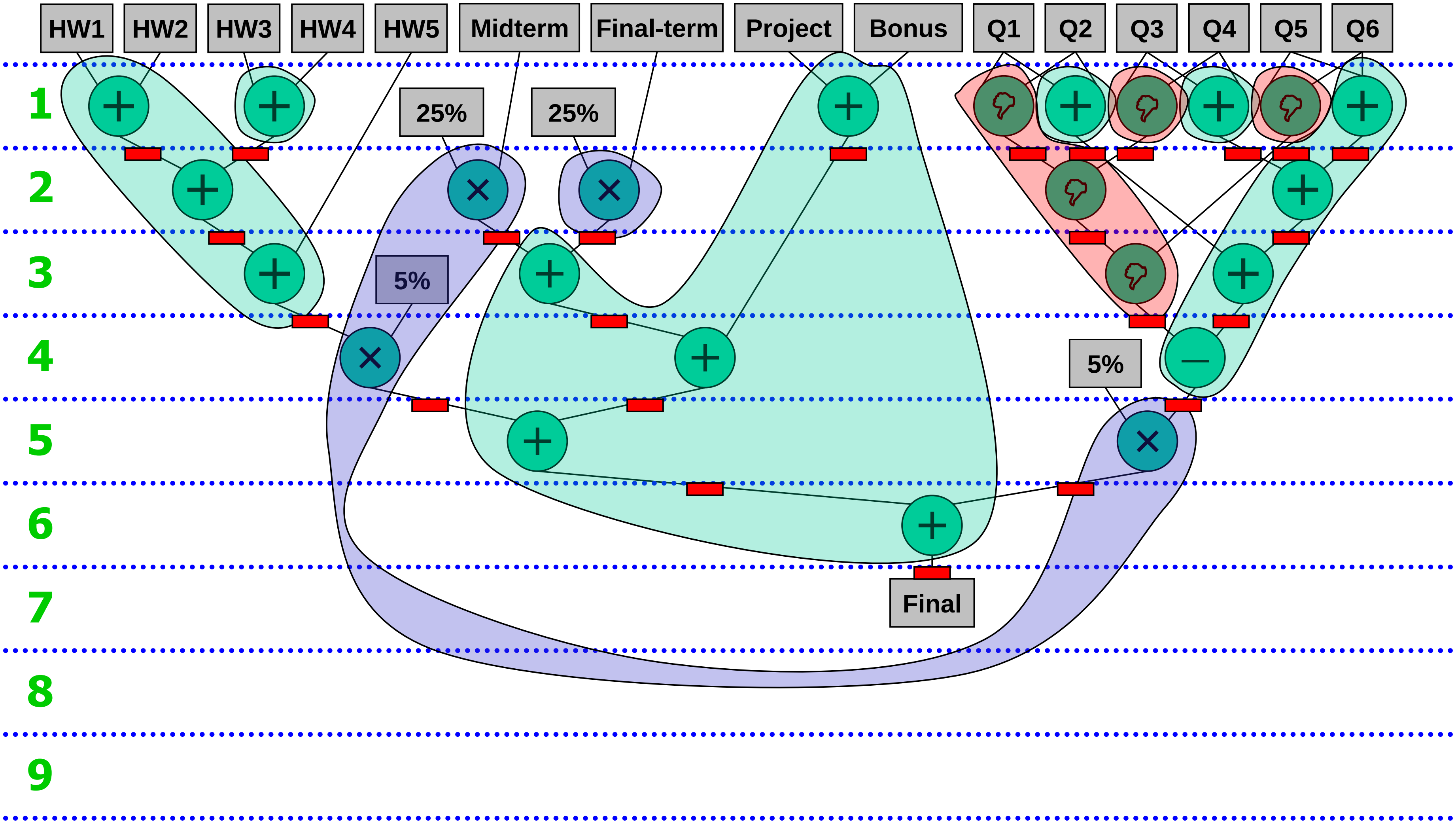
Single-cycle Implementation



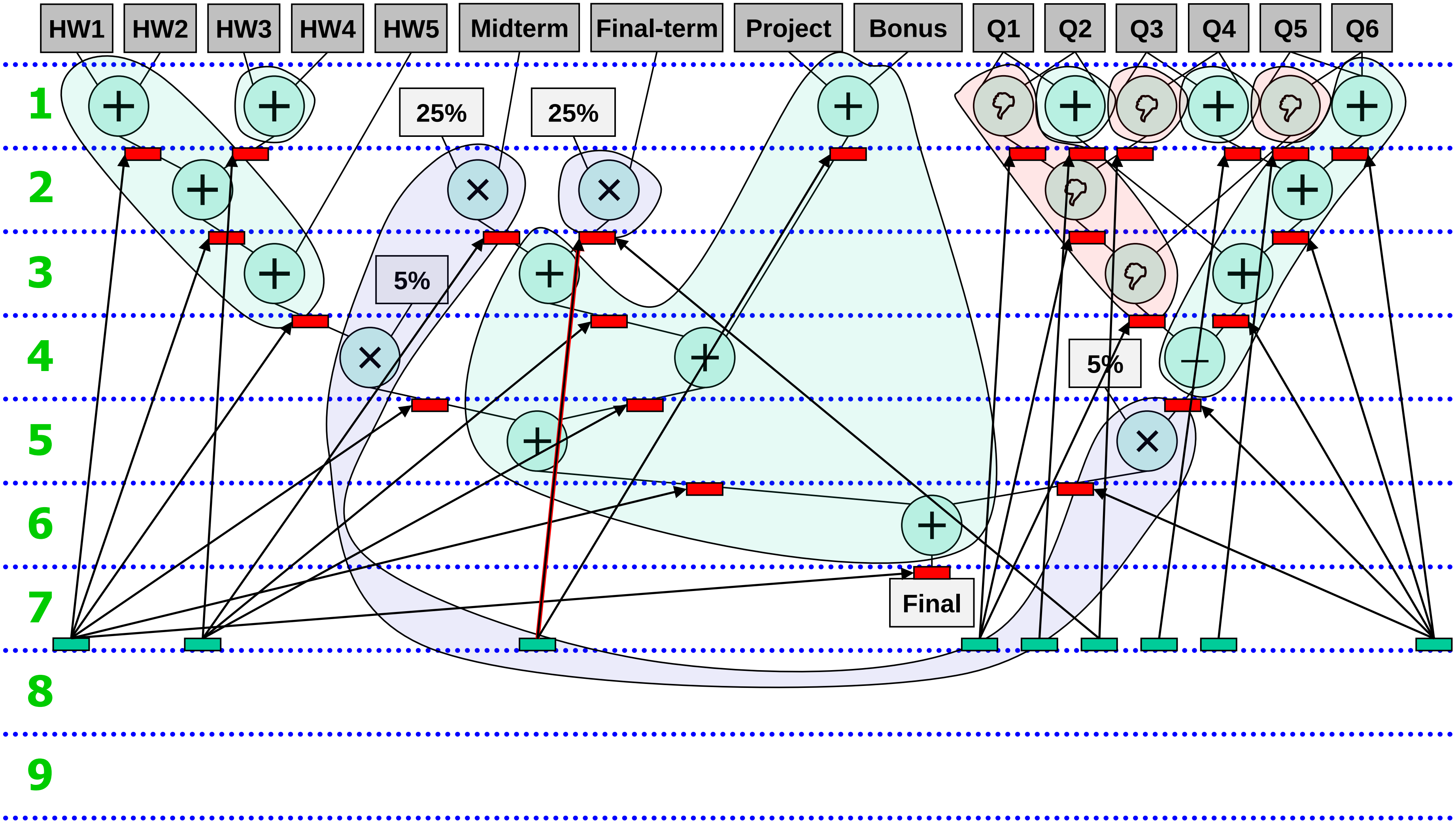
Multi-cycle Implementation



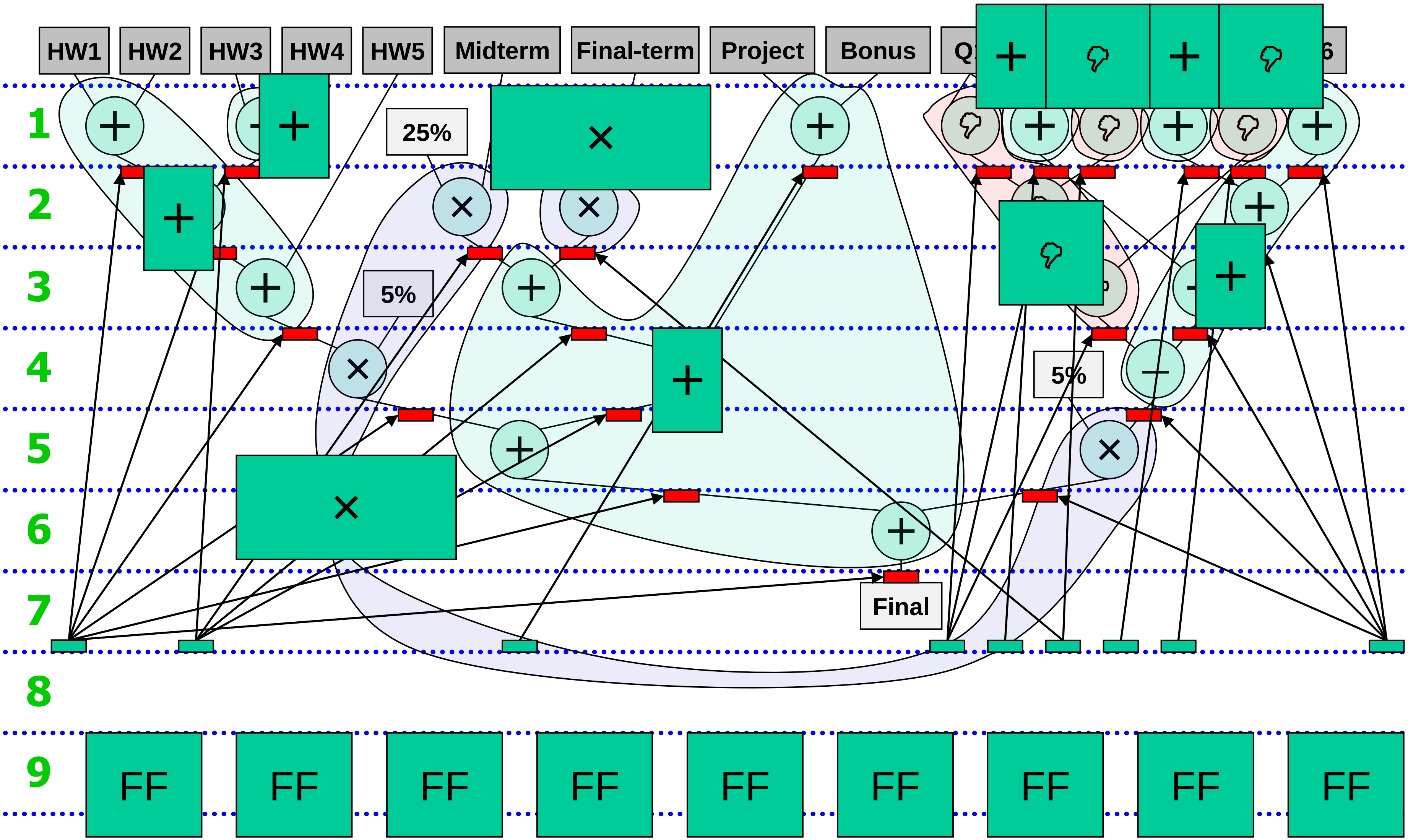
Resource Binding



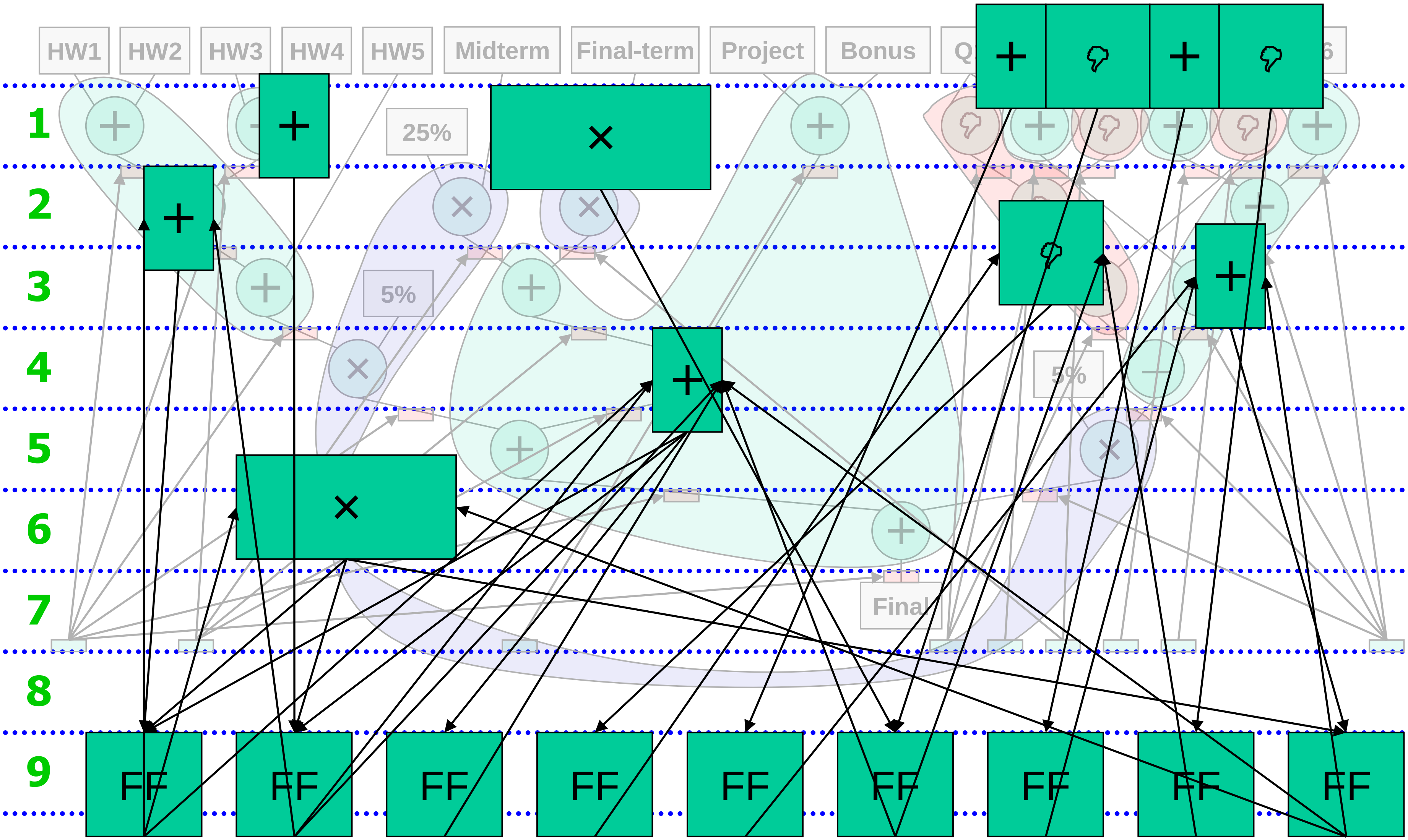
Register Binding



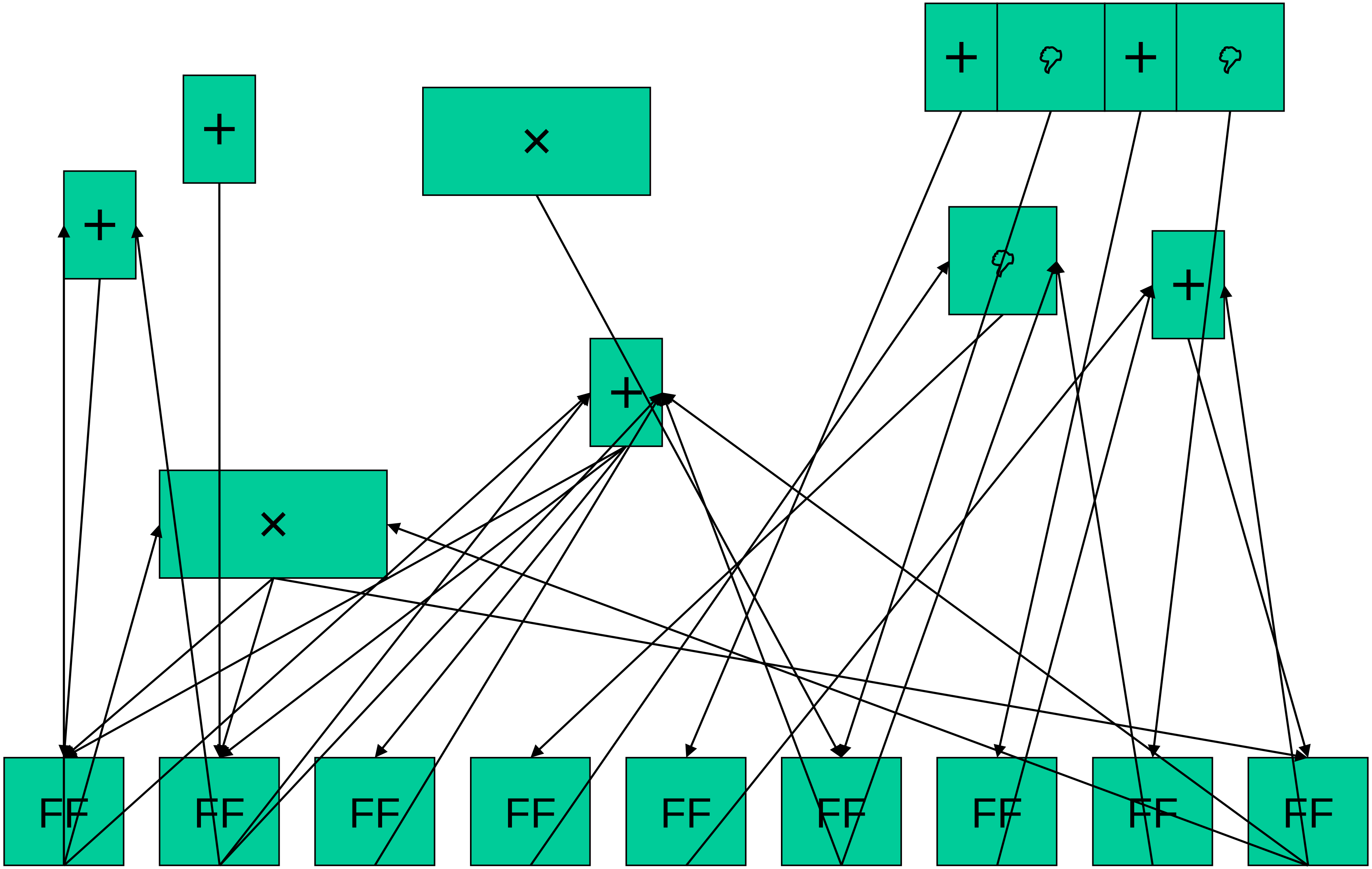
Resource Allocation



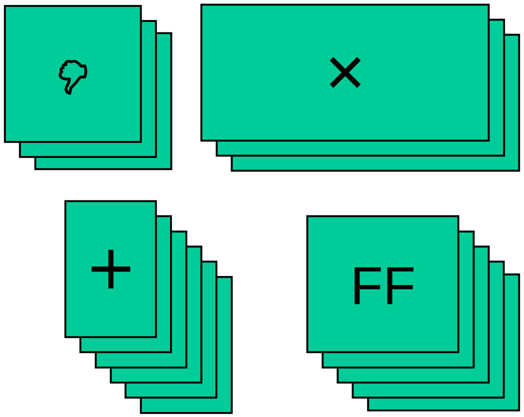
Resource Allocation



Cell Connectivity



Floorplan



1-row height

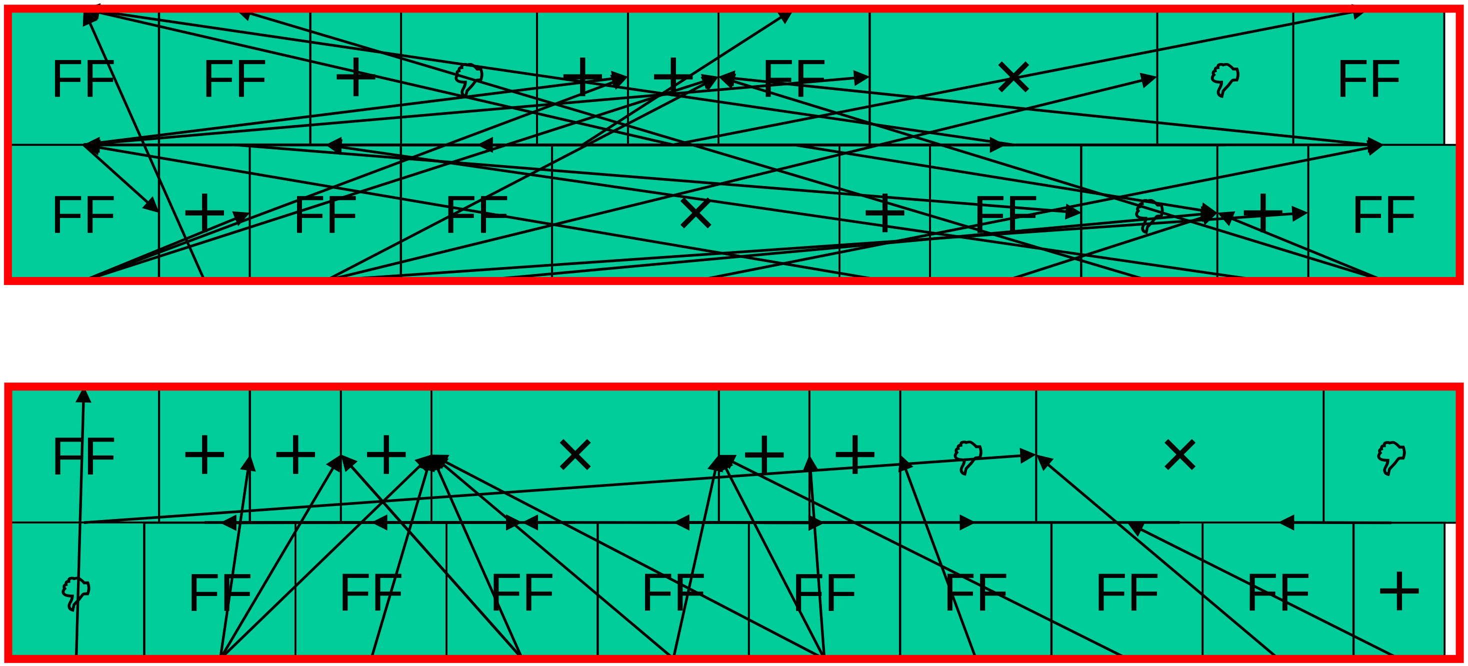
2-row height

3-row height

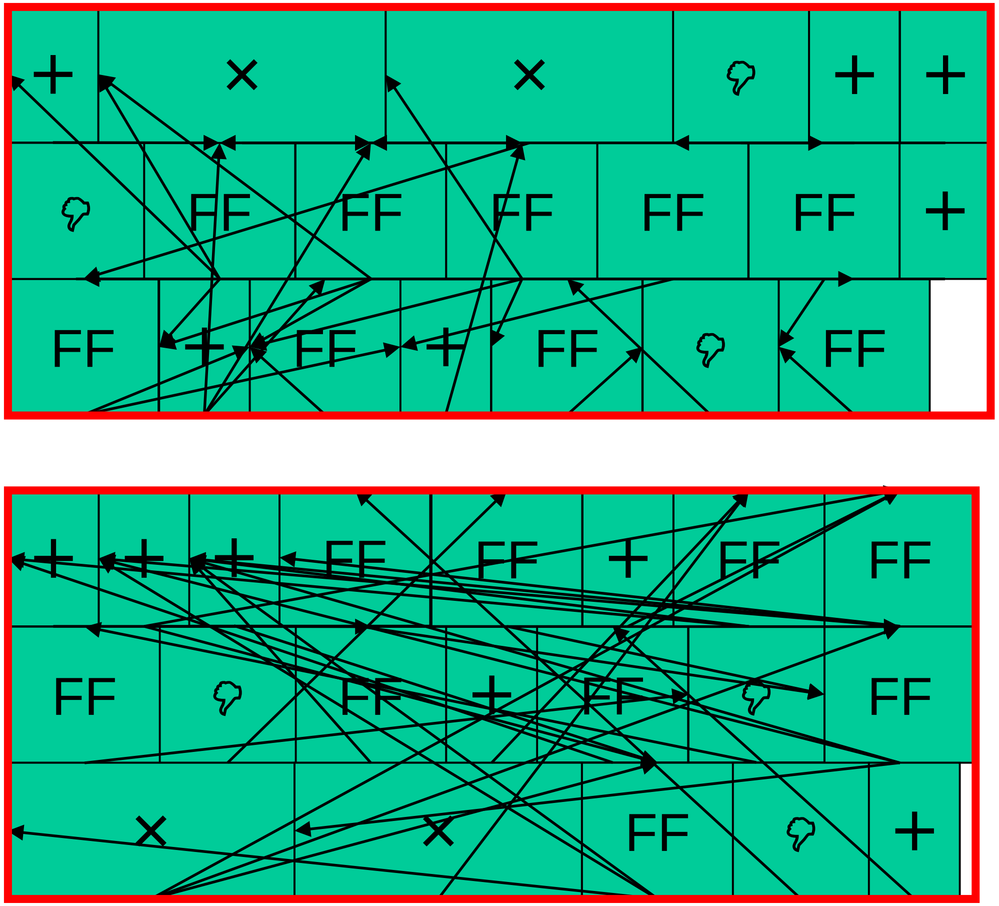
4-row height

5-row height

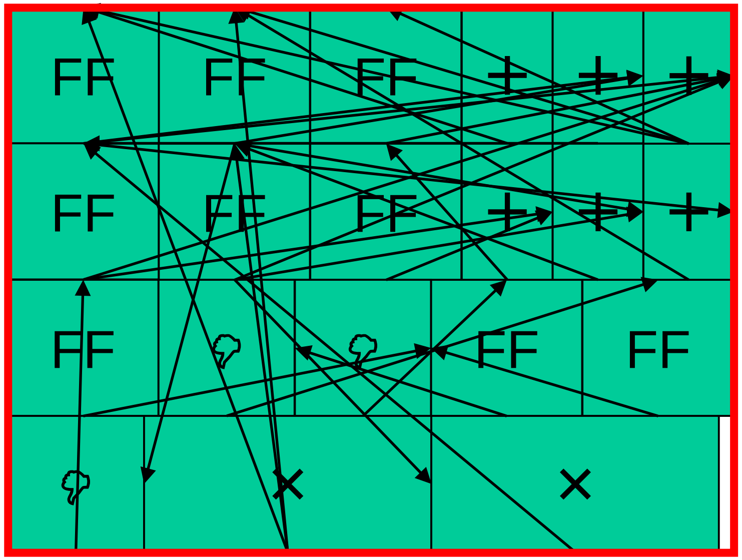
Cell Placement in 2 Rows

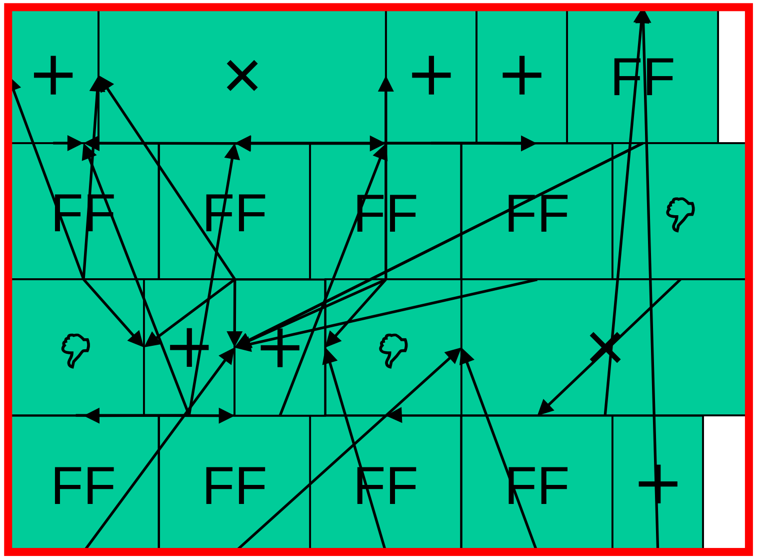


Cell Placement in 3 Rows

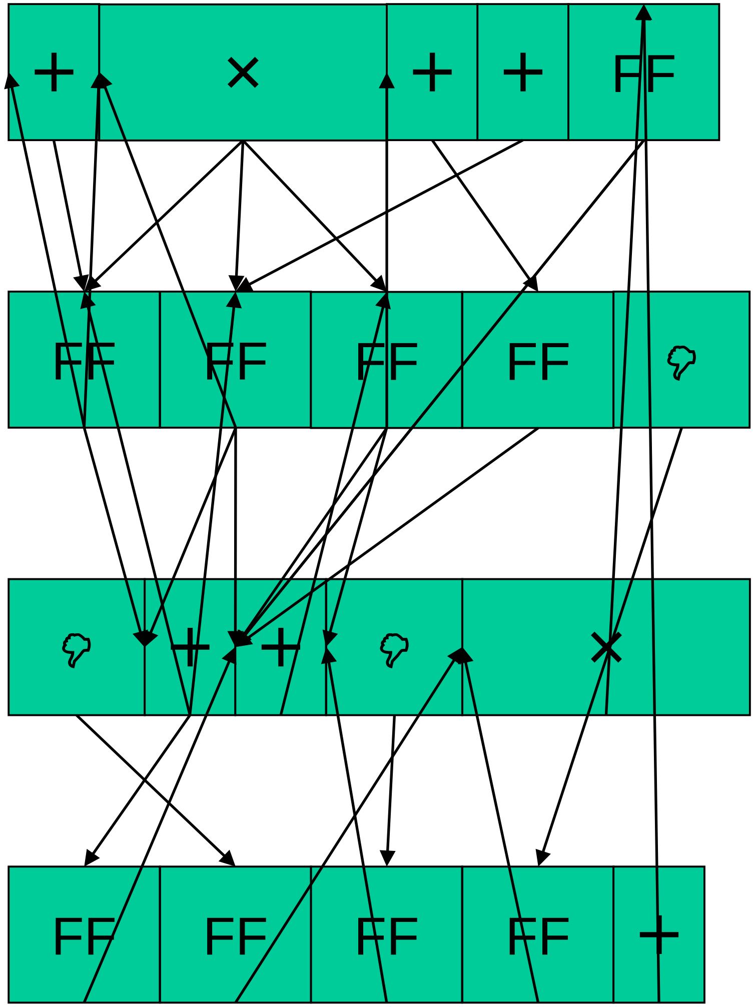


Cell Placement in 4 Rows

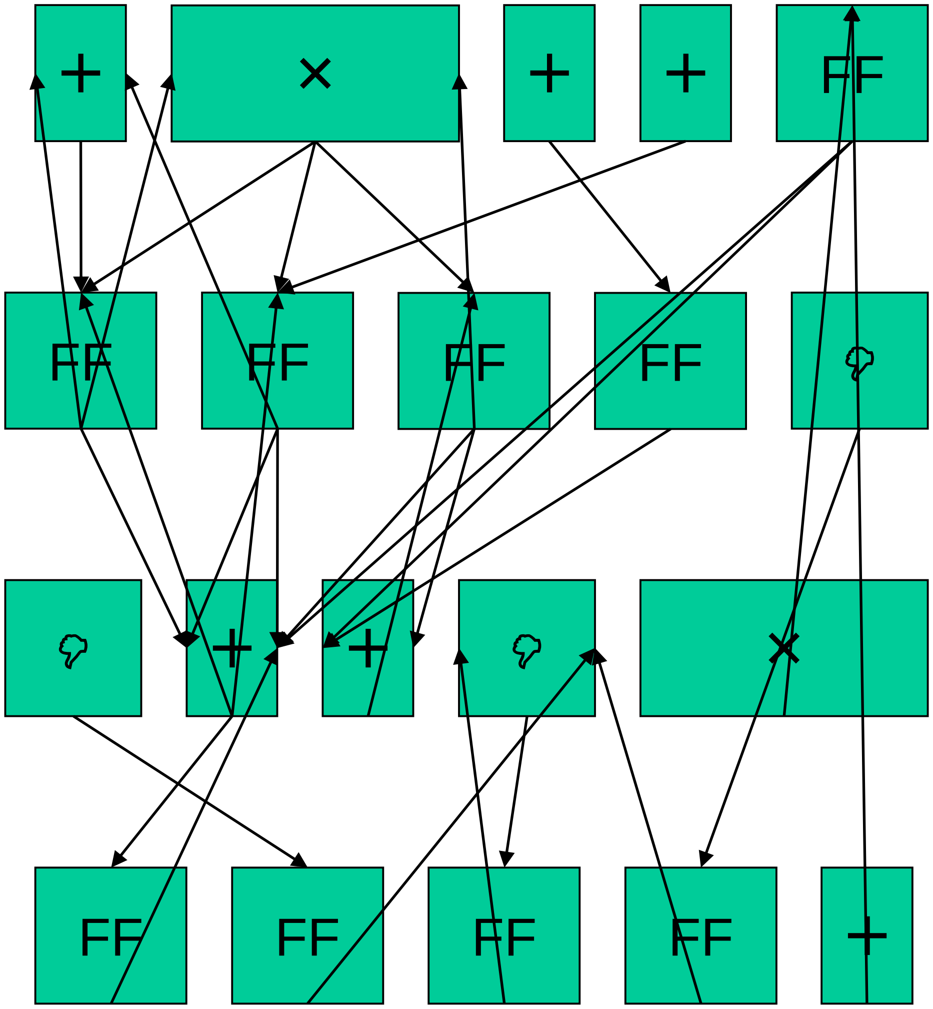




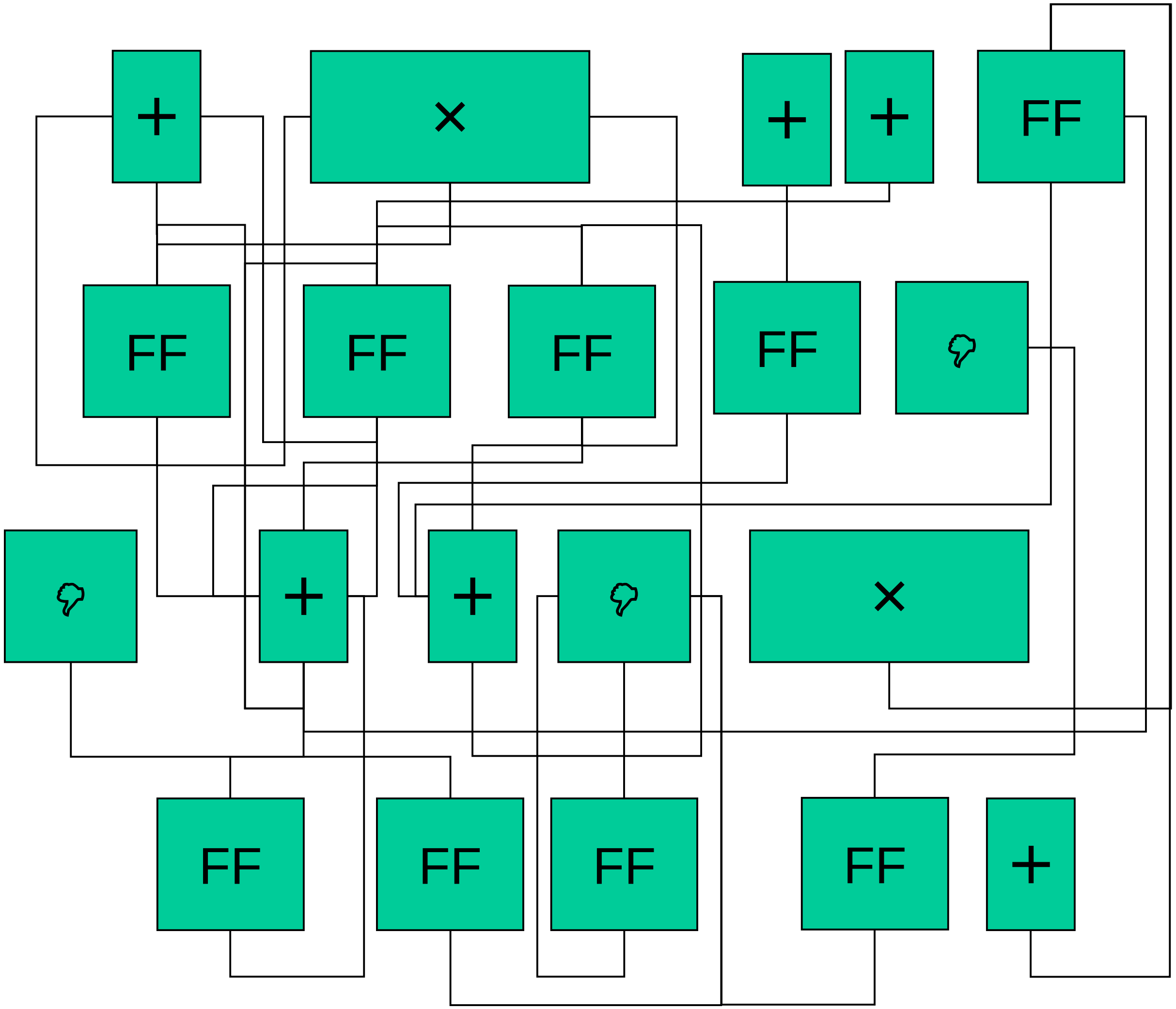
Routing



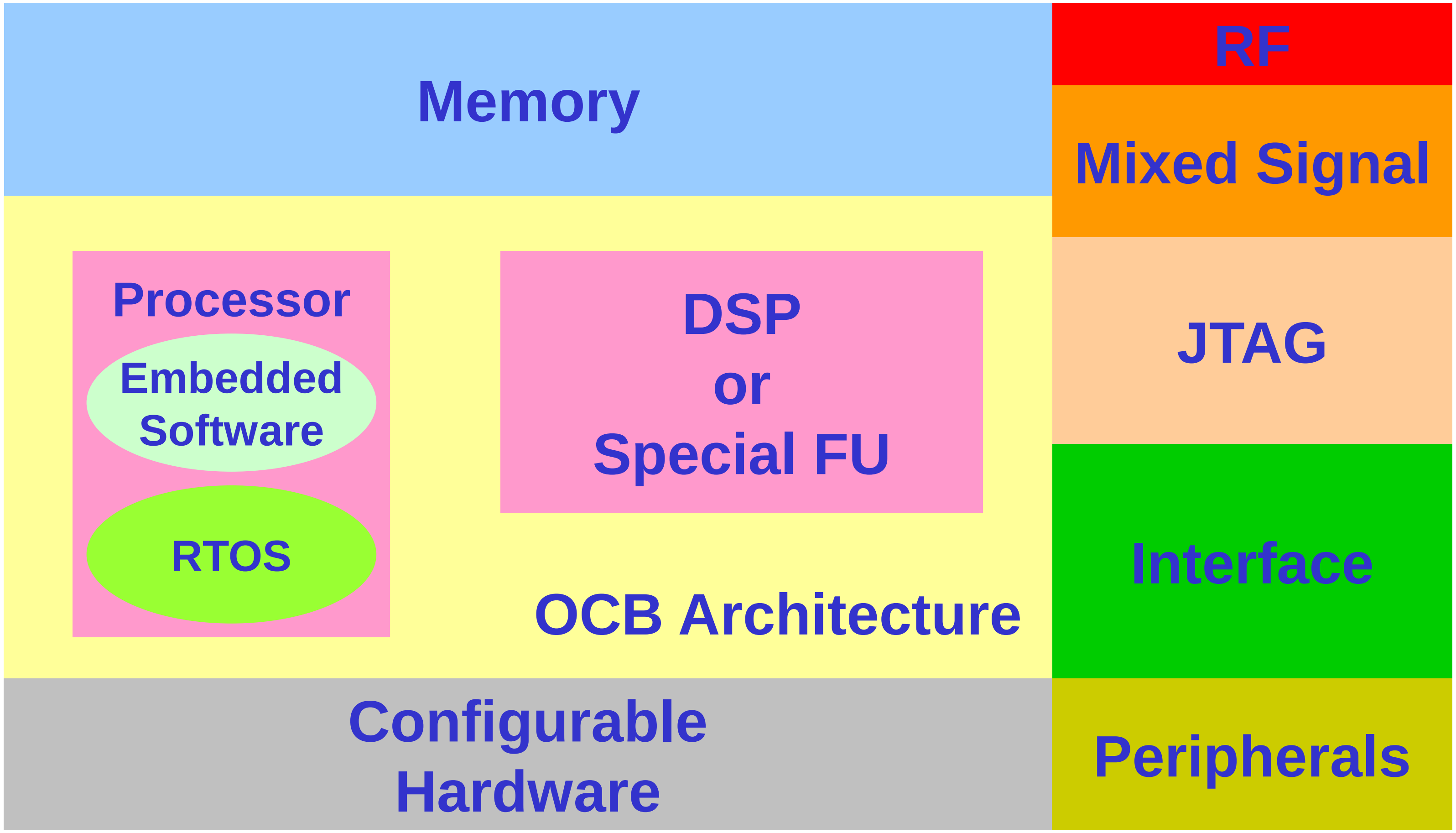
Routing



Routing



SoC Architecture



Memory

RF

Mixed Signal

Processor

Embedded Software

RTOS

DSP or Special FU

JTAG

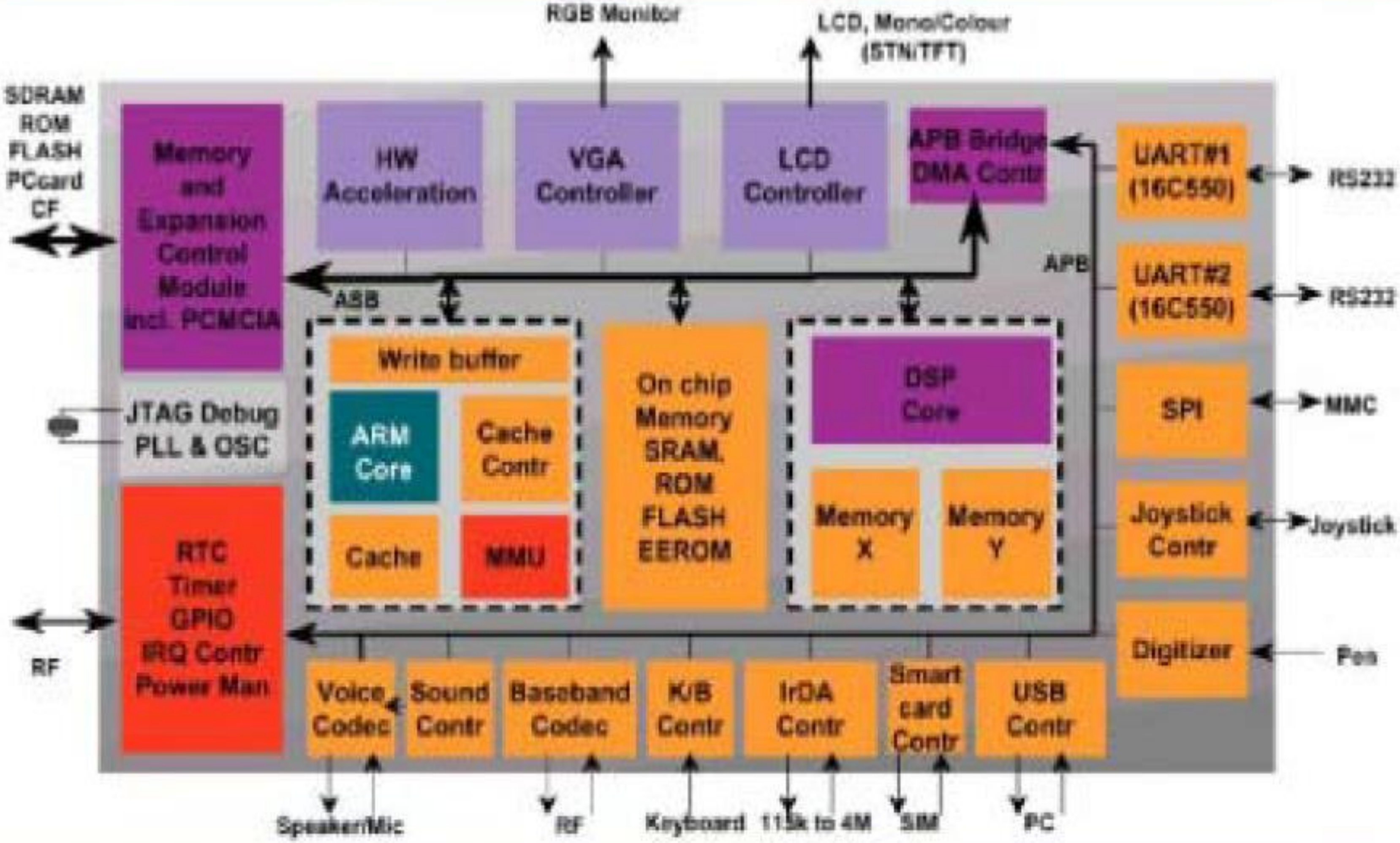
OCB Architecture

Interface

Configurable Hardware

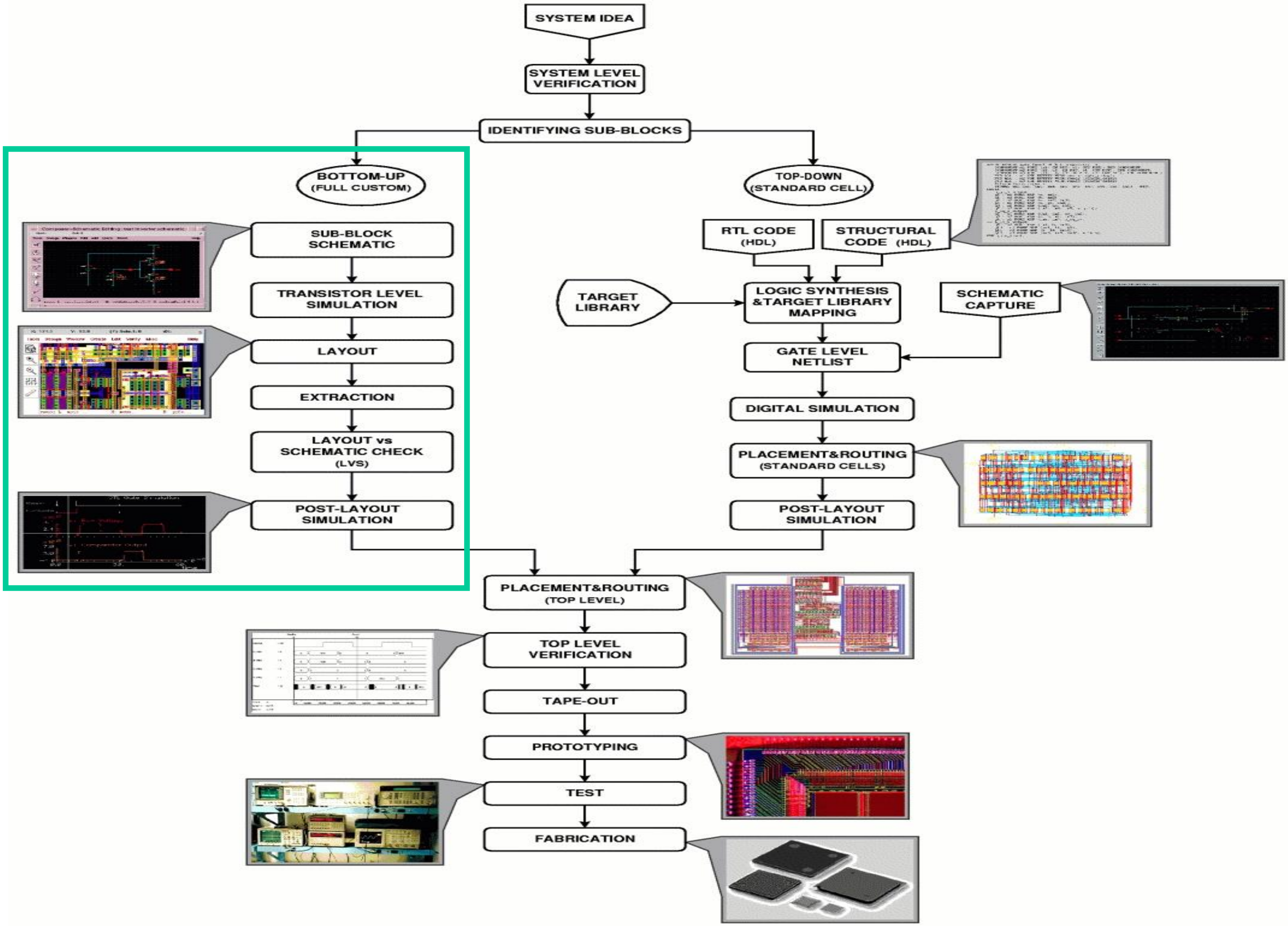
Peripherals

Generic Wireless / Computing



Source: **ARM Solutions** Smart Phones and Communicators

VLSI DESIGN FLOW



SYSTEM IDEA

SYSTEM LEVEL VERIFICATION

IDENTIFYING SUB-BLOCKS

BOTTOM-UP (FULL CUSTOM)

SUB-BLOCK SCHEMATIC

TRANSISTOR LEVEL SIMULATION

LAYOUT

EXTRACTION

LAYOUT vs SCHEMATIC CHECK (LVS)

POST-LAYOUT SIMULATION

TOP-DOWN (STANDARD CELL)

RTL CODE (HDL)

STRUCTURAL CODE (HDL)

TARGET LIBRARY

LOGIC SYNTHESIS &TARGET LIBRARY MAPPING

SCHEMATIC CAPTURE

GATE LEVEL NETLIST

DIGITAL SIMULATION

PLACEMENT&ROUTING (STANDARD CELLS)

POST-LAYOUT SIMULATION

PLACEMENT&ROUTING (TOP LEVEL)

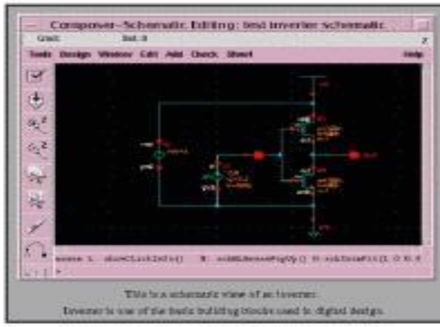
TOP LEVEL VERIFICATION

TAPE-OUT

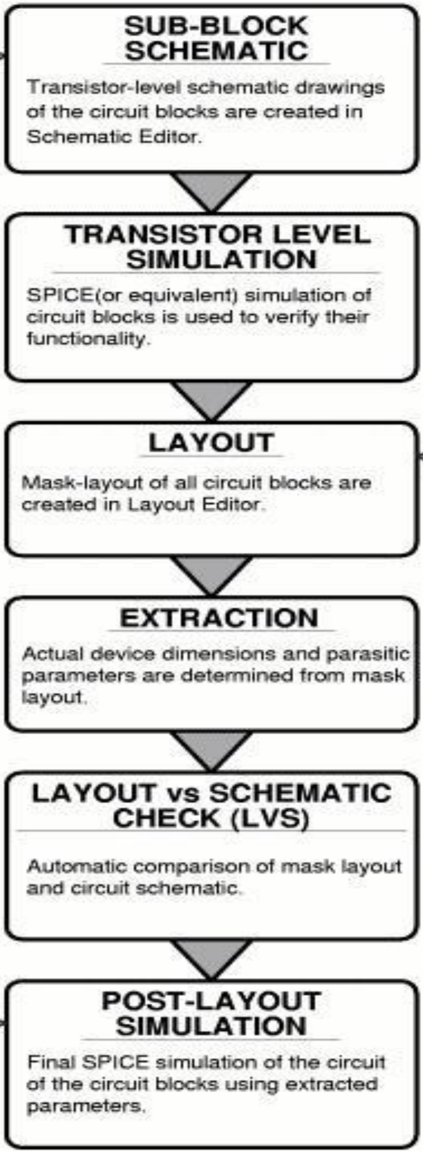
PROTOTYPING

TEST

FABRICATION


Composer–Schematic Editing: test inverter schematic
Tools Design Window Edit Add Check Sheet Help

This is a schematic view of an inverter.
Inverter is one of the basic building blocks used in digital design.



SUB-BLOCK SCHEMATIC

Transistor-level schematic drawings of the circuit blocks are created in Schematic Editor.

TRANSISTOR LEVEL SIMULATION

SPICE(or equivalent) simulation of circuit blocks is used to verify their functionality.

LAYOUT

Mask-layout of all circuit blocks are created in Layout Editor.

EXTRACTION

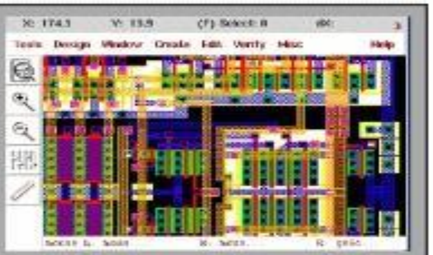
Actual device dimensions and parasitic parameters are determined from mask layout.

LAYOUT vs SCHEMATIC CHECK (LVS)

Automatic comparison of mask layout and circuit schematic.

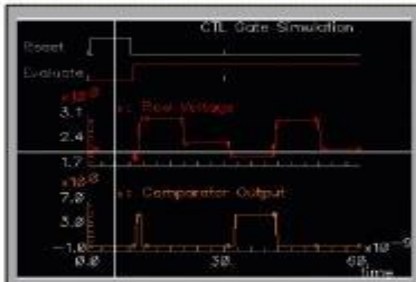
POST-LAYOUT SIMULATION

Final SPICE simulation of the circuit of the circuit blocks using extracted parameters.



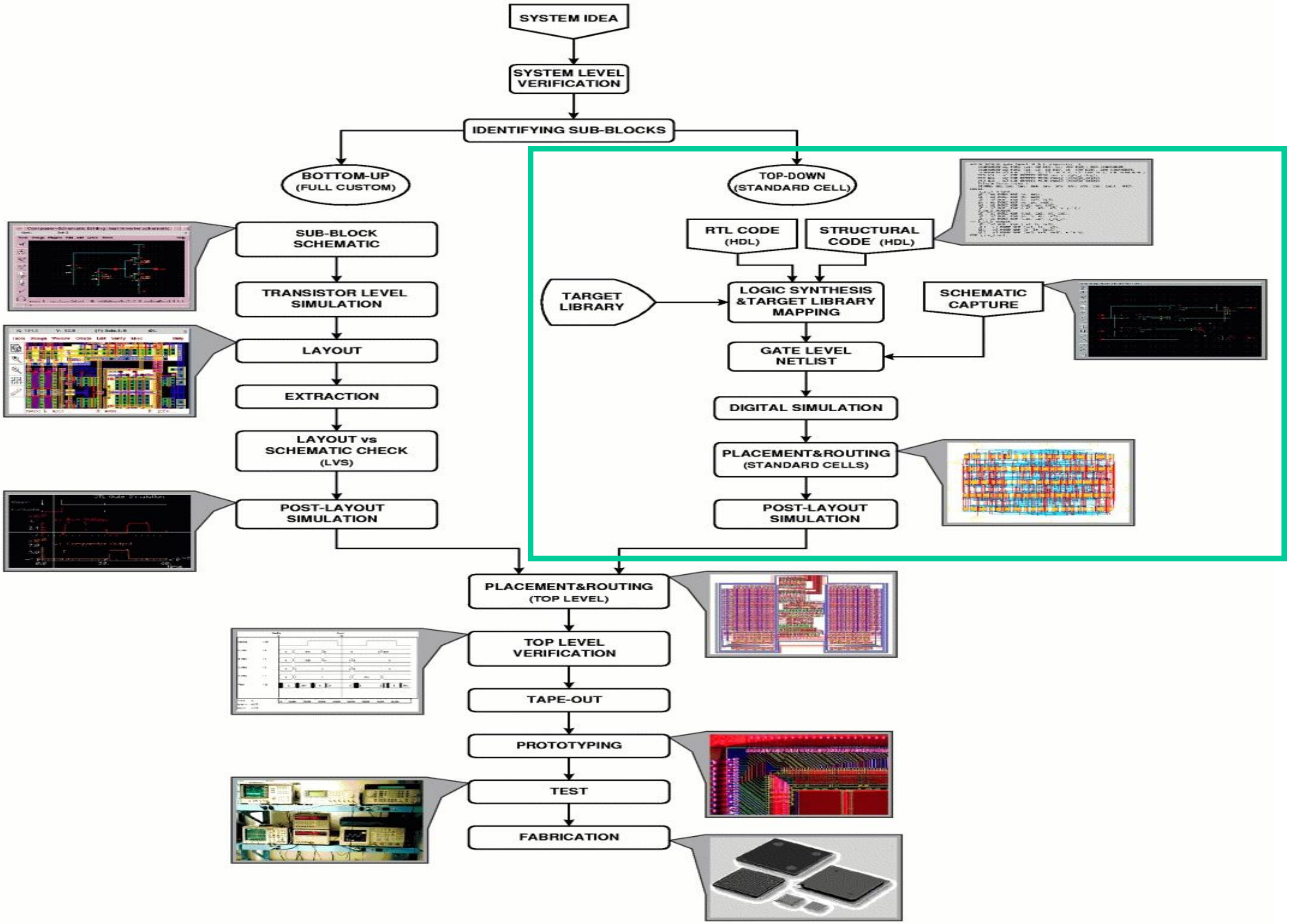
The layout view of a building block

Blue : Metal 1	Orange : Metal 2
Red : Poly layer	Green : Diffusion layer



Input and output waveforms of a block
The waveform shows the behavior of the block to pre-defined input vectors.

VLSI DESIGN FLOW



SYSTEM IDEA

SYSTEM LEVEL VERIFICATION

IDENTIFYING SUB-BLOCKS

BOTTOM-UP (FULL CUSTOM)

SUB-BLOCK SCHEMATIC

TRANSISTOR LEVEL SIMULATION

LAYOUT

EXTRACTION

LAYOUT vs SCHEMATIC CHECK (LVS)

POST-LAYOUT SIMULATION

TOP-DOWN (STANDARD CELL)

RTL CODE (HDL)

STRUCTURAL CODE (HDL)

TARGET LIBRARY

LOGIC SYNTHESIS &TARGET LIBRARY MAPPING

SCHEMATIC CAPTURE

GATE LEVEL NETLIST

DIGITAL SIMULATION

PLACEMENT&ROUTING (STANDARD CELLS)

POST-LAYOUT SIMULATION

PLACEMENT&ROUTING (TOP LEVEL)

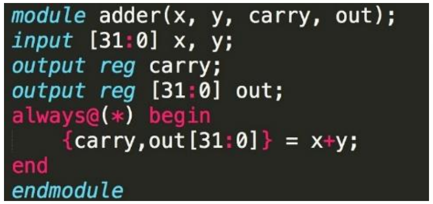
TOP LEVEL VERIFICATION

TAPE-OUT

PROTOTYPING

TEST

FABRICATION



```
module adder(x, y, carry, out);
input [31:0] x, y;
output reg carry;
output reg [31:0] out;
always@(*) begin
    {carry,out[31:0]} = x+y;
end
endmodule
```

RTL CODE (HDL)

Register-transfer level code to describe logic functionality.

STRUCTURAL CODE (HDL)

Detailed code to describe gate-level structure.

TARGET LIBRARY

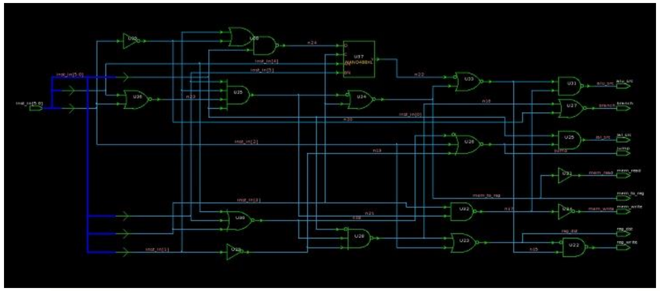
Available cells and functions.

LOGIC SYNTHESIS & TARGET LIBRARY MAPPING

Generate gate-level description using target library cells.

SCHEMATIC CAPTURE

Alternative to HDL code.



GATE LEVEL NETLIST

DIGITAL SIMULATION

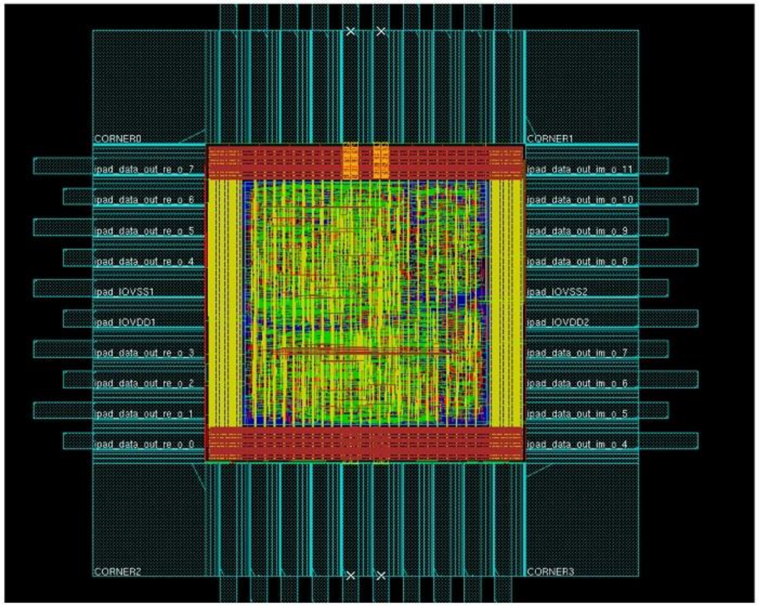
Verify the logic functionality of the circuit.

PLACEMENT&ROUTING (STANDARD CELLS)

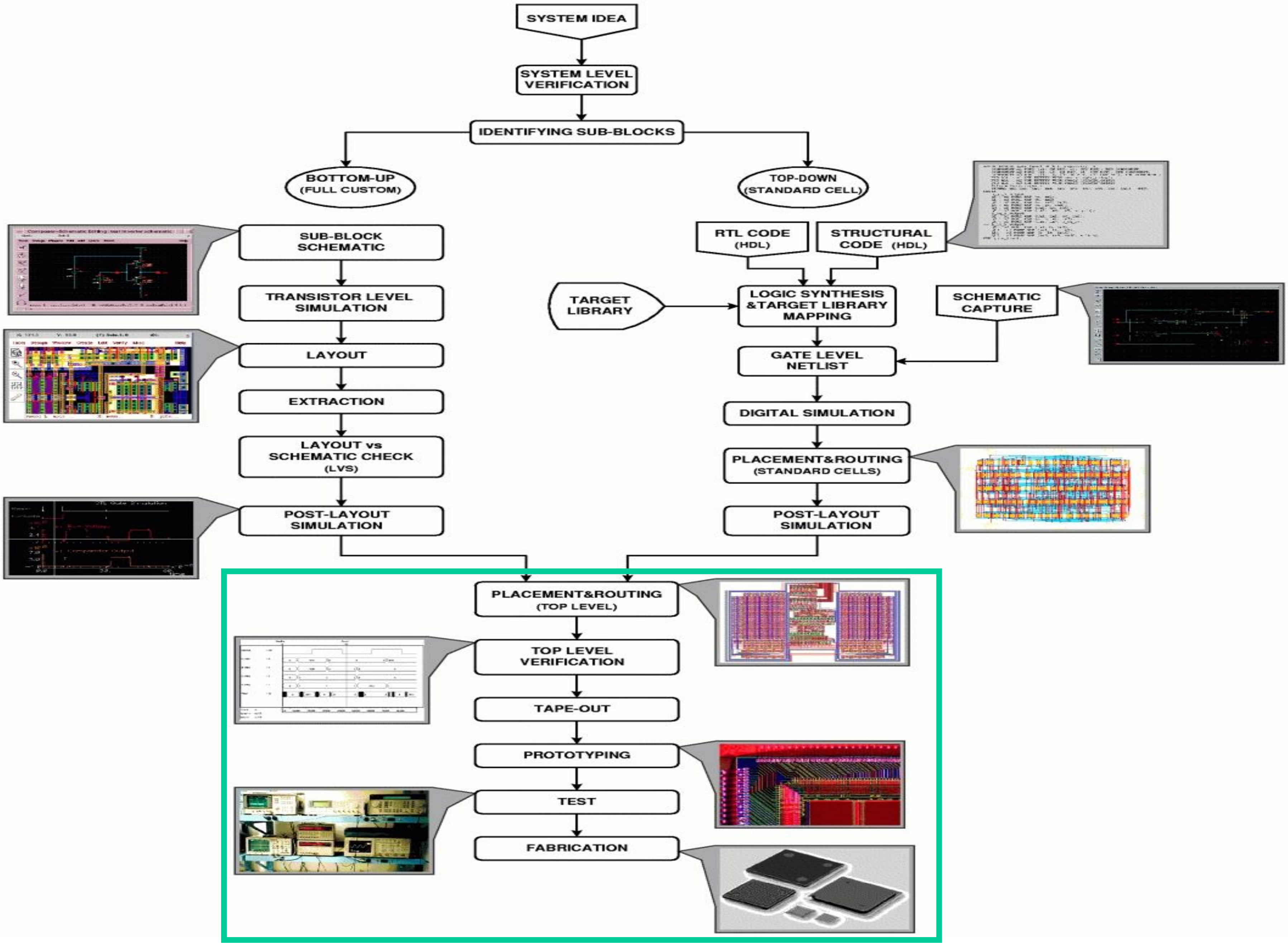
Create the circuit layout using an automatic placement and routing tool.

POST-LAYOUT SIMULATION

Final logic simulation to verify actual delays and circuit performance.



VLSI DESIGN FLOW



SYSTEM IDEA

SYSTEM LEVEL VERIFICATION

IDENTIFYING SUB-BLOCKS

BOTTOM-UP (FULL CUSTOM)

SUB-BLOCK SCHEMATIC

TRANSISTOR LEVEL SIMULATION

LAYOUT

EXTRACTION

LAYOUT vs SCHEMATIC CHECK (LVS)

POST-LAYOUT SIMULATION

TOP-DOWN (STANDARD CELL)

RTL CODE (HDL)

STRUCTURAL CODE (HDL)

TARGET LIBRARY

LOGIC SYNTHESIS &TARGET LIBRARY MAPPING

SCHEMATIC CAPTURE

GATE LEVEL NETLIST

DIGITAL SIMULATION

PLACEMENT&ROUTING (STANDARD CELLS)

POST-LAYOUT SIMULATION

PLACEMENT&ROUTING (TOP LEVEL)

TOP LEVEL VERIFICATION

TAPE-OUT

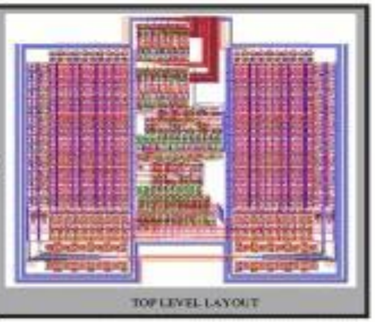
PROTOTYPING

TEST

FABRICATION

PLACEMENT&ROUTING (TOP LEVEL)

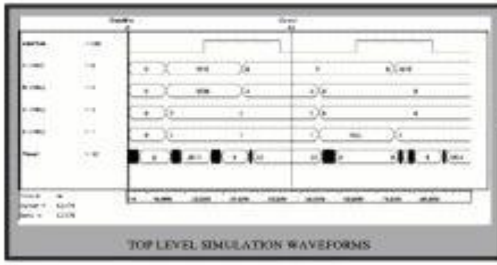
Mask level layout of the entire chip



TOP LEVEL LAYOUT

TOP LEVEL VERIFICATION

Simulation (mixed-mode) to verify functionality and performance of the entire chip.



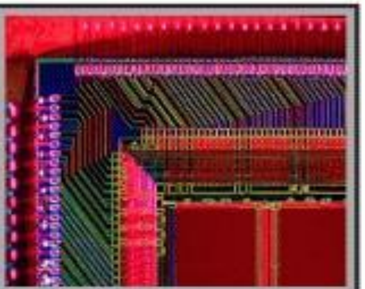
TOP LEVEL SIMULATION WAVEFORMS

TAPE-OUT

Create universal format file to describe mask layers to manufacturer.

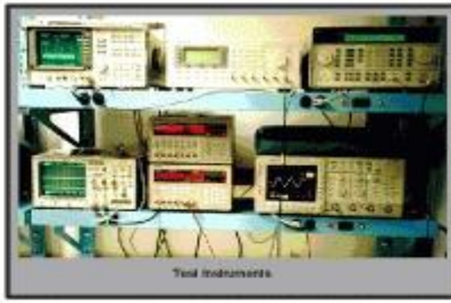
PROTOTYPING

Sample chips manufactured in fab.



TEST

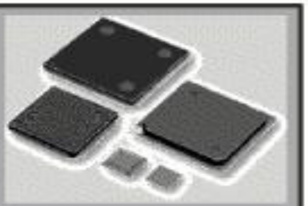
Performance verification and debugging of the prototype.



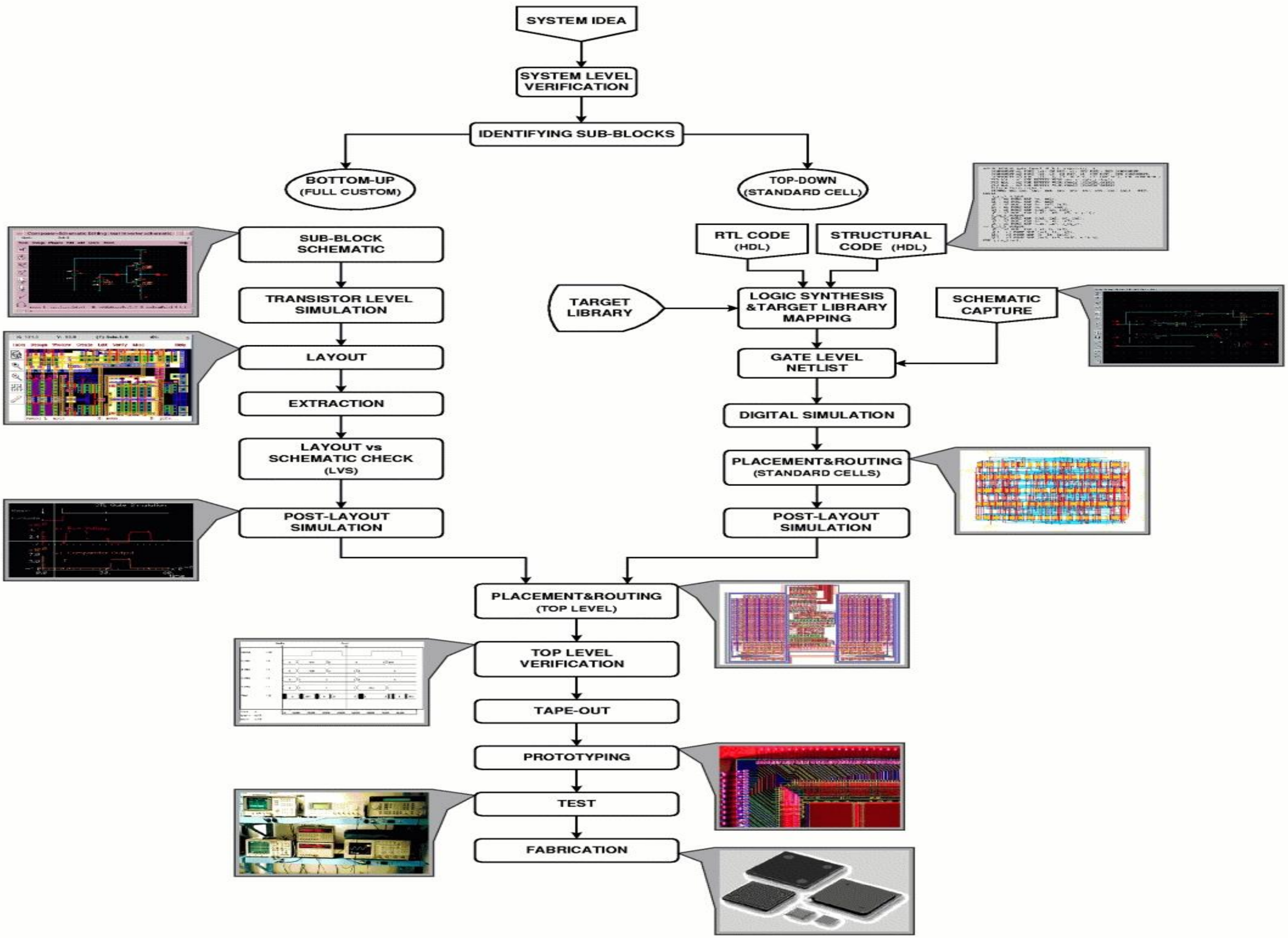
Test Instruments

FABRICATION

Mass-production of the designed chip.



VLSI DESIGN FLOW



SYSTEM IDEA

SYSTEM LEVEL VERIFICATION

IDENTIFYING SUB-BLOCKS

BOTTOM-UP (FULL CUSTOM)

SUB-BLOCK SCHEMATIC

TRANSISTOR LEVEL SIMULATION

LAYOUT

EXTRACTION

LAYOUT vs SCHEMATIC CHECK (LVS)

POST-LAYOUT SIMULATION

TOP-DOWN (STANDARD CELL)

RTL CODE (HDL)

STRUCTURAL CODE (HDL)

TARGET LIBRARY

LOGIC SYNTHESIS &TARGET LIBRARY MAPPING

SCHEMATIC CAPTURE

GATE LEVEL NETLIST

DIGITAL SIMULATION

PLACEMENT&ROUTING (STANDARD CELLS)

POST-LAYOUT SIMULATION

PLACEMENT&ROUTING (TOP LEVEL)

TOP LEVEL VERIFICATION

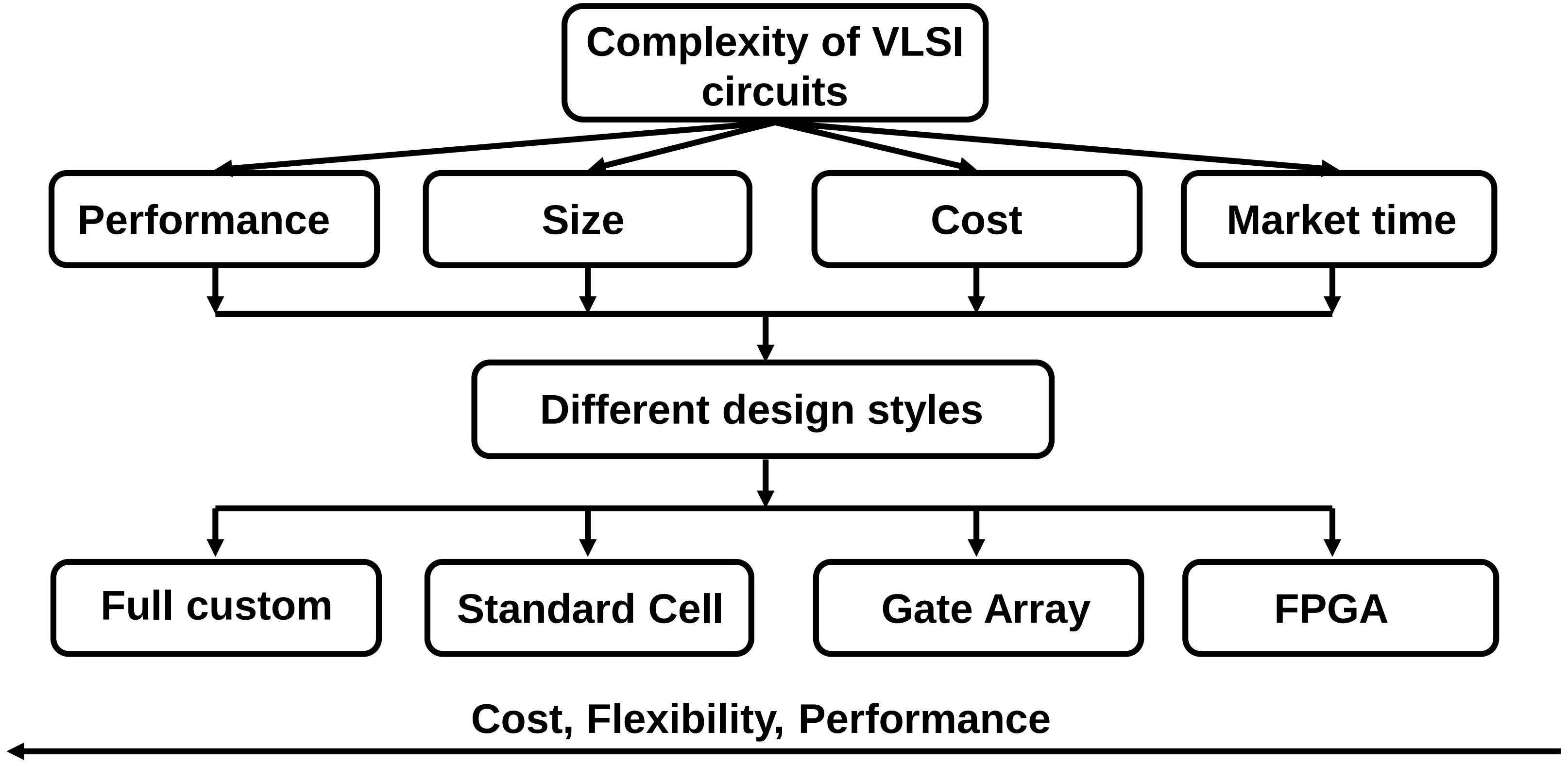
TAPE-OUT

PROTOTYPING

TEST

FABRICATION

Design Styles



Complexity of VLSI circuits

Performance

Size

Cost

Market time

Different design styles

Full custom

Standard Cell

Gate Array

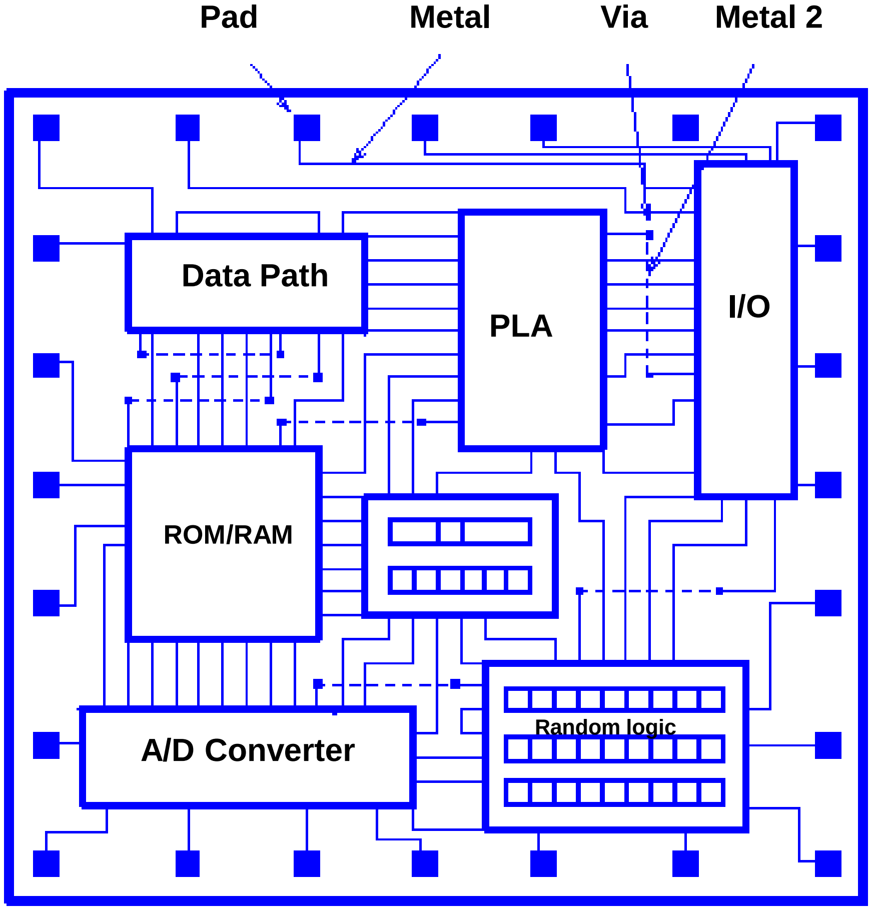
FPGA

Cost, Flexibility, Performance

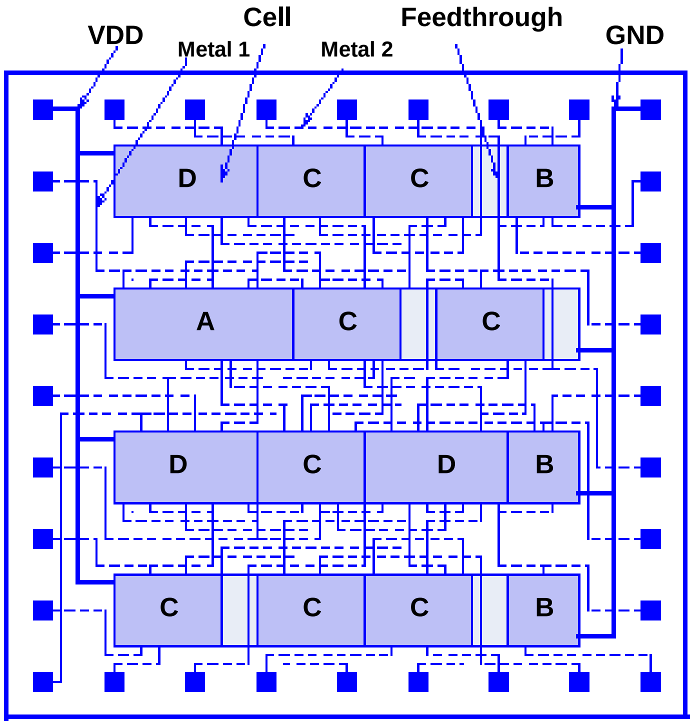
Design Styles

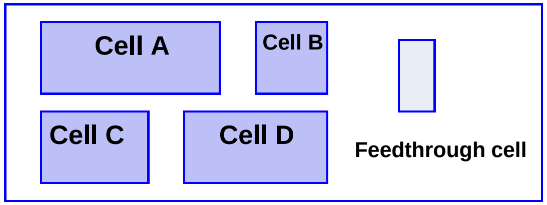
- ❑ Full Custom Design Style
  - – Design every component from scratch
- ❑ Standard Cell Design Style
  - – Selects pre-designed cells (of same height)
- ❑ Gate Array Design Style
  - – Use arrays of prefabricated transistors
  - – Needs wiring customization to implement logic
- ❑ Field Programmable Gate Arrays (FPGA)
  - – Logic and interconnects are both prefabricated
  - – Program the logic functions and interconnects

Full Custom Design Style

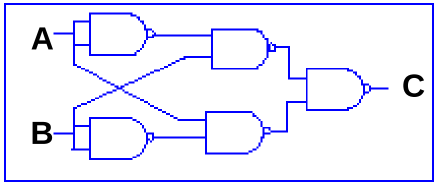


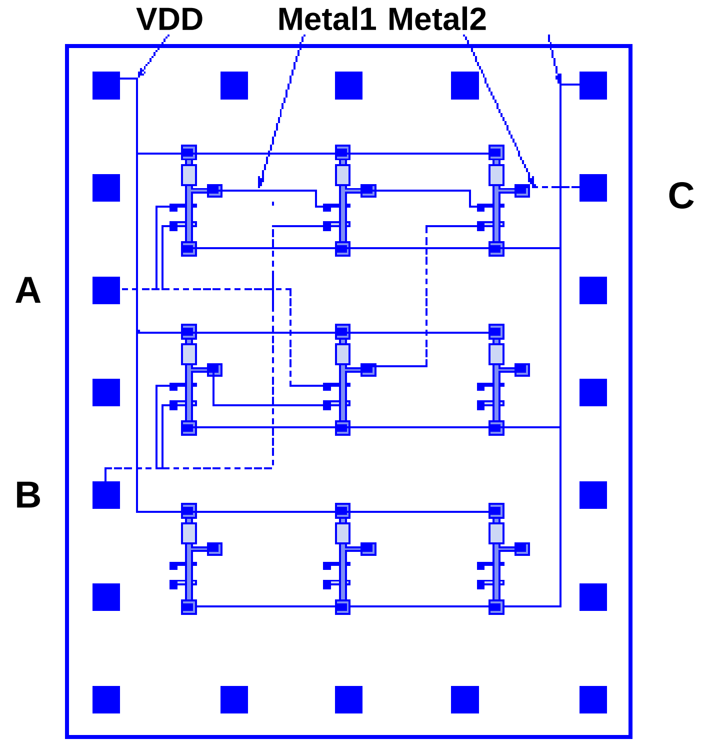
Standard Cell Design Style





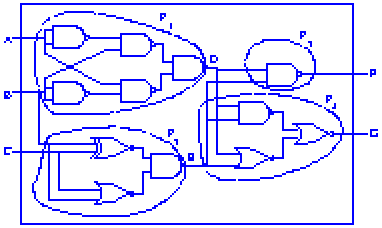
Gate Array Design Style





Structured ASICs are essentially gate array

FPGA Design Style



(a)

P_1

A	B	D
0	0	0
0	1	1
1	0	1
1	1	0

P_2

B	C	E
0	0	0
0	1	1
1	0	1
1	1	1

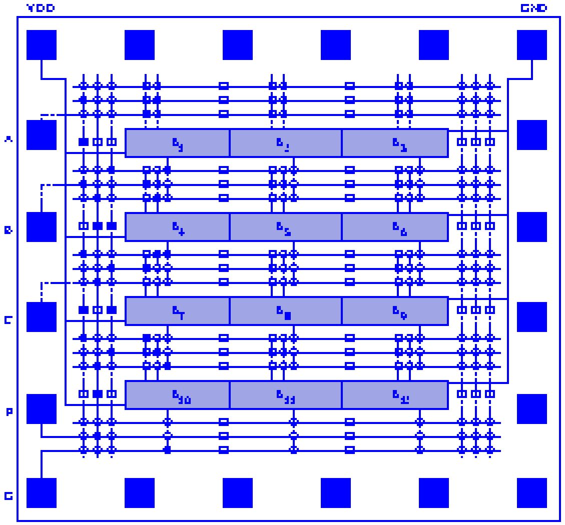
P_3

D	E	F
0	0	1
0	1	1
1	0	1
1	1	0

P_4

D	E	G
0	0	0
0	1	0
1	0	0
1	1	1

(b)



(c)

Comparisons of Design Styles

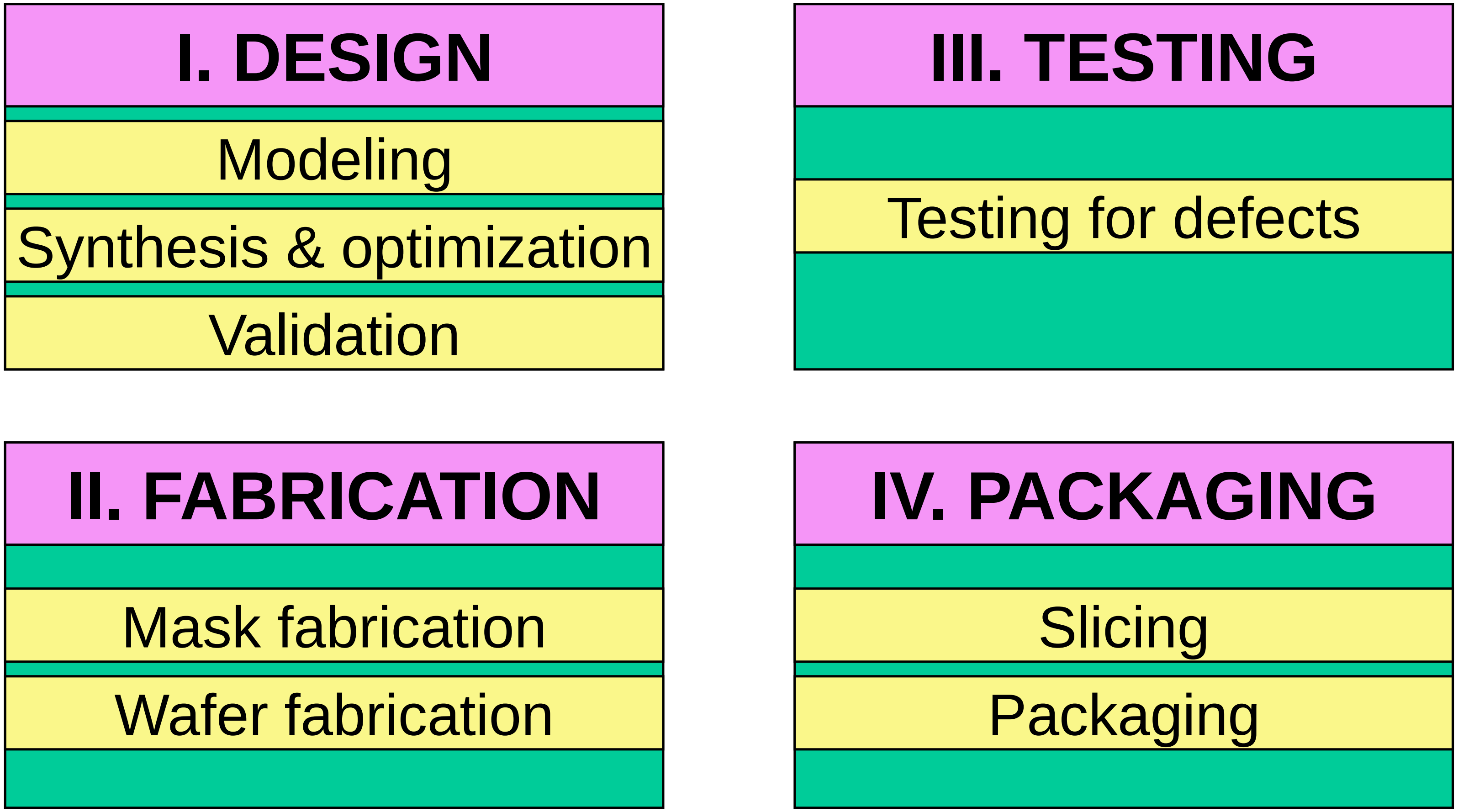
	Style			
	full-custom	standard cell	gate array	FPGA
cell size	variable	fixed height *	fixed	fixed
cell type	variable	variable	fixed	programmable
cell placement	variable	in row	fixed	fixed
interconnections	variable	variable	variable	programmable
design cost	high	medium	medium	low

* uneven height cells may be used

Comparisons of Design Styles

	style			
	full-custom	standard cell	gate array	FPGA
Area	compact	compact to moderate	moderate	large
Performance	high	high to moderate	moderate	low
Fabrication layers	all	all	routing layers	none

Four Stages in Creation of an IC



I. DESIGN

- Modeling
- Synthesis & optimization
- Validation

III. TESTING

- Testing for defects

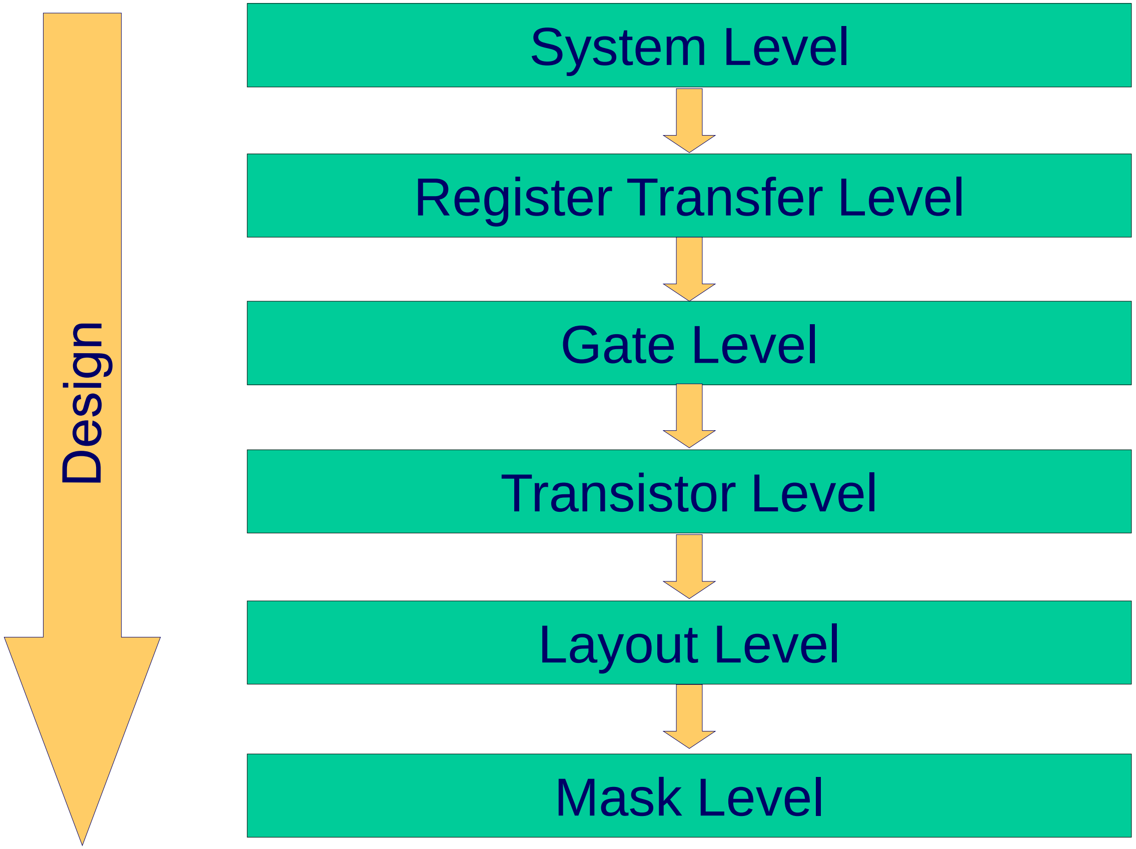
II. FABRICATION

- Mask fabrication
- Wafer fabrication

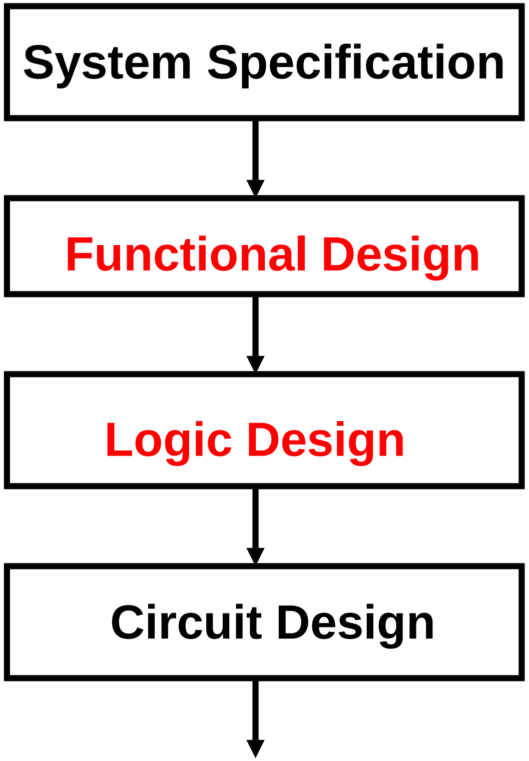
IV. PACKAGING

- Slicing
- Packaging

Design of Integrated Systems



VLSI Design Cycle

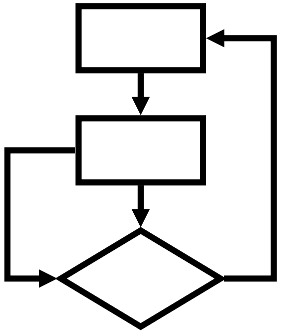


System Specification

Functional Design

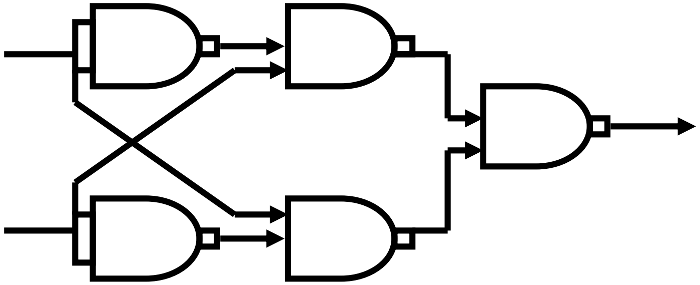
Logic Design

Circuit Design

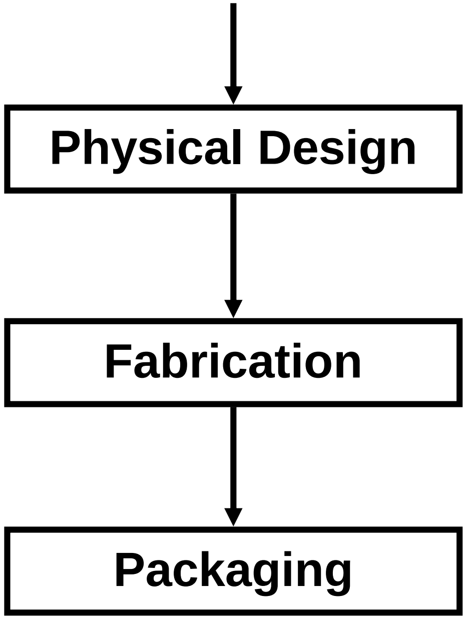


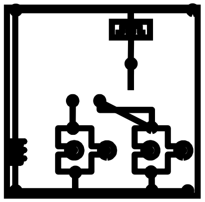
X=(AB*CD)+(A+D)+(A(B+C))

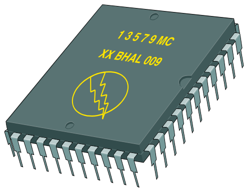
Y=(A(B+C))+AC+D+A(BC+D))



VLSI Design Cycle (cont.)







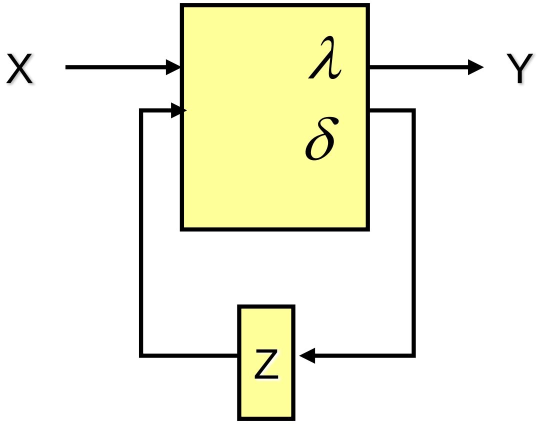
Focus of Synthesis

- CAD tools for synthesis and verification at logic-level of abstraction
- Theory behind: functions representation and manipulation
  - representation ⇔ data structures
  - manipulation ⇔ algorithms
- In-depth course:
  - You should be able create a small CAD-tool

Why Logic Level?

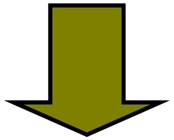
- ❑ Logic-level synthesis is the core of today's CAD flows for IC and system design
  - course covers many algorithms that are used in a broad range of CAD tools
  - basis for other optimization techniques
  - basis for functional verification techniques
- ❑ Most algorithms are computationally hard
  - covered algorithms and flows are good example for approaching hard algorithmic problems
  - course covers theory as well as implementation details
  - demonstrates an engineering approaches based on theoretical solid but also practical solutions
    - very few research areas can offer this combination

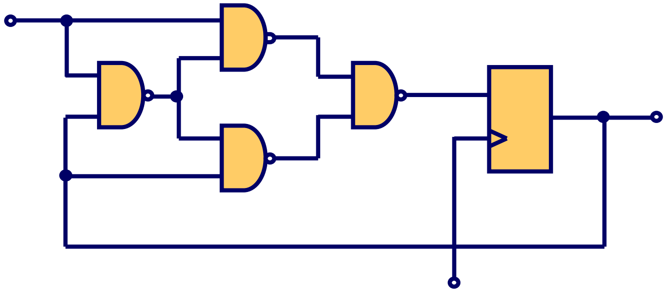
What is Logic Synthesis?



Given: Finite-State Machine F(X,Y,Z,λ,δ) where:

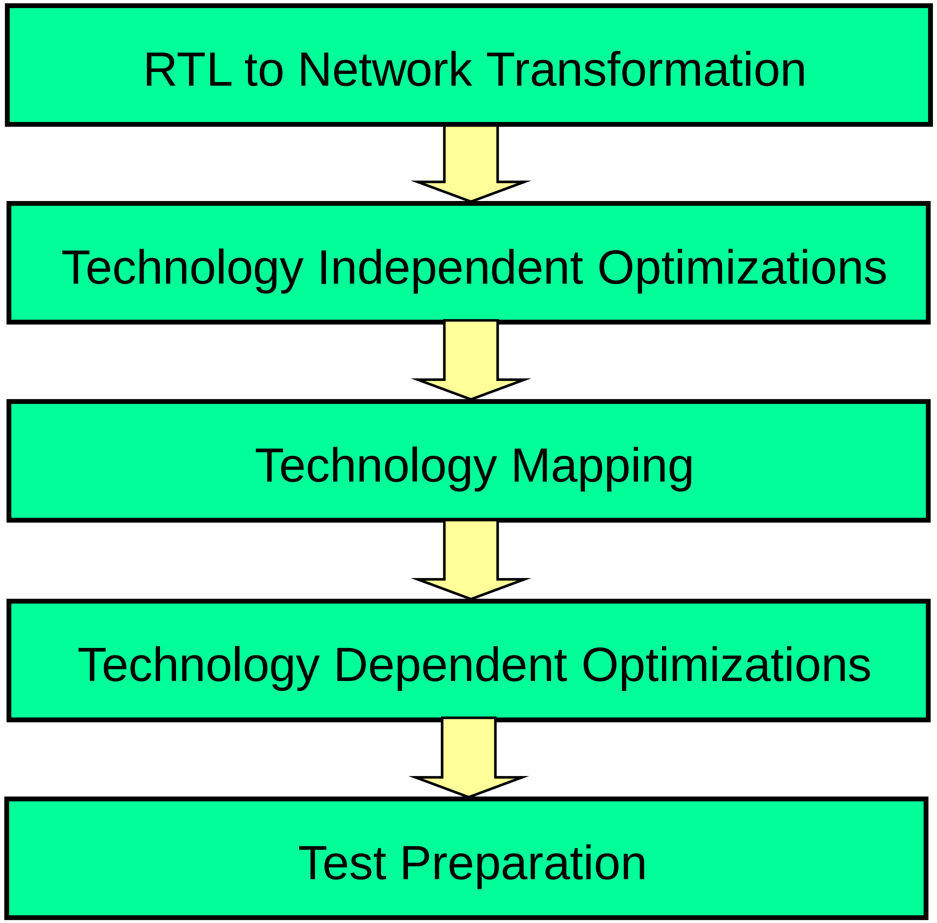
X: Input alphabet
Y: Output alphabet
Z: Set of internal states
λ : X × Z → Y (output function)
δ : X × Z → Z (next state function)





Target: Circuit C(G, W) where
G: set of circuit components g (Boolean gates flip-flops, etc}
W: set of wires connecting G

Typical Logic Synthesis Scenario


RTL to Network Transformation
Technology Independent Optimizations
Technology Mapping
Technology Dependent Optimizations
Test Preparation

- Read Verilog/VHDL
- Control/data flow analysis

- Basic logic restructuring
- Crude measures for goals

- Use logic gates from target cell library

- Timing optimization
- Physically driven optimizations

- Improve testability
- Test logic insertion

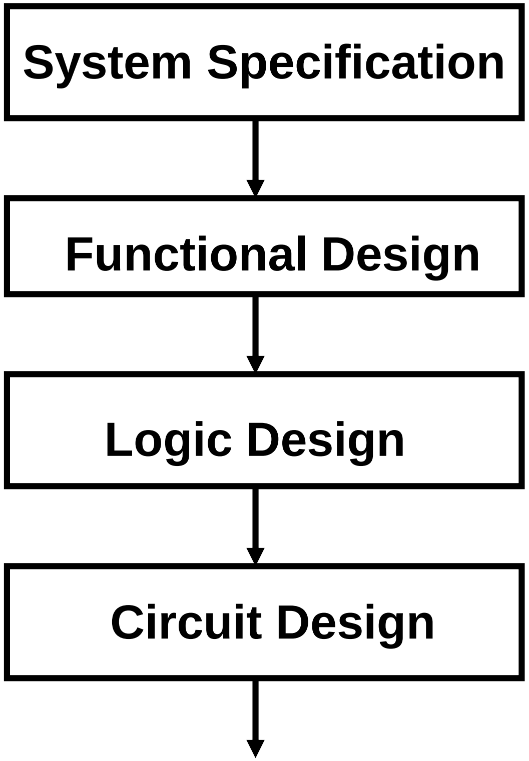
Objective Function for Synthesis

- Minimize area
  - in terms of literal count, cell count, register count, etc.
- Minimize power
  - in terms of switching activity in individual gates, blocks, etc.
- Maximize performance
  - in terms of maximal clock frequency of synchronous systems, throughput for asynchronous systems
- Any combination of the above
  - combined with different weights
  - formulated as a constraint problem
    - Ex: minimize area for a clock speed > 300MHz

Constraints on Synthesis

- ❑ Given implementation style:
  - – two-level implementation (PLA)
  - – multi-level logic
  - – FPGAs
- ❑ Given performance requirements
  - – minimal clock speed requirement
- ❑ Given cell library
  - – set of cells in standard cell library
  - – fan-out constraints (maximum number of gates connected to another gate)

VLSI Design Cycle

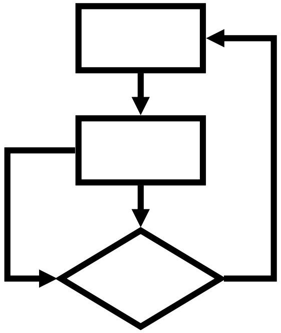


System Specification

Functional Design

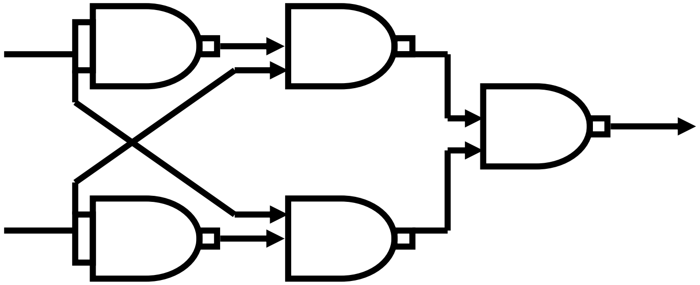
Logic Design

Circuit Design

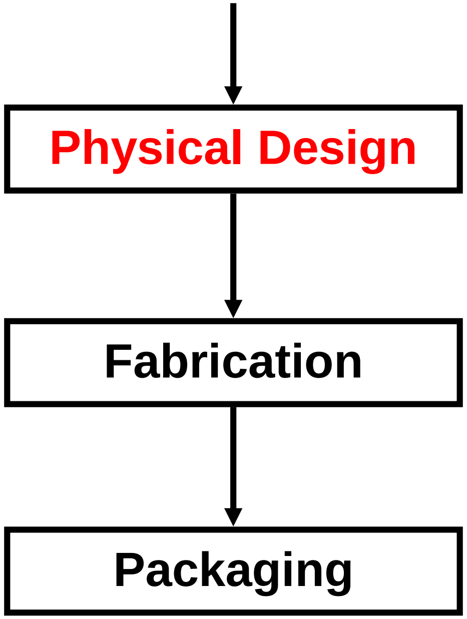


X=(AB*CD)+(A+D)+(A(B+C))

Y=(A(B+C))+AC+D+A(BC+D))

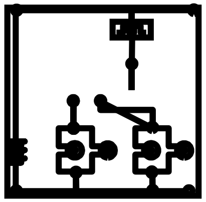


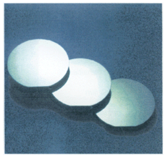
VLSI Design Cycle (cont.)

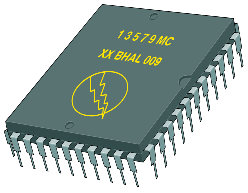

Physical Design

Fabrication

Packaging







Physical Design

- Physical design converts a circuit description into a geometric description. This description is used to manufacture a chip. The physical design cycle consists of
  - Partitioning
  - Floorplanning and Placement
  - Routing
  - Compaction

Traditional View

Physical Design Process

Design Steps:

Partition & Clustering

Floorplan & Placement

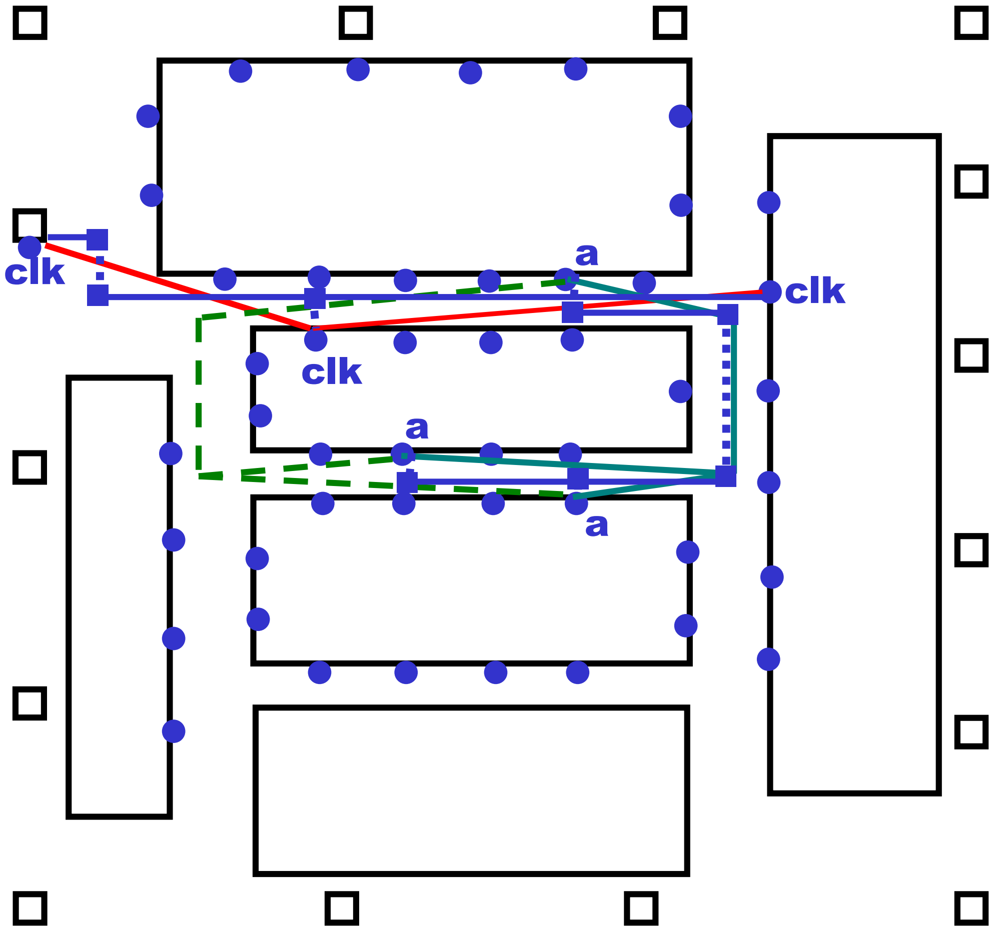
Pin Assignment

Global Routing
Global Routing

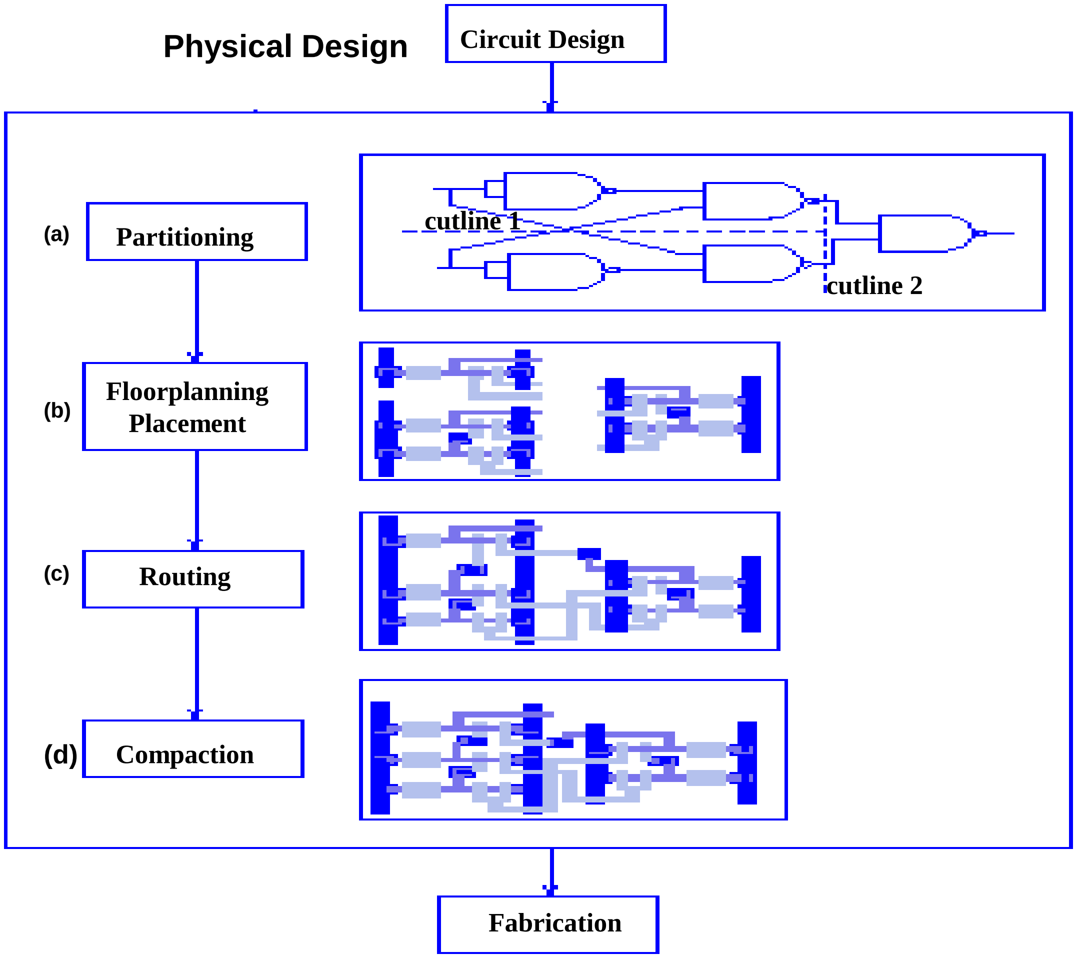
Detailed Routing

Methodology:

Divide-and-Conquer



Physical Design Cycle



Physical Design

Circuit Design

(a) Partitioning

(b) Floorplanning Placement

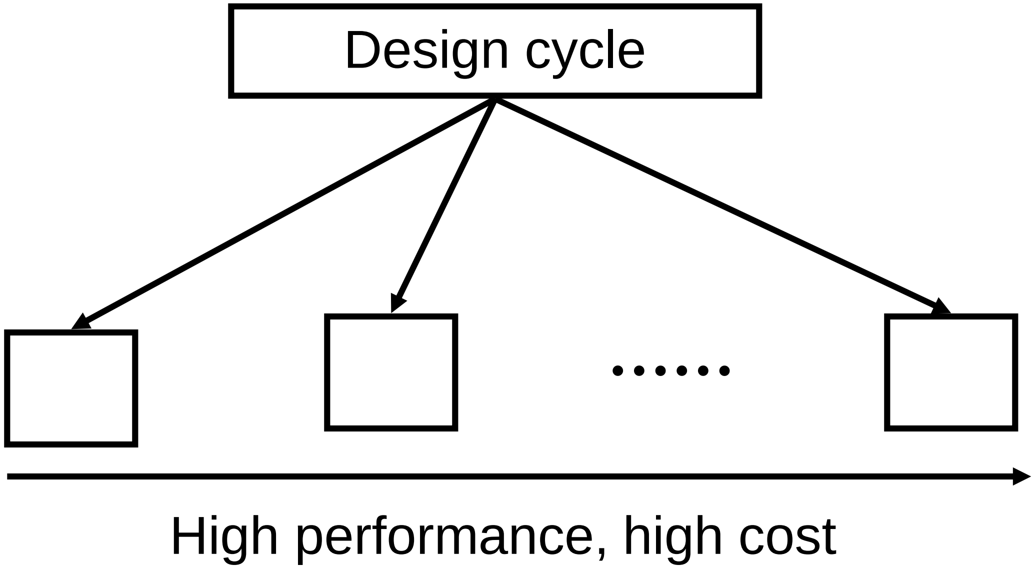
(c) Routing

(d) Compaction

Fabrication

Complexities of Physical Design

- ❑ More than 100 million transistors
- ❑ Performance driven designs
- ❑ Power-constrained designs
- ❑ Time-to-Market



History 101 of Physical Design

- ❑ Born in early 60’s (board layout)
- ❑ Passed teenage in 70’s (standard cell place and route)
- ❑ Entered early adulthood in 80’s (over-the-cell routing)
- ❑ Declared dead in late 80’s !!!
- ❑ Found alive and kicking in 90’s
- ❑ Physical Design (PD) has become a dominant force in overall design cycle,
  - – thanks to the deep submicron scaling
  - – expand vertically with logic synthesis and interconnect optimization, analysis…. => Design closure!

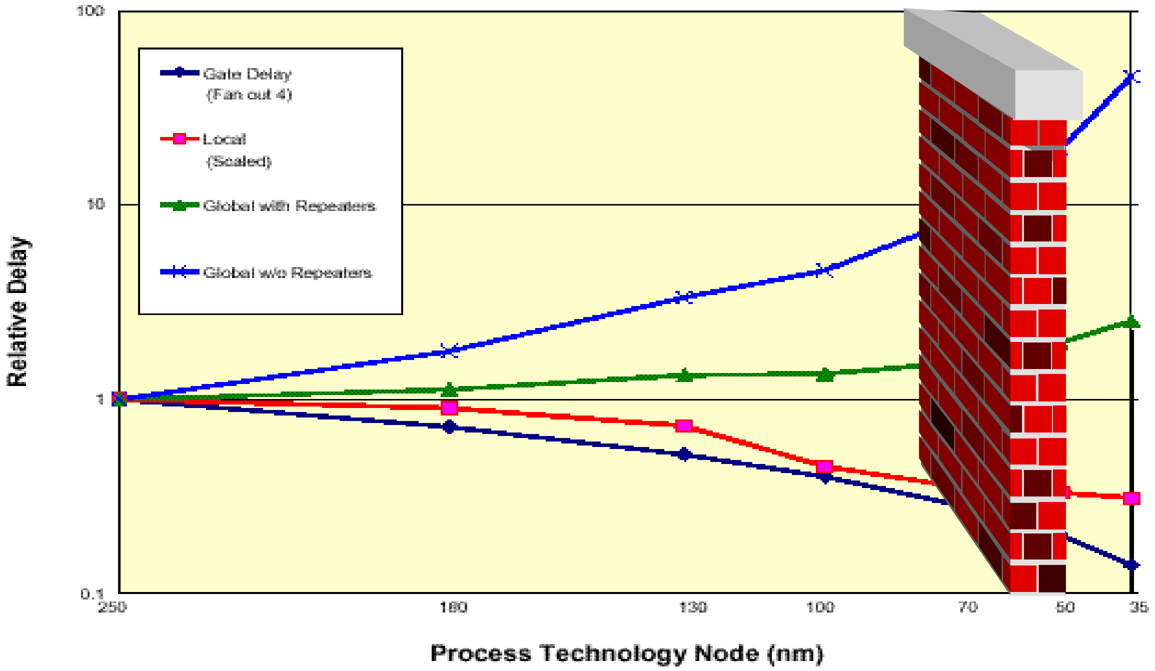
Why Physical Design is still HOT?

- ❑ Many existing solutions are still very suboptimal
  - – Ex: placement
- ❑ Interconnect dominates
  - – No physical layout, no accurate interconnect
- ❑ More new physical and manufacturing effects pop up
  - – Crosstalk noise, …
  - – OPC (manufacturability), etc.
- ❑ More vertical integration needed
- ❑ Physical design is the KEY linking step between higher level planning/optimization and lower level modeling

Moore's Law

- The minimum transistor feature size decreases by 0.7X every three years (Electronics Magazine, Vol. 38, April 1965)
- Consequences of smaller transistors:
  - Faster transistor switching
  - More transistors per chip
- Almost true for 50 years!
- And it has been facing tremendous challenges now
  - Need smarter and more powerful CAD tools than ever

Technology Trend and Challenges



Source: ITRS’03

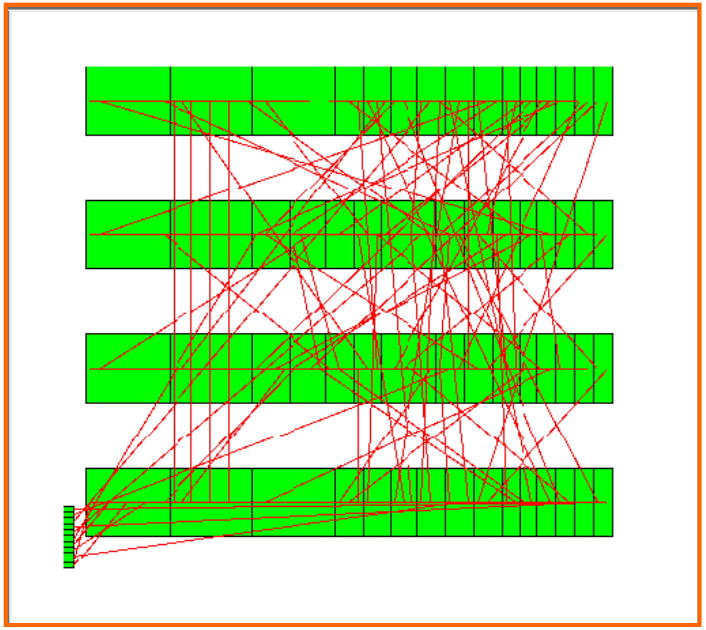
- ❑ Interconnect determines the overall performance
- ❑ In addition: noise, power => Design closure
- ❑ Furthermore: manufacturability => Manufacturing closure

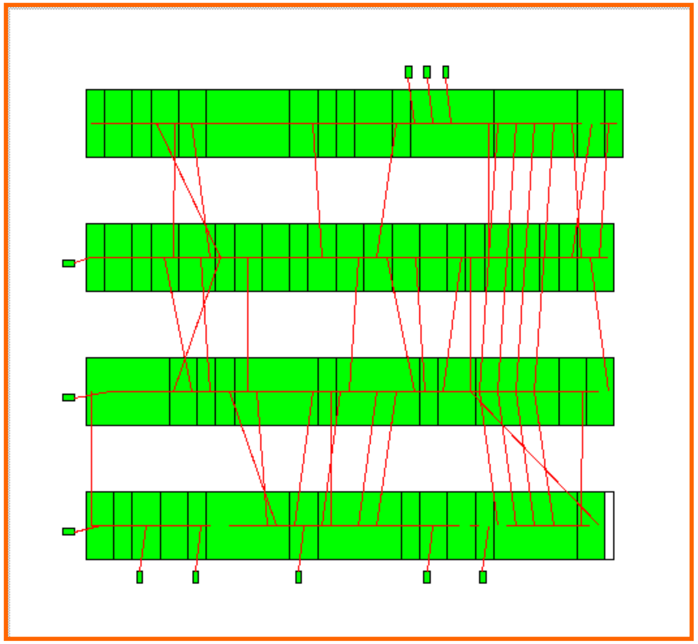
Placement Challenge

- Placement, to large extend, determines the overall interconnect
- If it sucks, no matter how well you interconnect optimization engine works, the design will suck
- Placement is a very old problem, but got renewed interest
  - Mixed-size (large macro blocks and small standard cells)
  - Optimality study shows that placement still a bottleneck
  - Not even to mention performance driven, and coupled with buffering, interconnect optimizations, and so on

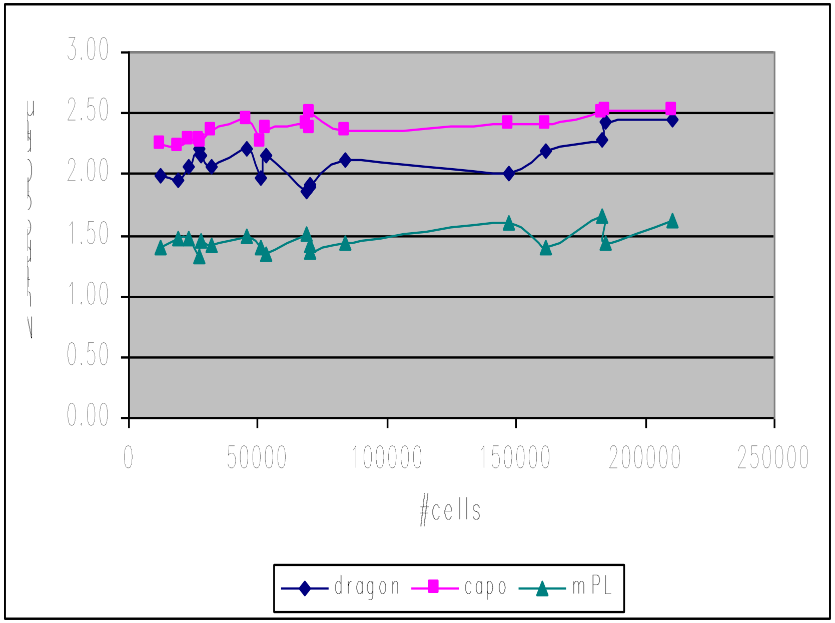
VLSI Global Placement Examples

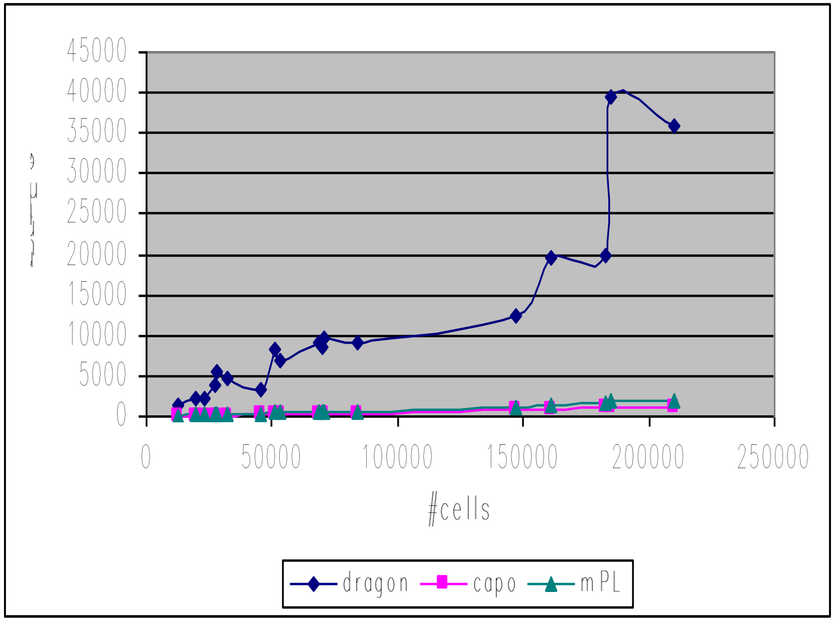
- Which one of the two placement results is better?





Comparison with Optimal



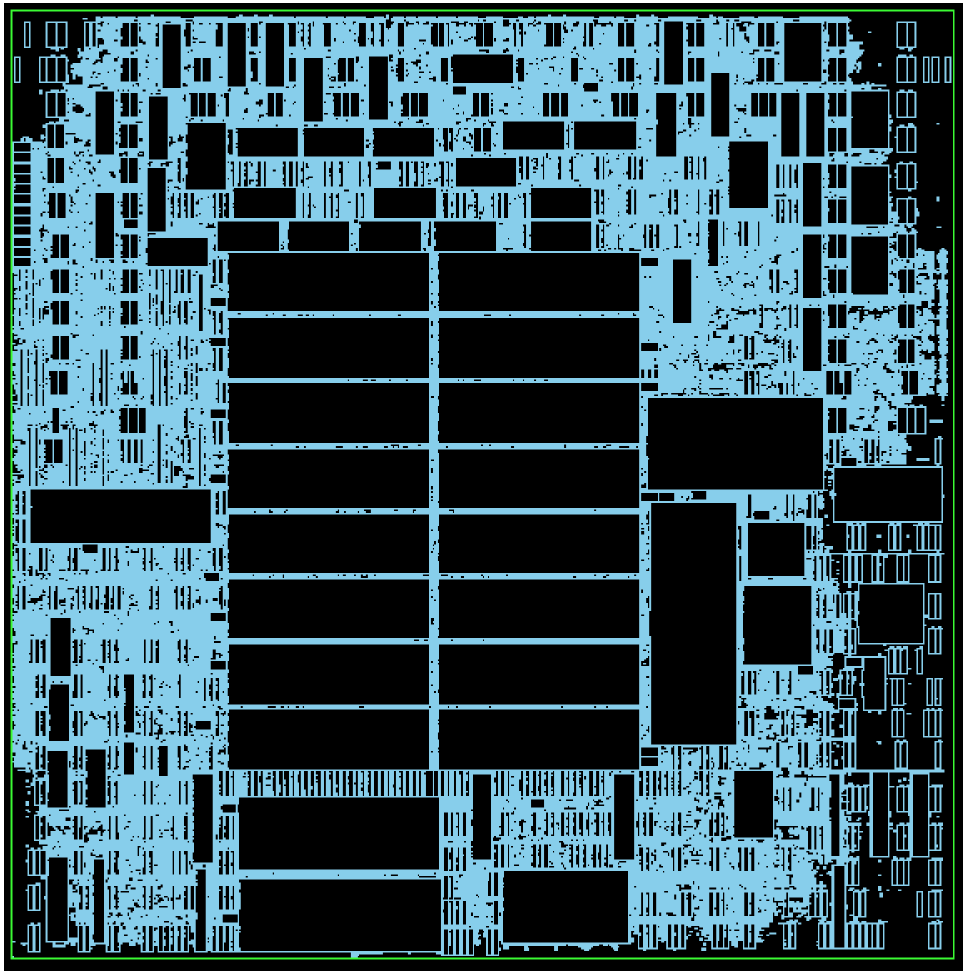


- Capo: Based on recursive min-cut (UCLA-UMich)
- Dragon: Recursive min-cut + SA refinement at each level (NWU-UCLA)
- mPL: multi-level placer (UCLA)

There is significant room for improvement in placement algorithms: existing algorithms are 50-150% away from optimal! [PEKO, 2004]

FloorPlacer (Mix-mode Placement)

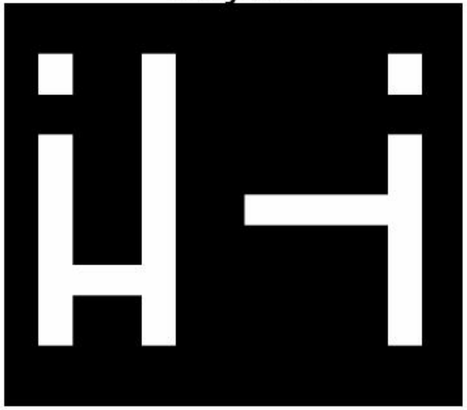
- ❑ Many macros
- ❑ data paths + dust logic
- ❑ I/O constraint
  (area I/O or wirebond)



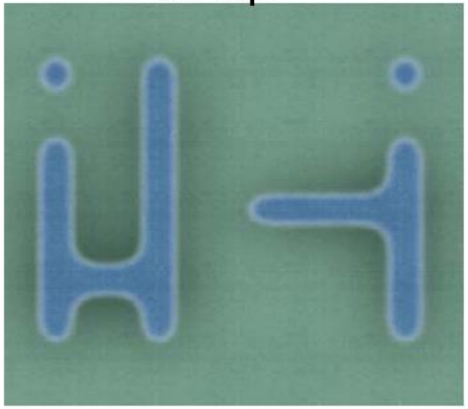
(source: IBM)

Lithography

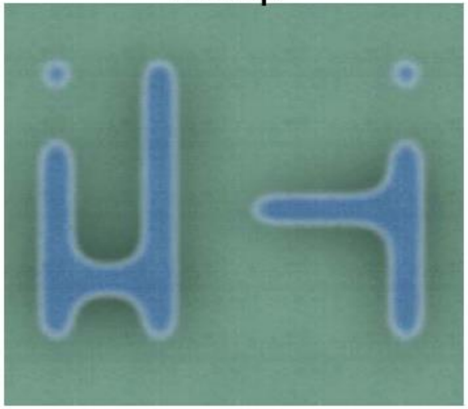
Layout



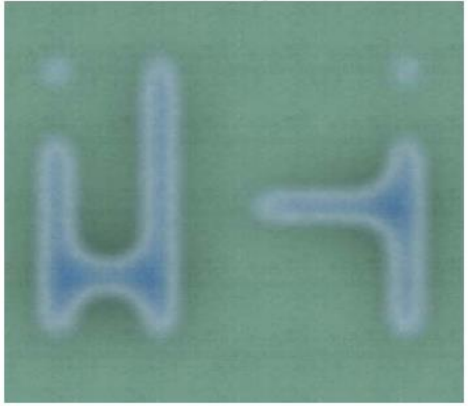
0.25μ



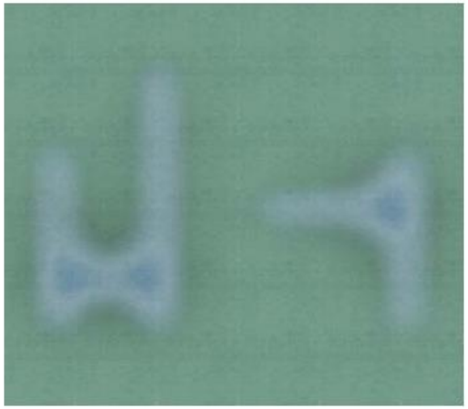
0.18μ



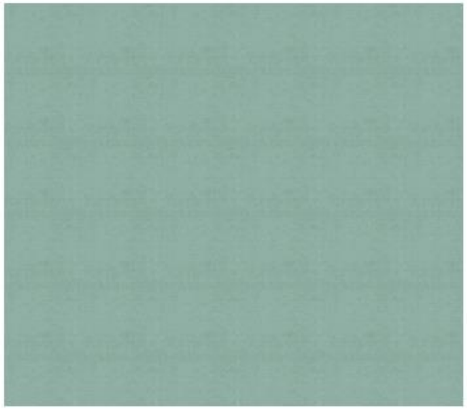
0.13μ



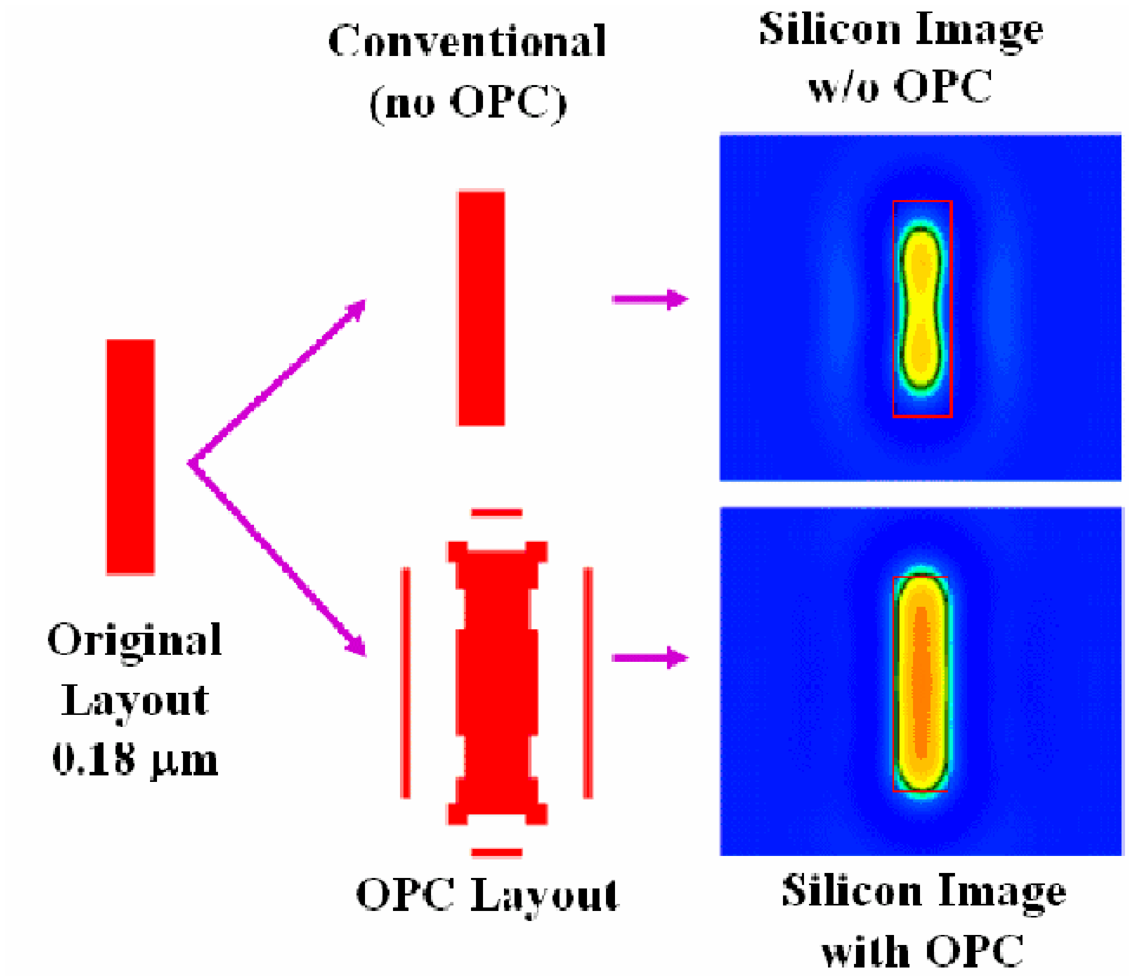
90-nm



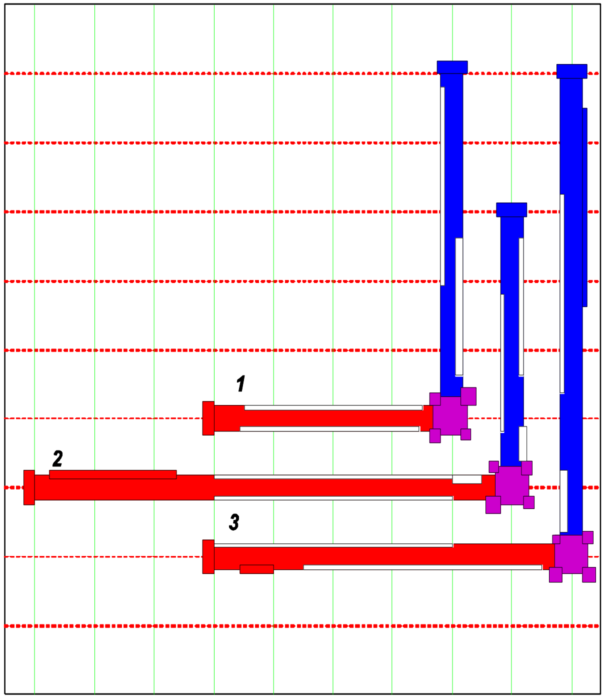
65-nm

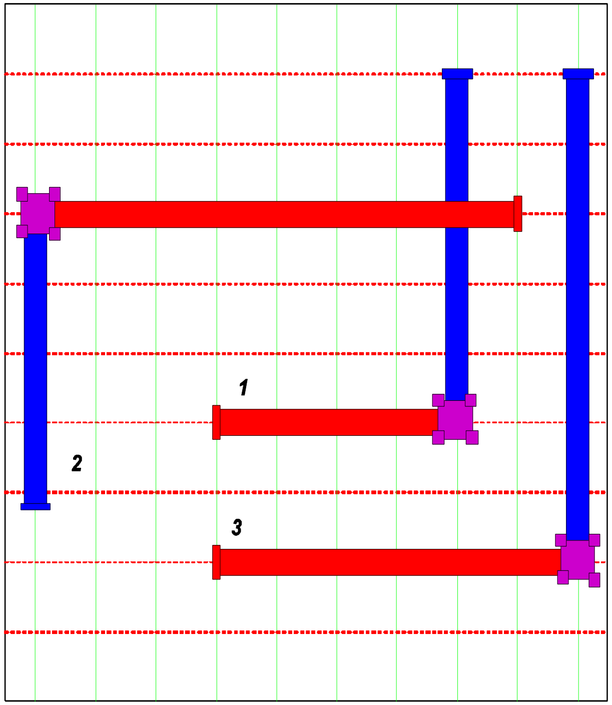


Optical Proximity Correction (OPC)



OPC-Aware Routing





More OPC friendly

We Need Algorithms

- ❑ To optimize design among different objectives, area, power, performance, and etc.
- ❑ Fundamental questions: How to do it smartly?
- ❑ Definition of algorithm in a board sense: A step-by-step procedure for solving a problem. Examples:
  - – Cooking a dish
  - – Making a phone call
  - – Sorting a hand of cards
- ❑ Definition for computational problem: A well-defined computational procedure that takes some value as input and produces some value as output

Computational Complexity

- Computational complexity is an abstract measure of the time and space necessary to execute an algorithm as function of its input size
  - The input is the graph G(V,E)
    - input size = $|V|$ and $|E|$
  - The input is the truth table of an n-variable Boolean function
    - input size = 2^n

Time and Space Complexity

- ❑ Time complexity is expressed in elementary computational steps
  - – example: addition (or multiplication, or value assignment etc.) is one step
  - – normally, by "most efficient" algorithm we mean the fastest
- ❑ Space complexity is expressed in memory locations
  - – e.g. in bits, bytes, words

Example: Selection Sort

- Input: An array of n numbers D[1]…D[n]
- Output: An array of n numbers E[1]…E[n] such that E[1] $\geq$E[2] $\geq$… $\geq$E[n]
- Algorithm:

```
For i from 1 to n do
    Select the largest remaining no. from D[1..n]
    Put that number into E[i]
```

Some Algorithm Design Techniques

- Greedy
- Divide and Conquer
- Dynamic Programming
- Network Flow
- Mathematical Programming (ex: linear programming, integer linear programming, quadratic programming, and etc.)

Reduction

- ❑ Idea: If we can solve problem A, and if problem B can be transformed into an instance of problem A, then we can solve problem B by reducing problem B to problem A and then solve the corresponding problem A.
- ❑ Example:
  - – Problem A: Sorting
  - – Problem B: Given n numbers, find the i-th largest numbers.

Analysis of Algorithm

- There can be many different algorithms to solve the same problem
- Need some way to compare 2 algorithms
- Usually the run time is the criteria used
- However, difficult to compare since algorithms may be implemented in different machines, use different languages, etc.
- Also, run time is input-dependent. Which input to use?
- Big-O notation is used

Big-O Notation

- ❑ Consider run time for the worst input
  - – upper bound on run time
- ❑ Express run time as a function input size n
- ❑ Interested in the run time for large inputs
- ❑ Therefore, interested in the growth rate
- ❑ Ignore multiplicative constant
- ❑ Ignore lower order terms

Big-O Notation

- $f = O(g)$, if two constants n_0 and K can be found such that for all $n \geq n_0$:

$$f(n) \leq K \cdot g(n)$$

- Examples:

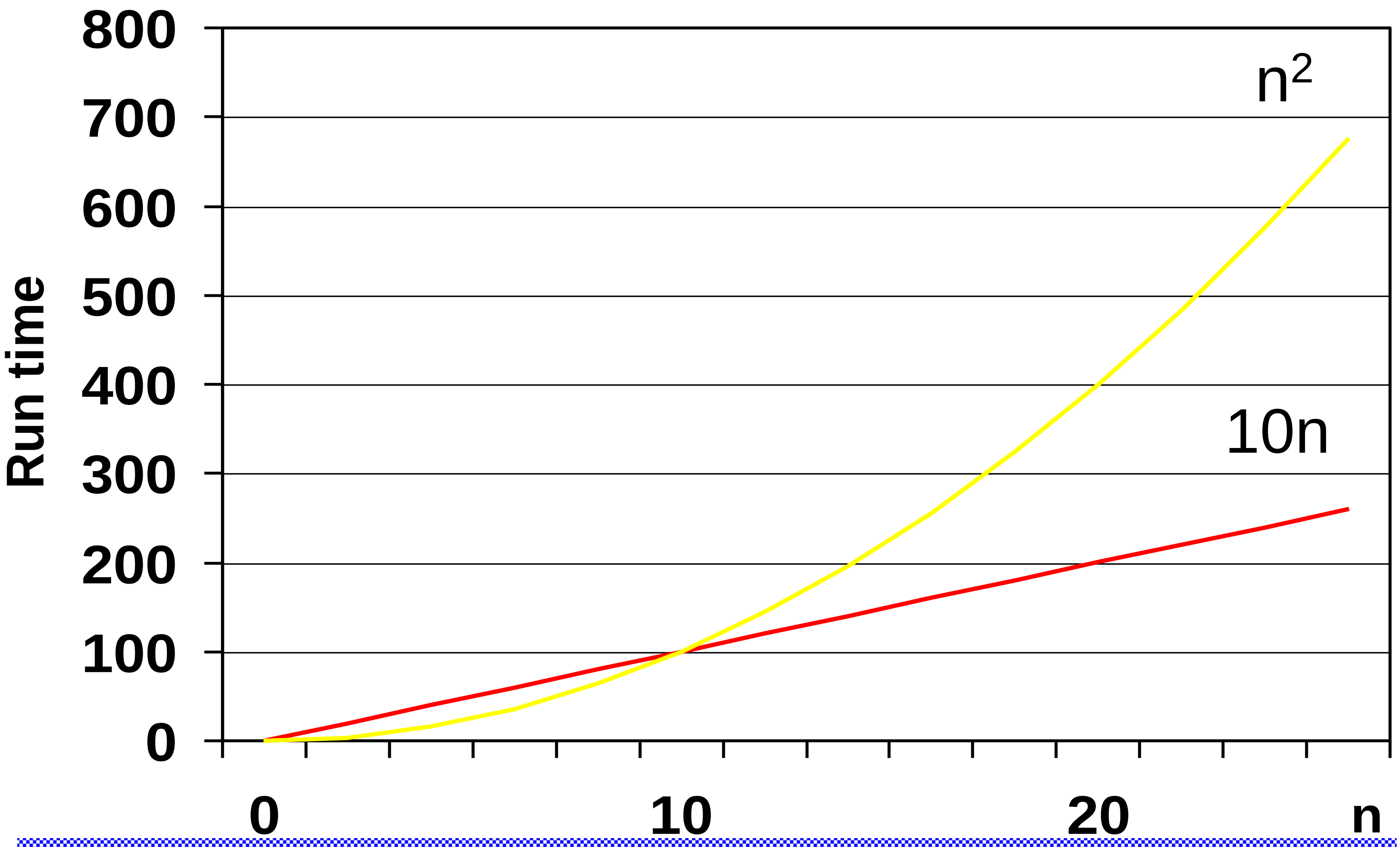
$2n^2 = O(n^2)$

$2n^2 + 3n + 1 = O(n^2)$

$n^{1.1} + 10000000000n$ is $O(n^{1.1})$

$n^{1.1} = O(n^2)$

Effect of Multiplicative Constant



Growth Rates of some Functions

$$O(\log n) < O(\log^2 n) < O(\sqrt{n}) < O(n)$$
$$< O(n \log n) < O(n \log^2 n) < O(n^{1.5}) < O(n^2)$$
$$< O(n^3) < O(n^4)$$

Polynomial Functions

$$O(n^c) = O(2^{c \log n}) \quad \text{for any constant } c$$
$$< O(n^{\log n}) = O(2^{\log^2 n})$$
$$< O(2^n) < O(3^n) < O(4^n)$$
$$< O(n!) < O(n^n)$$

Exponential Functions

Problem of Exponential Function

- ❑ Consider 2^n, value doubled when n is increased by 1.

n	2^n	$1\mu s \times 2^n$
10	10^3	0.001 s
20	10^6	1 s
30	10^9	16.7 mins
40	10^{12}	11.6 days
50	10^{15}	31.7 years
60	10^{18}	31710 years

- ❑ If you borrow $10 from a credit card with APR 18%, after 40 yrs, you will owe $12700!

Exponential Time Complexity

- An algorithm has an exponential time complexity if its execution time is given by the formula

$$\text{execution time} = k_1 \cdot (k_2)^n$$

where *n* is the size of the input data and k_1, k_2 are constants

Exponential Time Complexity

- The execution time grows so fast that even the fastest computers cannot solve problems of practical sizes in a reasonable time
- The problem is called intractable if the best algorithm known to solve this problem requires exponential time
- Many CAD problems are intractable

Optimization and Decision Problems

- ❑ Optimization problems ask to find a solution which has minimum "cost" among all other solutions
  - – Ex: find a minimal sum-of-product expression for the given function
- ❑ Decision problems have only two possible solutions: “yes” or “no”
  - – Ex: can the given function be represented as a sum of 3 products?

The Satisfiability Problem

❑ PROBLEM DEFINITION:

Given a product-of-sum Boolean expression C of n variables which consists of m sums, is there a satisfying truth assignment for the variables?

❑ Example: n=4, m=2

$C = (x_1 + x_2 + x_4)(x_1 + x_2 + x'_3)$

the answer is “yes”, if (1101) then C = 1

❑ Solver

- http://www.princeton.edu/~chaff/zchaff.html
- http://people.sc.fsu.edu/~burkardt/data/cnf/cnf.html

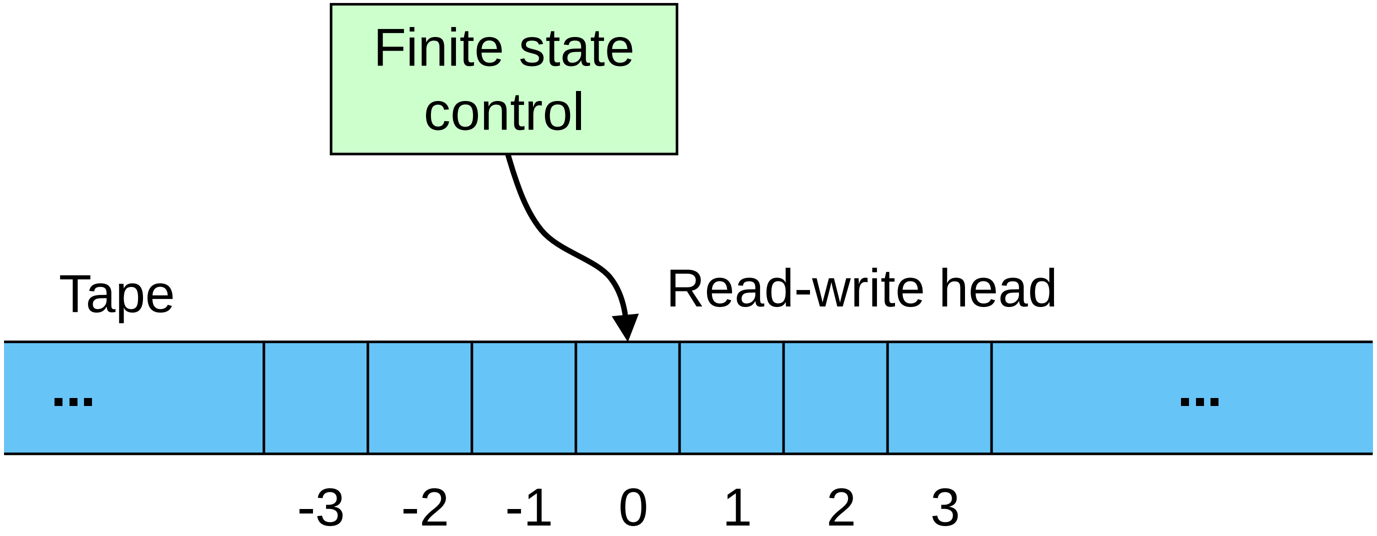
Complexity Classes

- Class P contains those problems that can be solved in polynomial time (the number of computation steps necessary can be expressed as a polynomial of the input size n).
- The computer concerned is a deterministic Turing machine

Deterministic Turing Machine

- Turing machine is a mathematical model of a universal computer
- Any computation that needs polynomial time on a Turing machine can also be performed in polynomial time on any other machine
- Deterministic means that each step in a computation is predictable

Deterministic One-tape Turing Machine

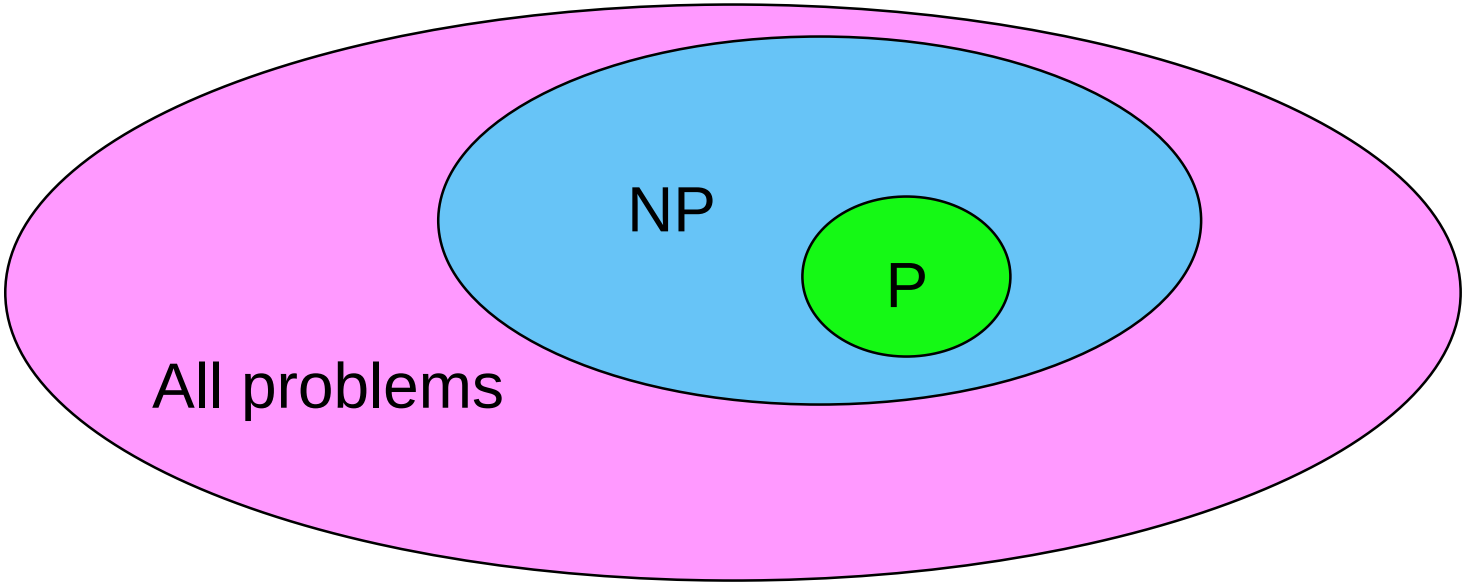


Non-deterministic Turing Machine

- If solution checking for some problem can be done in polynomial time on a deterministic machine, then the problem can be solved in polynomial time on a non-deterministic Turing machine
- non-deterministic - 2 stages:
  - make a guess what the solution is
  - check whether the guess is correct

NP-class

- Class NP contains those problems that can be solved in polynomial time on a non-deterministic Turing machine



NP-complete Problems

- ❑ A question which is still not answered:

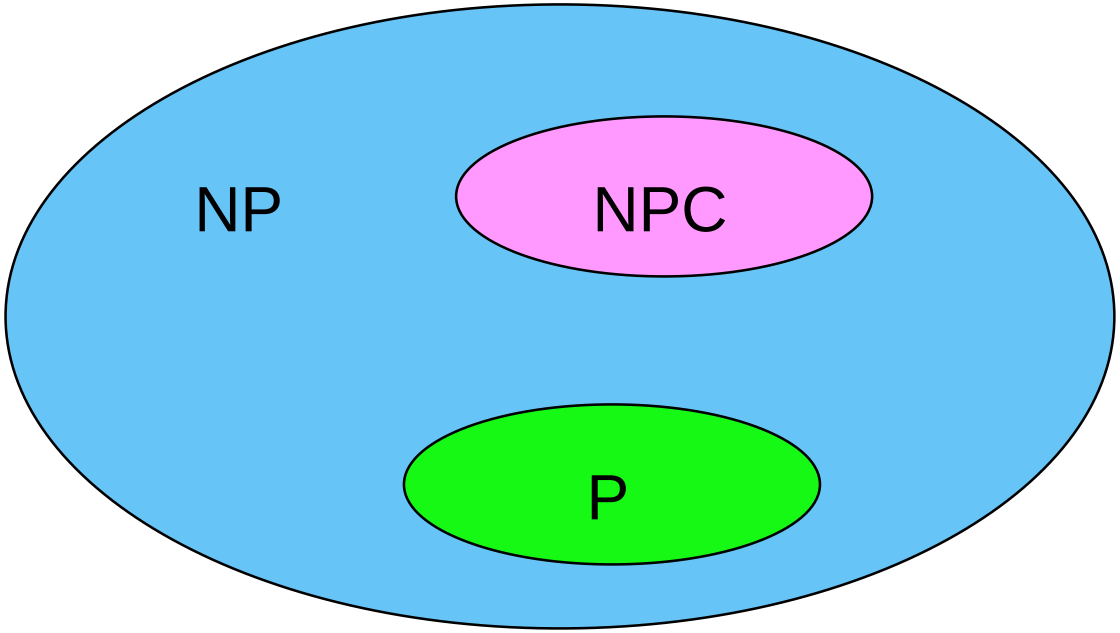
$$P \subset NP \text{ or } P \neq NP$$

- ❑ There is a strong belief that $P \neq NP$, due to the existence of NP-complete (NPC) problems (NPC)
  - – all NPC problems in have the same degree of difficulty: if one of them could be solved in polynomial time, all of them would have a polynomial time solution.

NP-complete Problems

- ❑ A problem is NP-complete if and only if
  - – It is in NP
  - – Some known NP-complete problem can be transformed to it in polynomial time
- ❑ Cook’s theorem:
  - – SATISFIABILITY is NP-complete

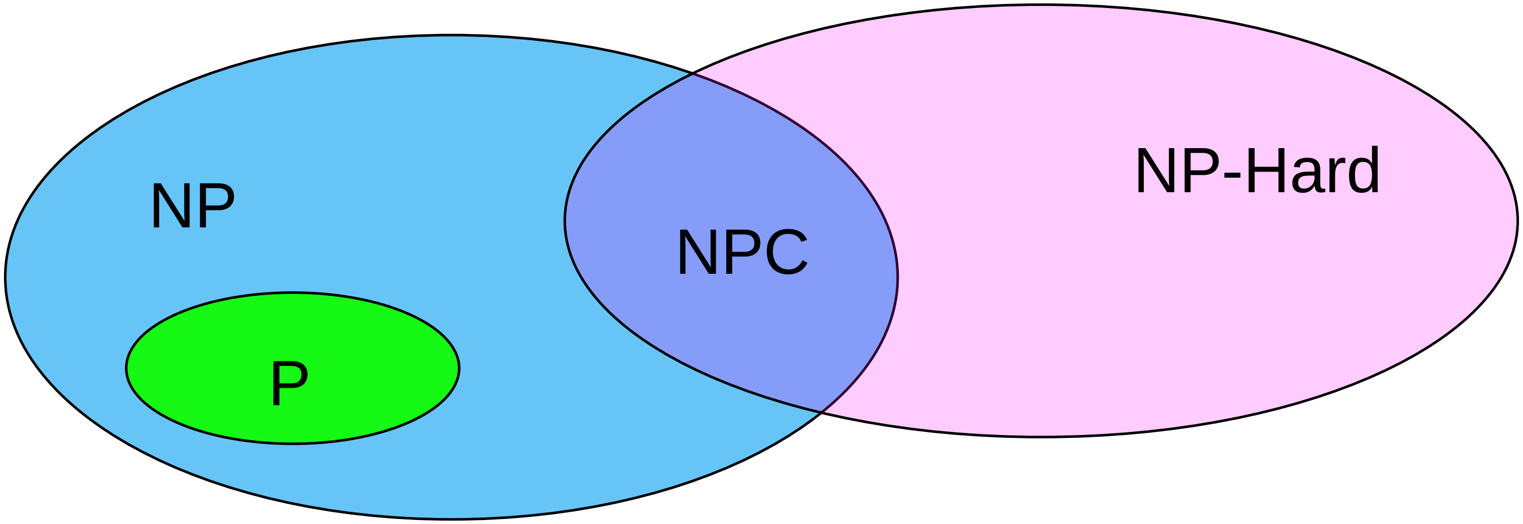
World of NP, Assuming P ≠ NP



NP-hard Problems

- ❑ Any decision problem (inside or outside of NP) to which we can transform an NP-complete problem to it in polynomial time will have a property that it cannot be solved in polynomial time, unless P = NP
- ❑ Such problems are called NP-hard
  - – “as hard as the NP-complete problems”

NP-Hard, NP, and NPC



Practical Consequences

- ❑ Many problems in CAD for VLSI are NP-complete or NP-hard. Therefore:
  - – Exact solutions to such problems can only be found when the problem size is small
  - – One should otherwise be satisfied with sub-optimal solutions found by:
    - Approximation algorithms: they can guarantee a solution within e.g. 20% of the optimum
    - Heuristics: nothing can be said a priori about the quality of the solution (experience-based)

Example

- Tractable and intractable problems can be very similar:
  - the SHORTEST-PATH problem for undirected graphs is in P
  - the LONGEST-PATH problem for undirected graphs is in NP-complete

Examples of NP-complete Problems

- ❑ Clique:
  - – Instance: graph G = (V,E), positive integer K ≤ |V|
  - – Question: does G contain a clique of size K or more?
- ❑ Minimum cover
  - – Instance: collection C of subsets of a finite set S, positive integer K ≤ |C|
  - – Question: does G contain a cover for S of size K or less?

Brief Summary

- The class NP-complete is a set of problems which we believe there is no polynomial time algorithms
- Therefore, it is a class of hard problems
- NP-hard is another class of problems containing the class NP-complete
- If we know a problem is in NP-complete or NP-hard, there is nearly no hope to solve it efficiently
  - Perhaps, quantum computing could save us?

Solution Type of Algorithms

- Polynomial time algorithms
- Exponential time algorithms
- Special case algorithms
- Approximation algorithms
- Heuristic algorithms

Resource

- ❑ Official, lots of useful information for paper search and citation
  - – IEEE Xplore: http://ieeexplore.ieee.org/
  - – ACM Digital Library: http://www.acm.org/dl/
- ❑ GSRC Bookshelf
  - – http://vlsicad.eecs.umich.edu/BK/Slots/
- ❑ SIA: Semiconductor Industry Association
  - – https://www.semiconductors.org/
    - Int'l Technology Roadmap for Semiconductor
    - ITRS report: http://www.itrs.net/reports.html

Resource

- ❑ Please check the web site for a set of reference, papers and links (will be updated frequently)
  - *EE Times* (www.eetimes.com) for recent trend/development
- ❑ Unofficial, but lots of useful information for paper search and citation
  - http://citeseer.com/
  - Google Scholar
- ❑ MIT OpenCourseWare
  - If you need to make up some knowledge (Cormen’s algorithm)
  - http://ocw.mit.edu/index.html

VLSI/CAD Conferences

- ❑ DAC: Design Automation Conference
- ❑ ICCAD: Int'l Conference on Computer-Aided Design
- ❑ DATE: Design Automation and Test in Europe
- ❑ ASP-DAC: Asia and South Pacific DAC
- ❑ ISPD: Int'l Symposium on Physical Design
- ❑ ISCAS: Int'l Symposium on Circuits and Systems
- ❑ IWLS: Int'l Workshop on Logic Synthesis
- ❑ ISQED: Int'l Symposium on Quality Electronic Design
- ❑ ISLPED: Int'l Symposium on Low Power Electronics and Design

VLSI/CAD Conferences

- ❑ ISCA: Int'l Symposium on Computer Architecture
- ❑ HPCA: Int'l Symposium on High Performance Computer Architecture
- ❑ Micro: Int'l Symposium on Microarchitecture
- ❑ CODES+ISSS: Int'l Conference on Hardware/Software Codesign & System Synthesis
- ❑ CASES: Int'l Conf on Compilers, Architecture, & Synthesis for Embedded Systems

- ❑ Google: CS conference ranking

VLSI/CAD Related Journals

❑ IEEE TCAD

- – IEEE Transactions on CAD of Integrated Circuits and Systems

❑ ACM TODAES

- – ACM Transactions on Design Automation of Electronic Systems

❑ IEEE TVLSI

- – IEEE Transactions on VLSI Systems

❑ Integration, the VLSI Journal

❑ IEEE TCAS (I and II)

- – IEEE Transactions on Circuits and Systems