EE6094
CAD for VLSI Design

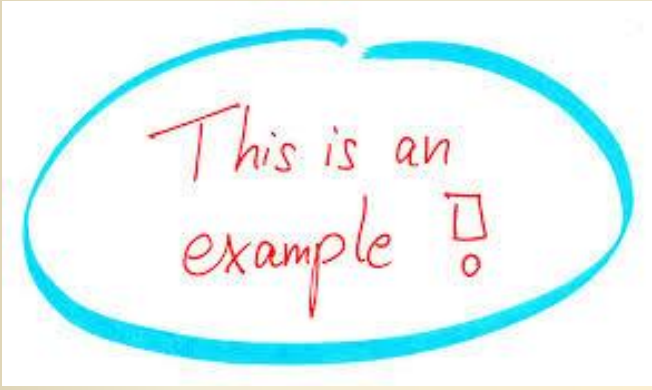
Chapter 2
Circuit Modeling

Spring 2021
Andy, Yu-Guang Chen
Assistant Professor, Department of EE
National Central University
andyygchen@ee.ncu.edu.tw

Outline

- An example of system design
- Circuit models and abstraction
- Synthesis
- SOC design flow Model

Let's Start with an Example…

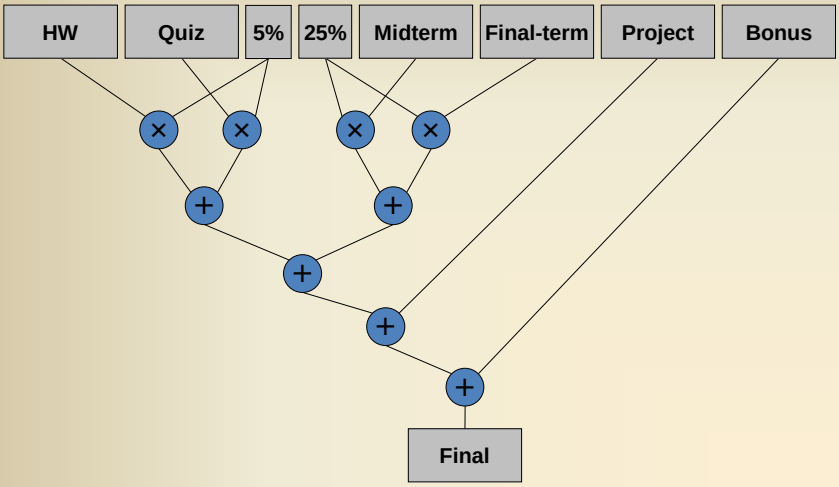


Design a Grading System

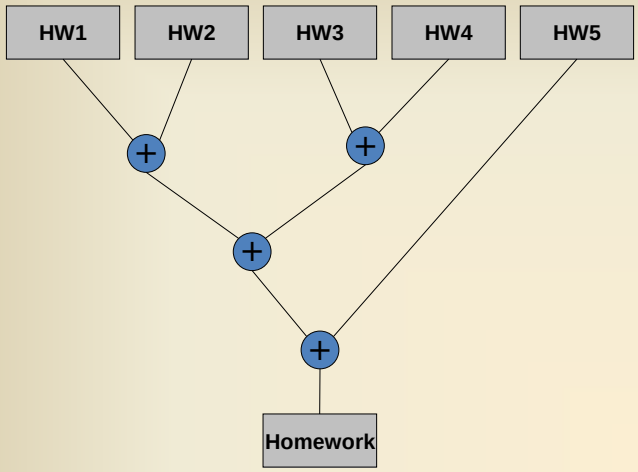
- 5 homework assignments (25%)
- 6 in-class quizzes (ignore lowest score) (25%)
- 1 midterm examination (25%)
- 1 final-term examination (25%)
- 1 extra term project
- Some bonus – class participation

- Can you **design a grading system**?

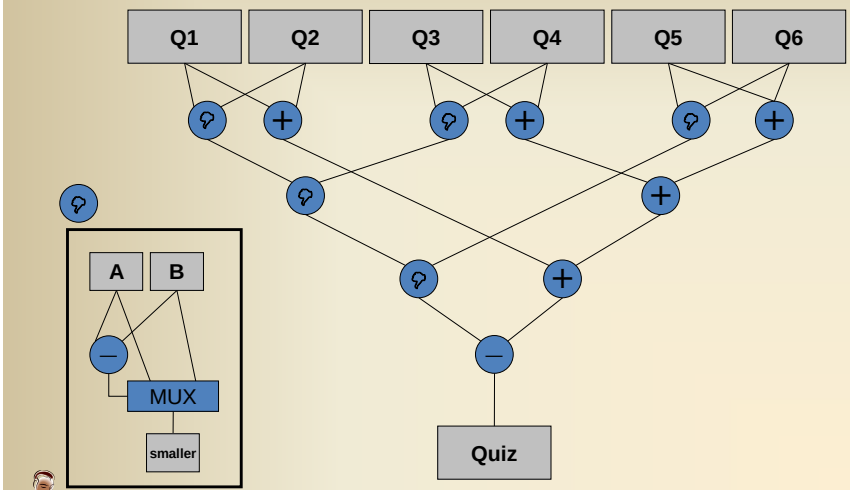
Overall Data Flow Graph (DFG)



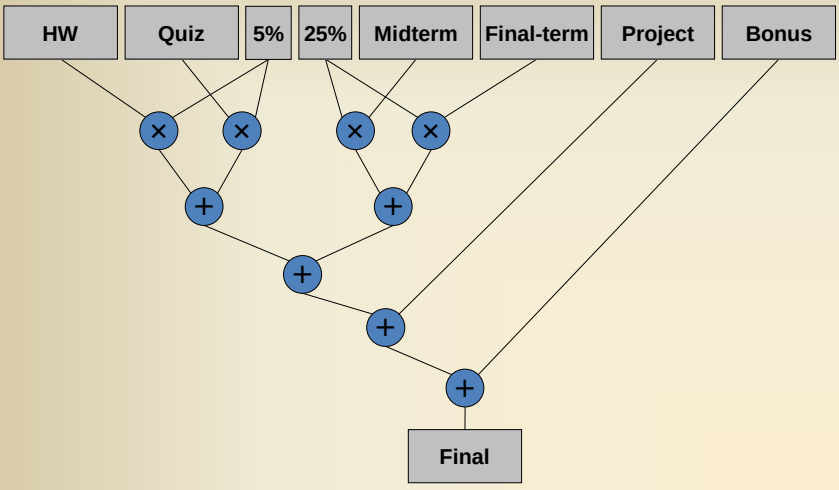
Homework DFG



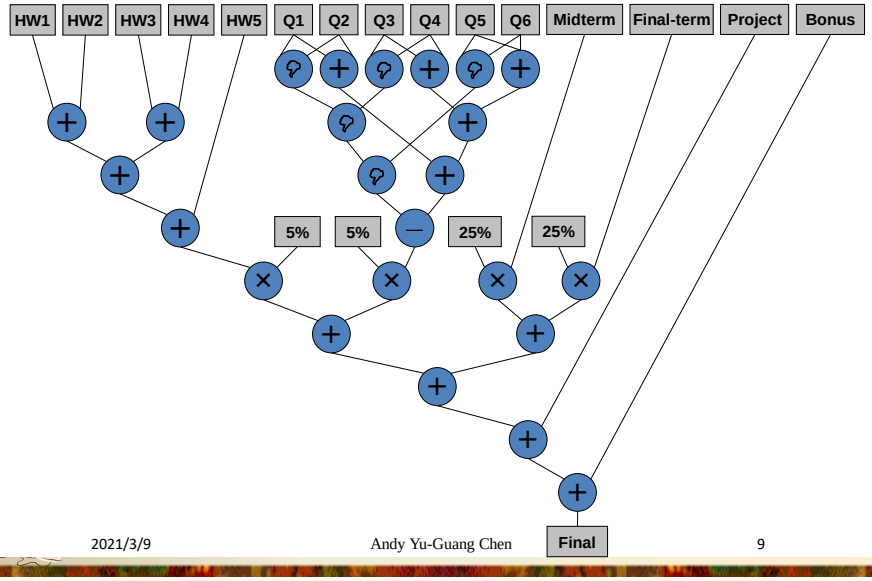
Quiz DFG



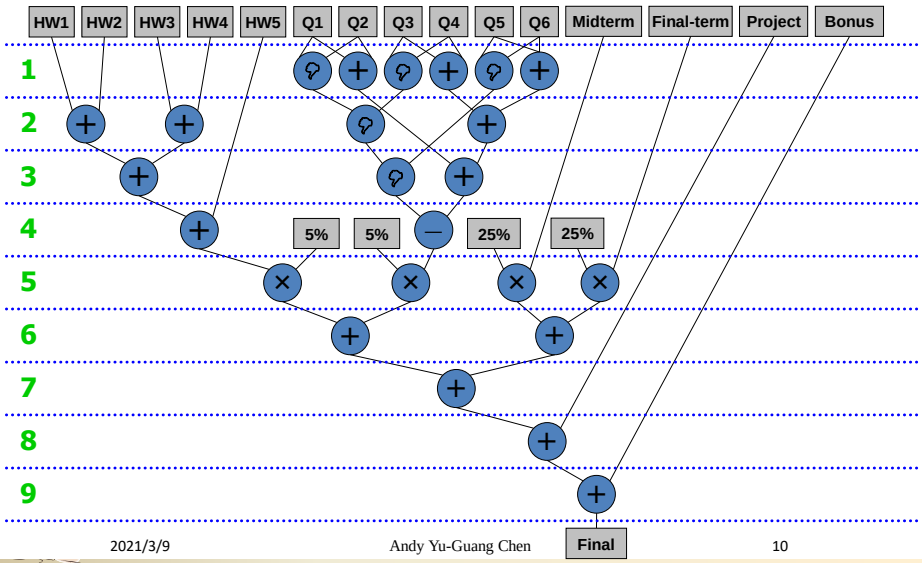
Overall DFG



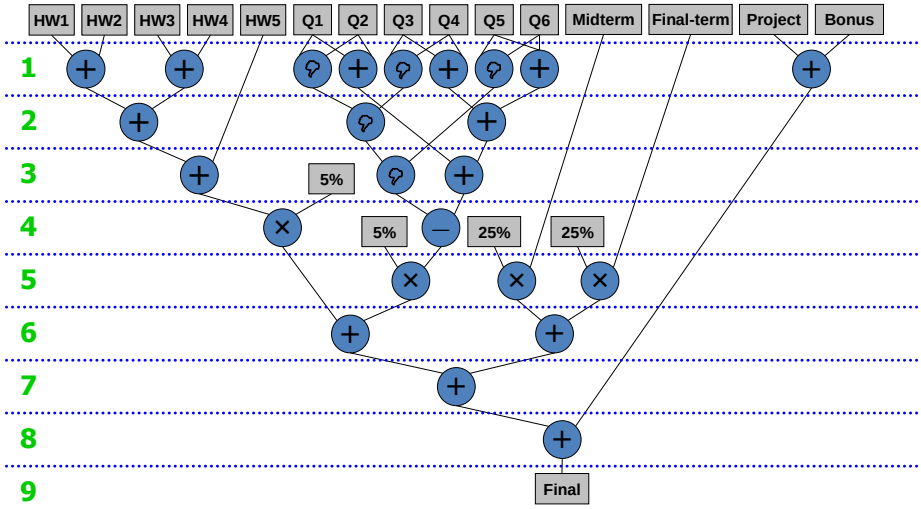
Overall DFG



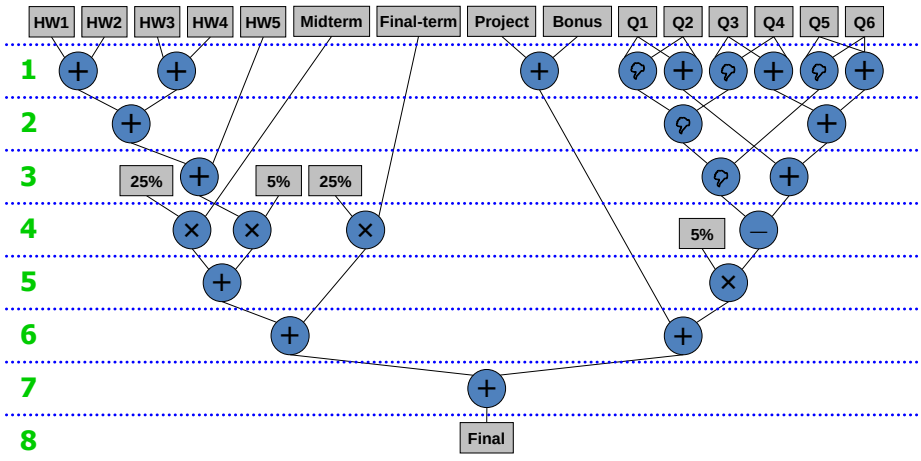
Control Steps



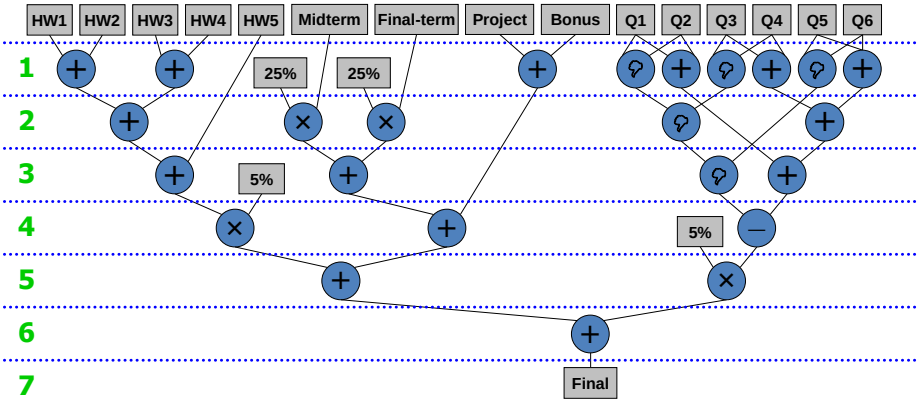
8 Control Steps



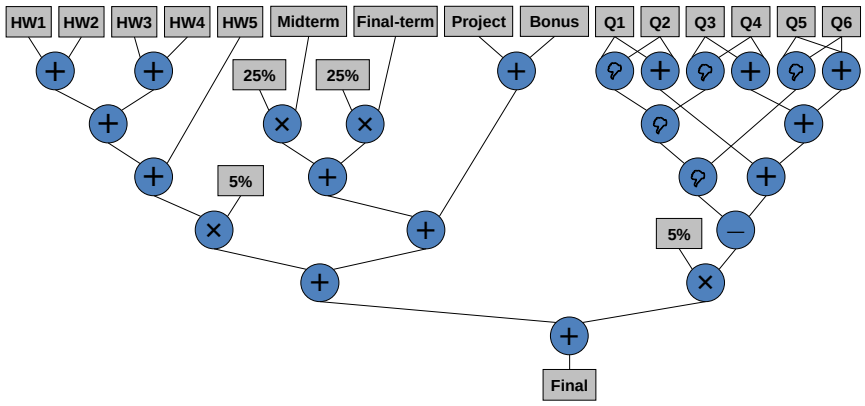
7 Control Steps



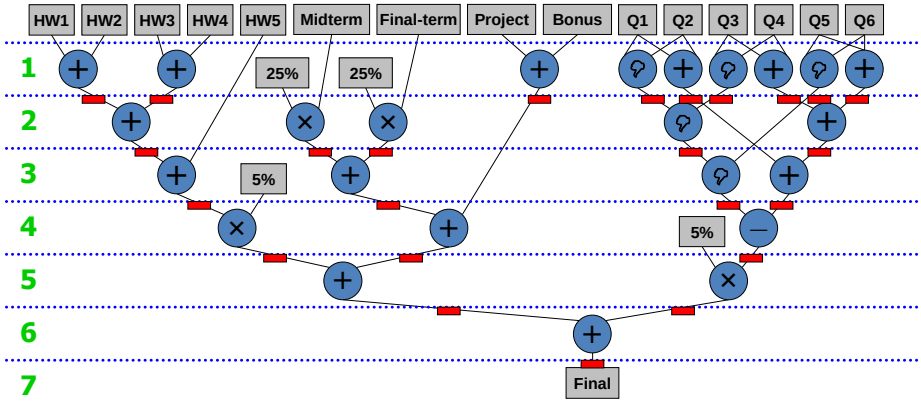
6 Control Steps



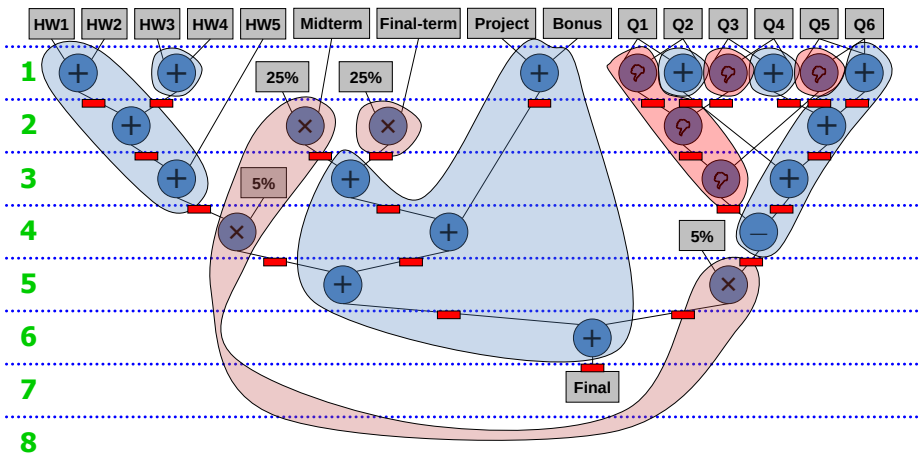
Single-cycle Implementation



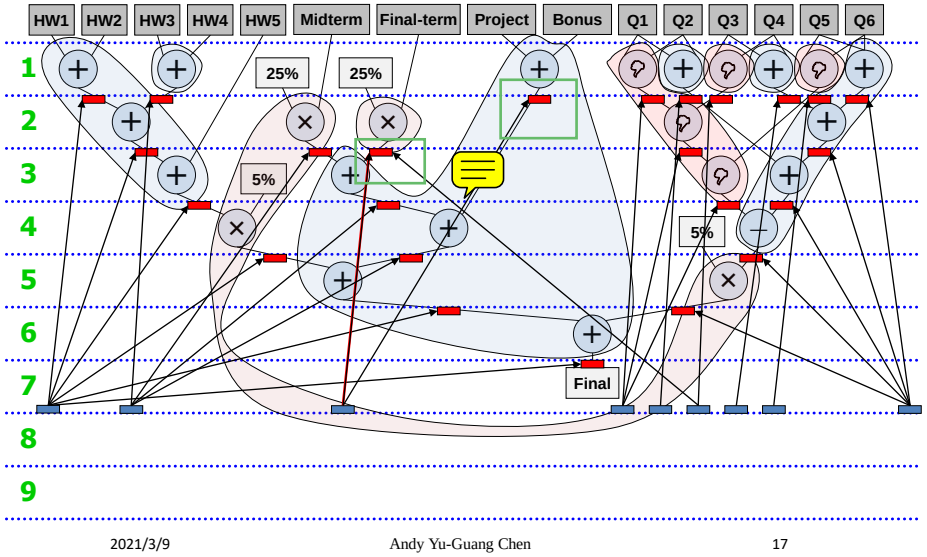
Multi-cycle Implementation



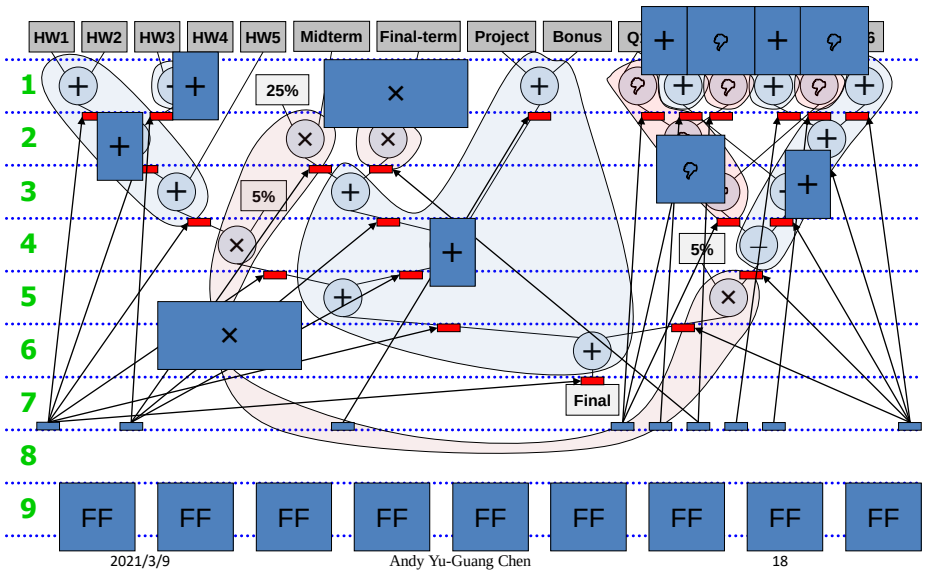
Resource Binding



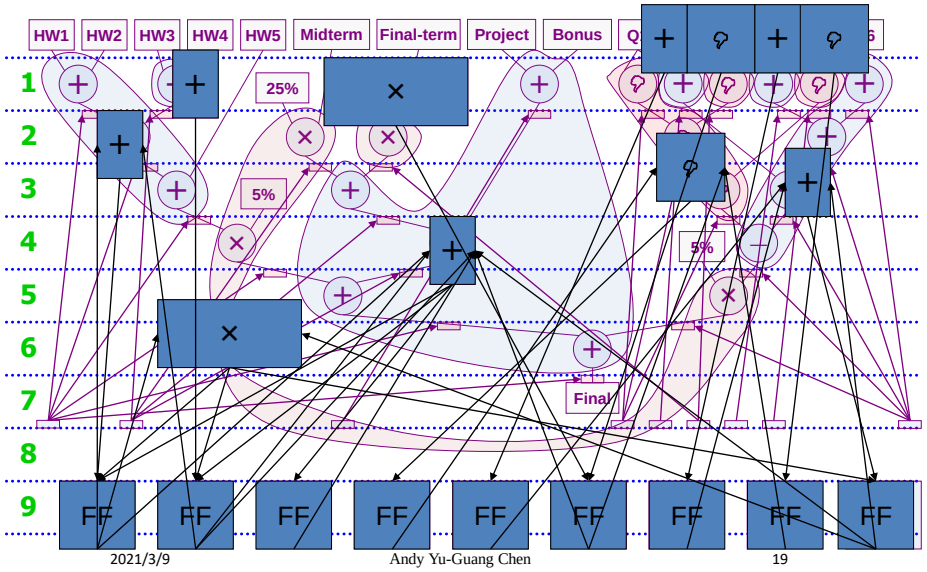
Register Binding



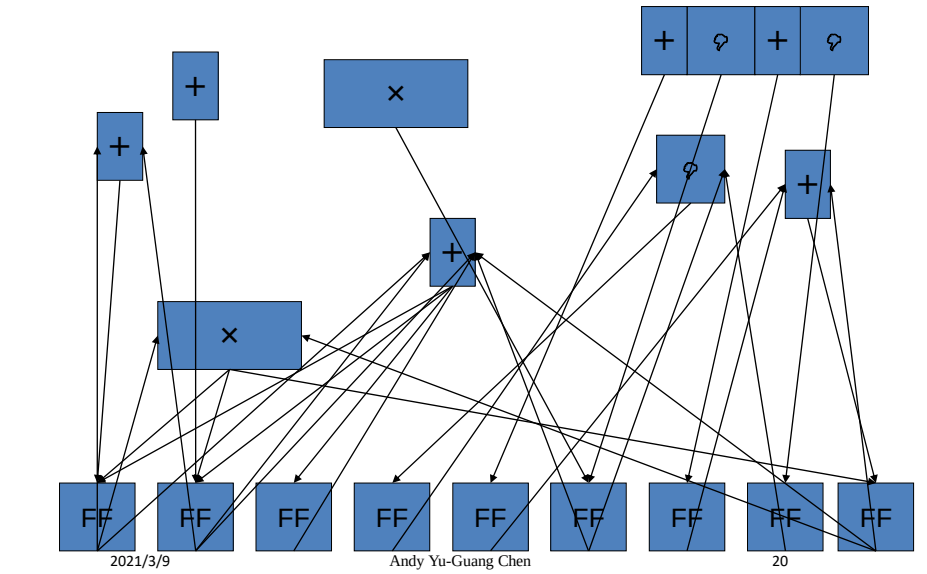
Resource Allocation



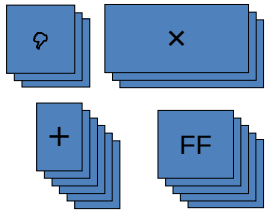
Resource Allocation



Cell Connectivity



Floorplan



1-row height

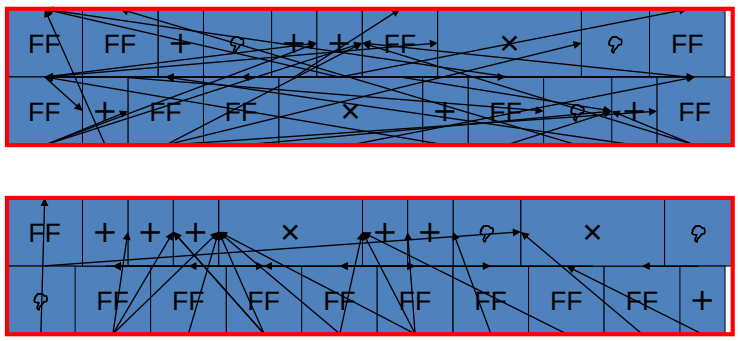
2-row height

3-row height

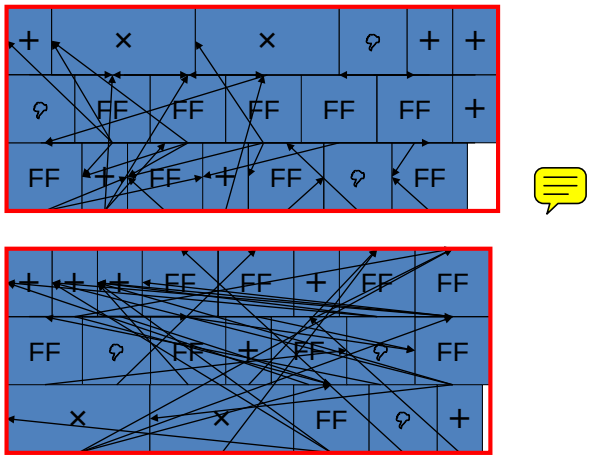
4-row height

5-row height

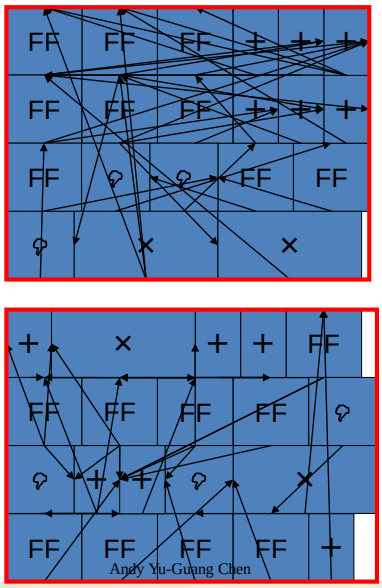
Cell Placement in 2 Rows



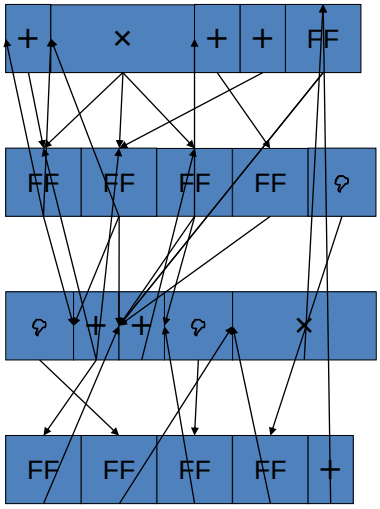
Cell Placement in 3 Rows



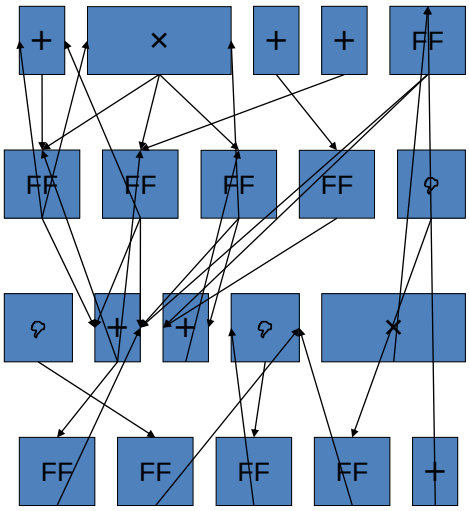
Cell Placement in 4 Rows



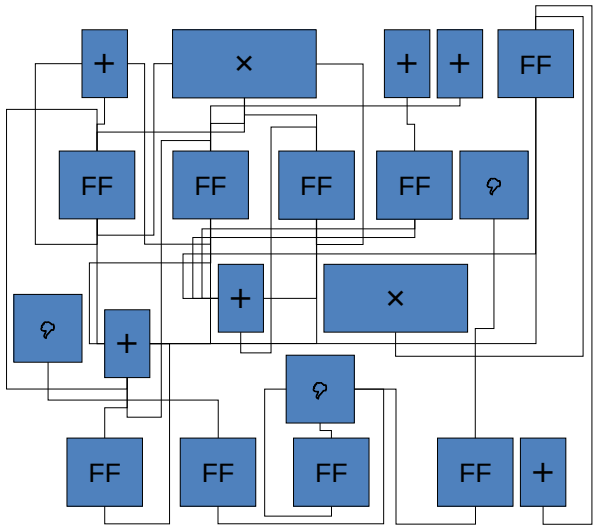
Routing



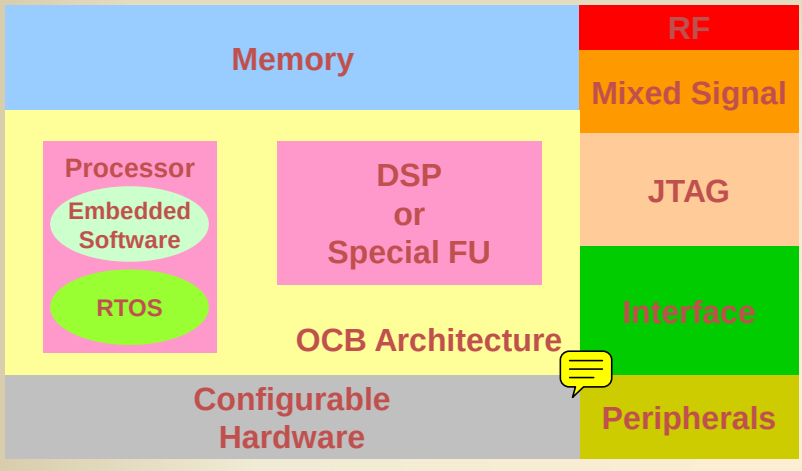
Routing



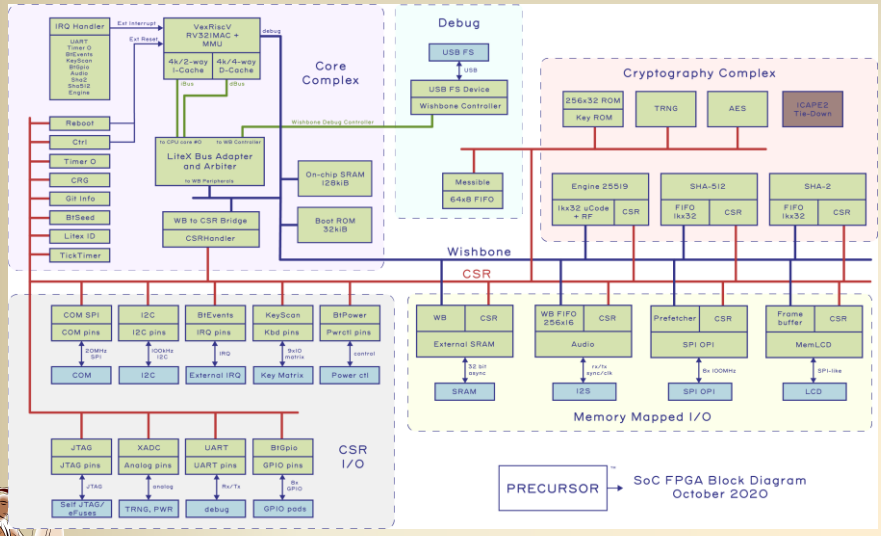
Routing



SoC Architecture



SoC Architecture



Circuit Models

- Circuit model
  - Abstraction
  - Representation of relevant features only
- Unambiguously transferring the design information
  - Humans to humans
  - Humans to CAD tools

Model Classifications

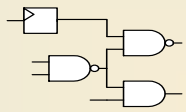
- Abstraction levels
  - Architectural
  - Logic
  - Geometrical
- Views
  - Behavioral
  - Structural
  - Physical
- Media
  - Language
  - Diagram
  - Mathematical model

Abstraction Levels

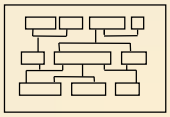
Architectural level

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

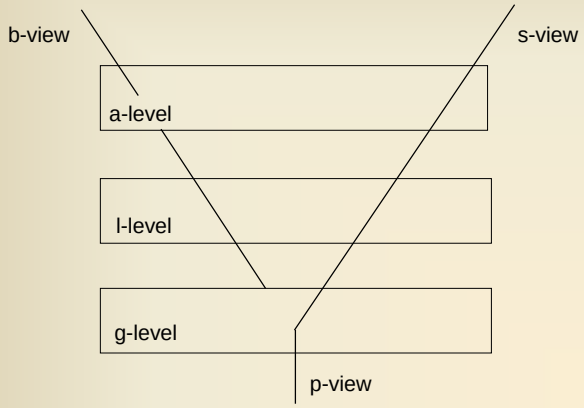
Logic level



Geometrical level



Views and Abstraction Levels



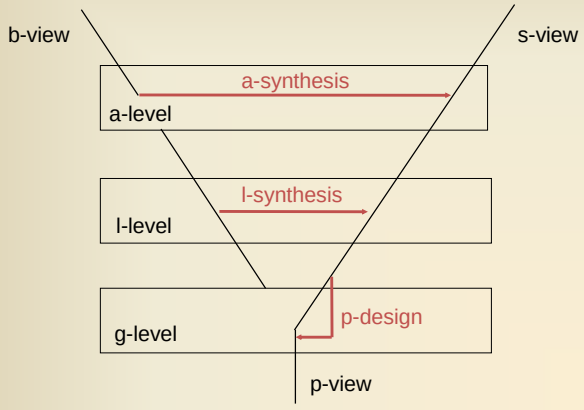
Levels of Abstractions and Corresponding Views

Behavioral View	Structural View	Views / Levels
... PC = PC + 1; FETCH(PC); DECODE(INST); ...	MULT, ADD, CONTROL, RAM	Architectural Level
$state_0$, $state_1$, $state_2$, $state_3$	a, b, c, d, x, y	Logic Level

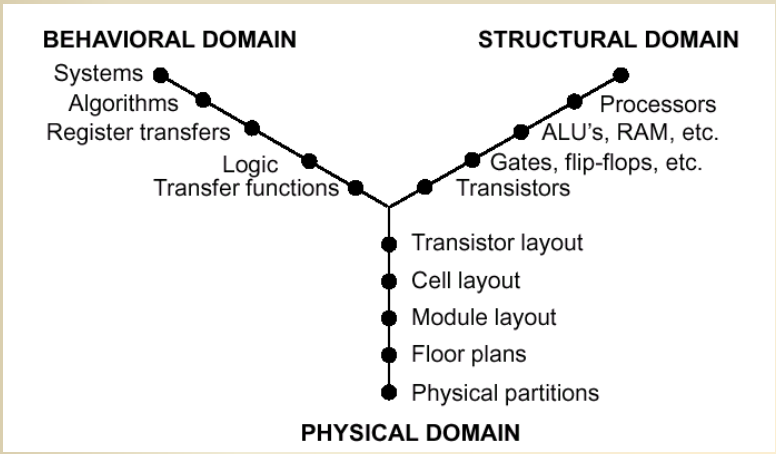
Synthesis

- 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

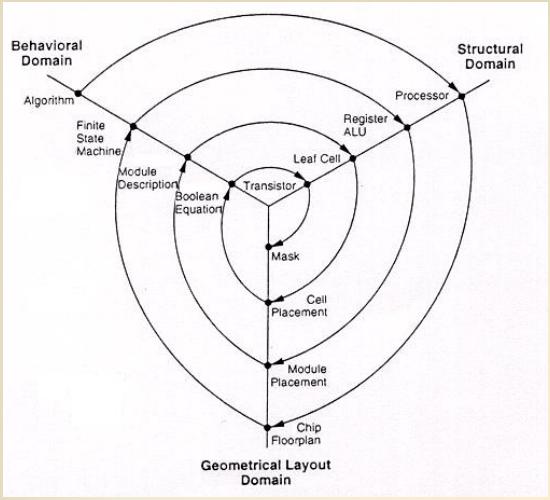
Synthesis Tasks



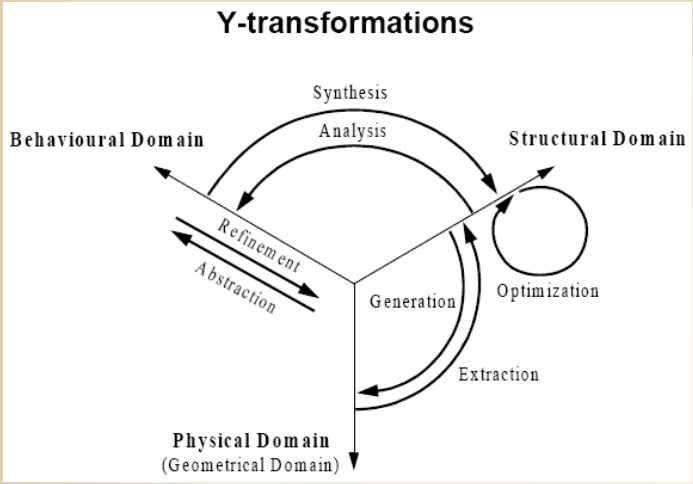
Gajski's Y-chart



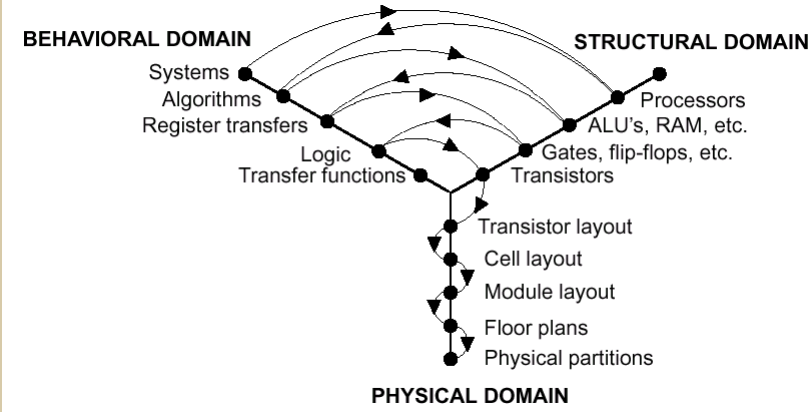
Y-chart Domain Mapping



Y-transformations



Top-down Design



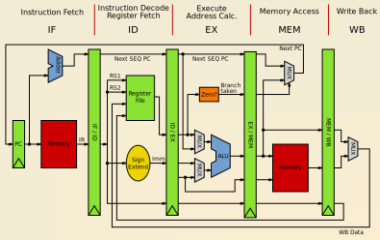
Levels Revealed

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, schematic entry, test
Logic	Boolean function of gate circuits	Schematic entry, synthesis and simulation, verification, PLA tools
Switch	electrical properties of transistor circuits	RC extraction, timing verification, electrical analysis
Layout	geometric constraints	layout editor/compactor, netlist extractor, 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

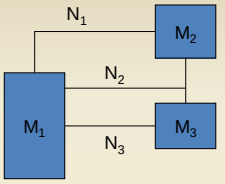


Abstract Models

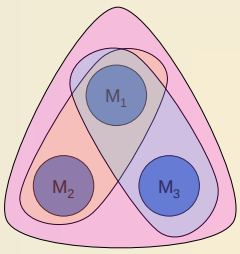
- Abstract models for different circuit views
  - Structure
  - Logic network
    - Combinational logic network
    - Sequential logic network
  - State diagram
  - Data-flow and Sequencing Graph

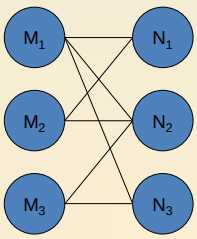
Abstract Models – Structure

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



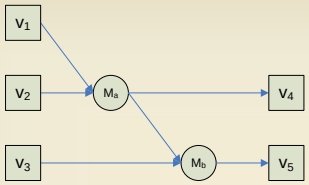
$$\begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \end{array} \begin{array}{c} \begin{array}{ccc} N_1 & N_2 & N_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$



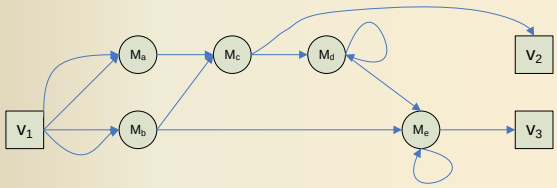


Abstract Models – Logic Network

- Combinational logic network



- Sequential logic network



Abstract Models – State Diagram

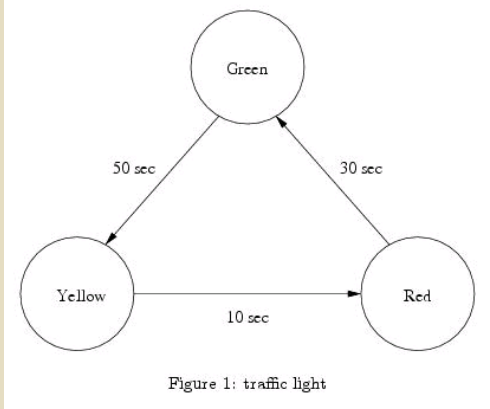
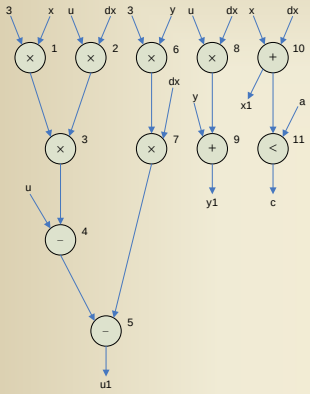
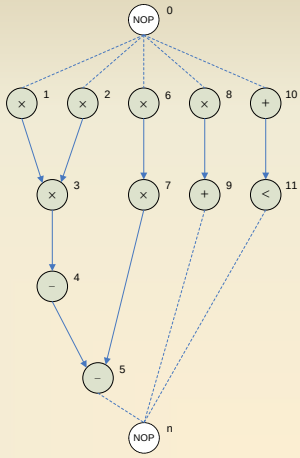


Figure 1: traffic light

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

Hardware Description Language

```
module Full_Adder( A, B, Cin, Sum, Cout );

    input A, B, Cin;
    output Sum, Cout;

    wire W1, W2, W3;

    xor xor1( W1, A, B );
    and and1( W2, W1, Cin );
    and and2( W3, A, B );
    xor xor2( Sum, W1, Cin );
    or  or1( Cout, W2, W3 );

endmodule
```

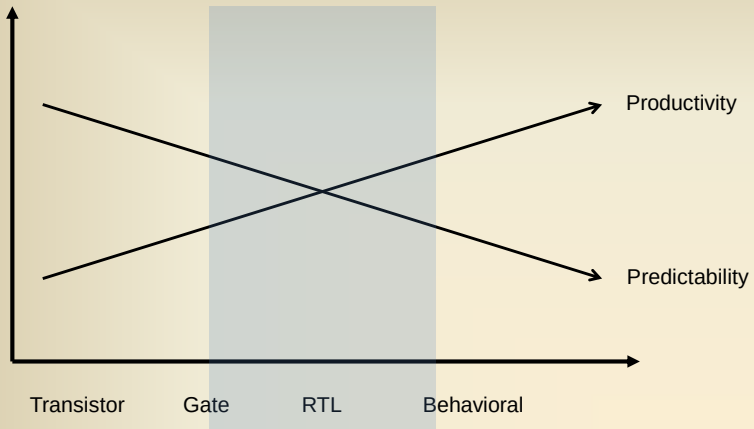
```
module Full_Adder( A, B, Cin, Sum, Cout );

    input A, B, Cin;
    output Sum, Cout;

    wire W1, W2, W3;

    assign W1 = A^B;
    assign W2 = W1&Cin;
    assign W3 = A&B;
    assign Sum = W1^Cin;
    assign Cout = W2|W3;

endmodule
```

```
module Full_Adder( A, B, Cin, Sum, Cout );

    input A, B, Cin;
    output Sum, Cout;

    wire W1, W2, W3;

    always @( A, B, Cin ) begin
        { Cout, Sum } = A + B + Cin;
    end

endmodule
```

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

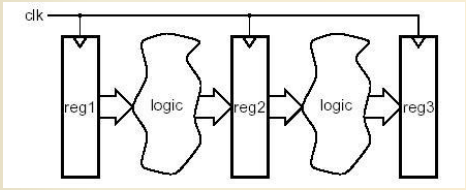
Productivity vs. Predictability



∴ 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

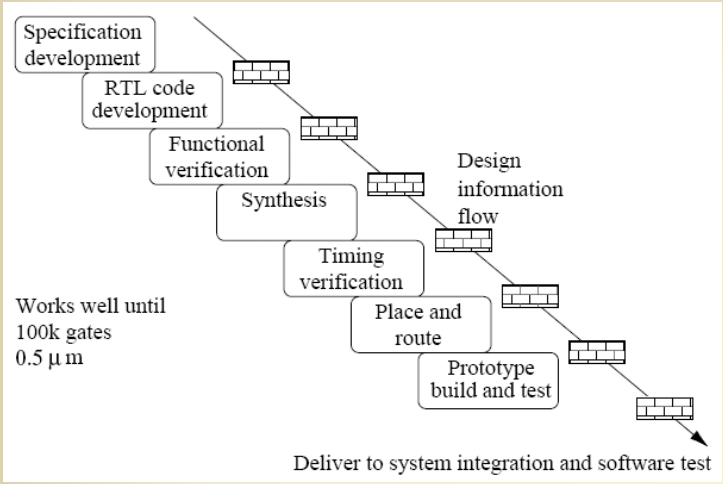
- Structural description

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

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

endmodule
```

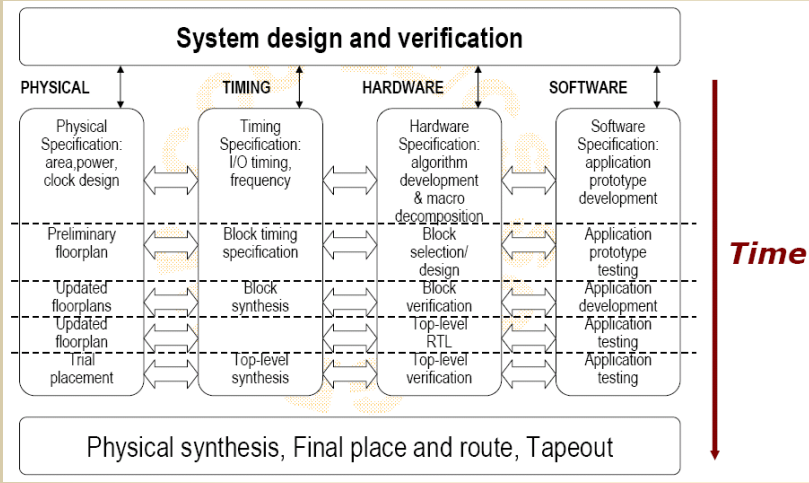
- Behavioral description

```
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



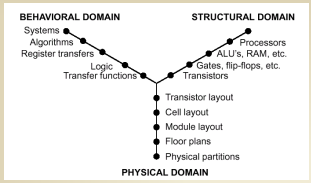
Spiral SOC Design Flow

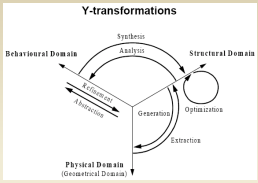
- Characteristics
  - Parallel, concurrent development of hardware and software
  - Parallel verification and synthesis of modules
  - Floorplan, placement, and routing are included in the synthesis process
  - Modules developed only if a pre-designed hard or soft macro is not available – reusability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

Summary





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, schematic entry, test
Logic	Boolean function of gate circuits	Schematic entry, synthesis and simulation, verification, PLA tools
Switch	electrical properties of transistor circuits	RC extraction, timing verification, electrical analysis
Layout	geometric constraints	layout editor/compactor, netlist extractor, DRC, placement and routing