

Σχεδιασμός Συστημάτων VLSI

Εαρινό 2023

ΕΠΙΚ. Καθ. Γεώργιος Ζερβάκης

Διαδικαστικά

- ❑ Παραδόσεις: Πέμπτη 11:00 - 14:00
- ❑ Εργαστήριο: Θα ανακοινωθεί.
 - Τετάρτη; Παρασκευή;
 - Έναρξη: 3^η ή 4^η εβδομάδα
- ❑ Προαπαιτούμενα για το μάθημα
 - Λογική Σχεδίαση (I & II)
 - Εισαγωγή Σε VLSI
 - Αρχιτεκτονικής Υπολογιστών I,II και Μικροεπεξεργαστών

Διαδικαστικά

- ❑ Συγγράμματα
 - ΣΧΕΔΙΑΣΗ ΟΛΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ CMOS VLSI
 - ΨΗΦΙΑΚΗ ΣΧΕΔΙΑΣΗ: ΑΡΧΕΣ ΚΑΙ ΠΡΑΚΤΙΚΕΣ
 - ΣΧΕΔΙΑΣΜΟΣ ΚΥΚΛΩΜΑΤΩΝ ΜΕ ΤΗ VHDL
- ❑ Καλό ανάγνωσμα για Verilog:
 - Verilog: The Verilog Hardware Description Language, Thomas, Moorby
- ❑ **eclass**
 - ΣΧΕΔΙΑΣΜΟΣ ΣΥΣΤΗΜΑΤΩΝ VLSI
 - Ανακοινώσεις, επικοινωνία, εργαστήρια



Διαδικαστικά

- ❑ Εργαστηριακή εξάσκηση (υποχρεωτική)
 - 5+ εργαστηριακές ασκήσεις
 - Δικαίωμα για δύο απουσίες
 - Μπορείτε να αναπληρώσετε τη μία

- ❑ Τελικός Βαθμός = $0,6 * \text{Βαθμός Θεωρητικής Εξέτασης}$
+ $0,4 * \text{Βαθμός Εργαστηρίου}$

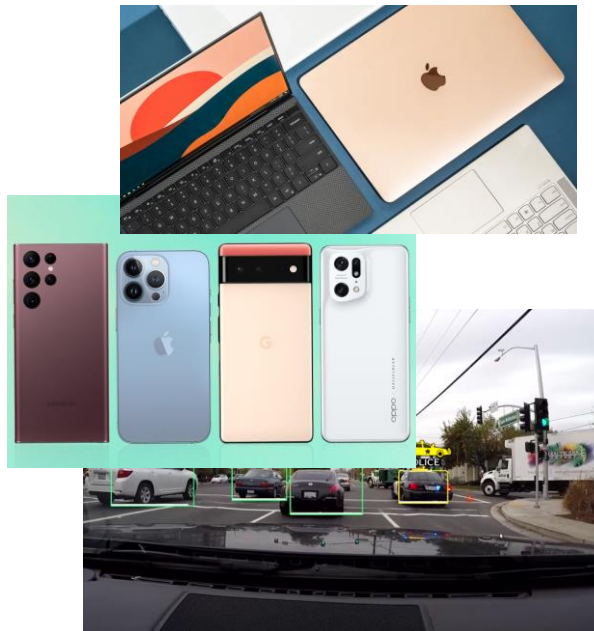
αν και μόνο αν

Βαθμός Θεωρητικής Εξέτασης ≥ 5 και
Βαθμός Εργαστηρίου ≥ 5

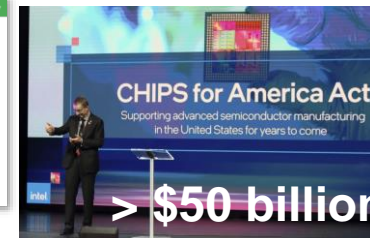
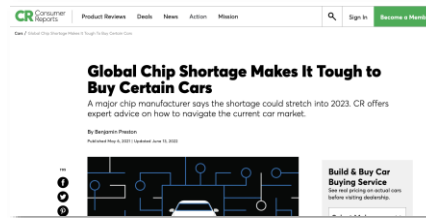
Introduction

Digital Design

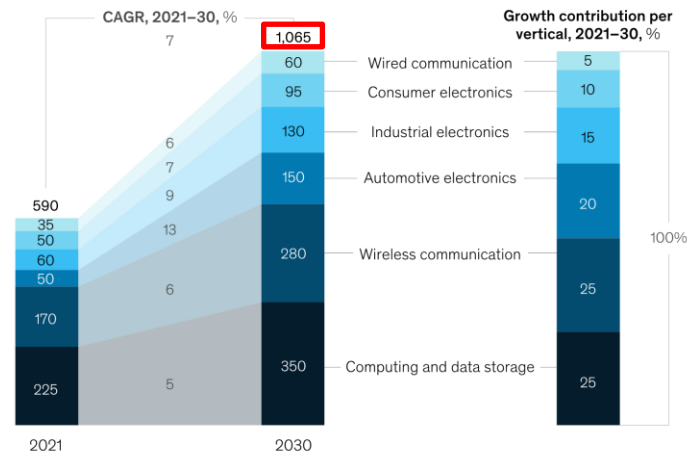
Life



Economy



Global semiconductor market value by vertical, indicative, \$ billion



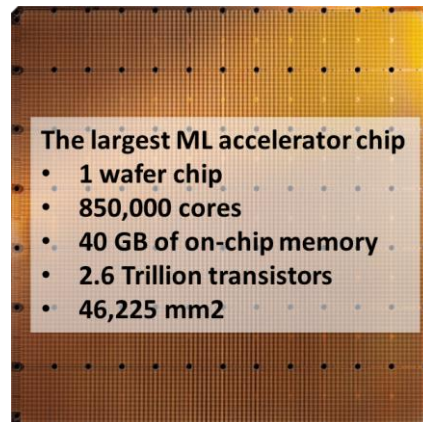
Courtesy: McKinsey & Company

Digital Design

- ❑ Renaissance in Computer Architecture and ASIC design
- ❑ Explosion of Domain-Specific AI accelerators and chips
 - e.g., ARM Ethos, Google TPU, Samsung NPU, IBM AI chip, Intel NNP-I, NVIDIA NVDLA, Groq, Graphcore, etc



Google TPU

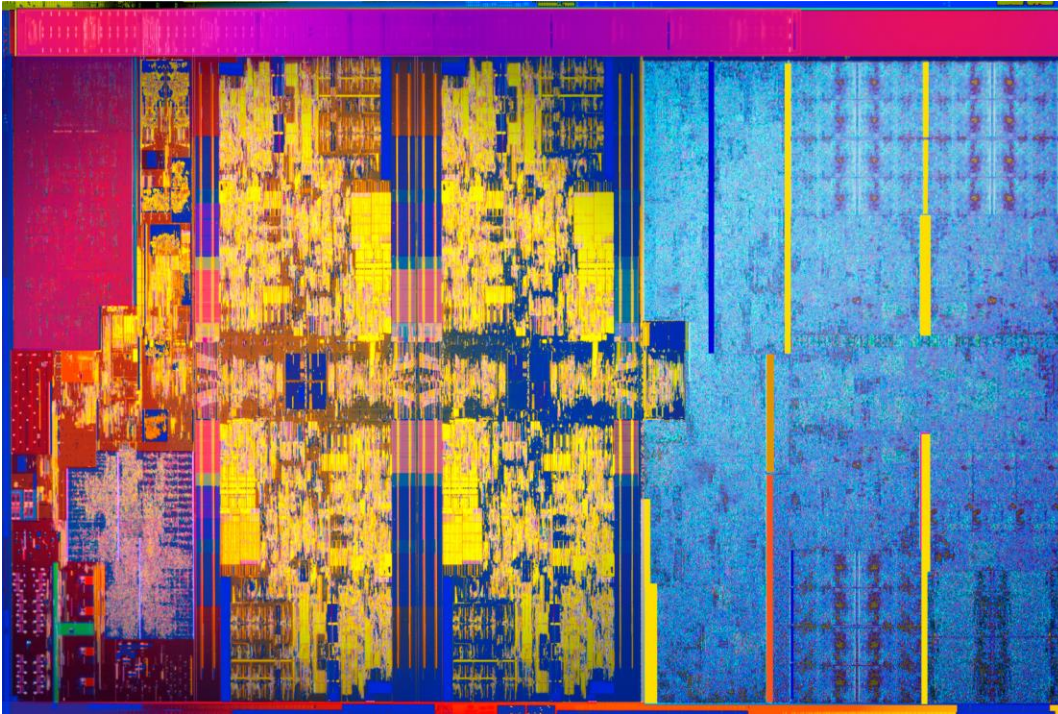


Cerebras WSE2



Samsung Exynos 9 [samsung.com]

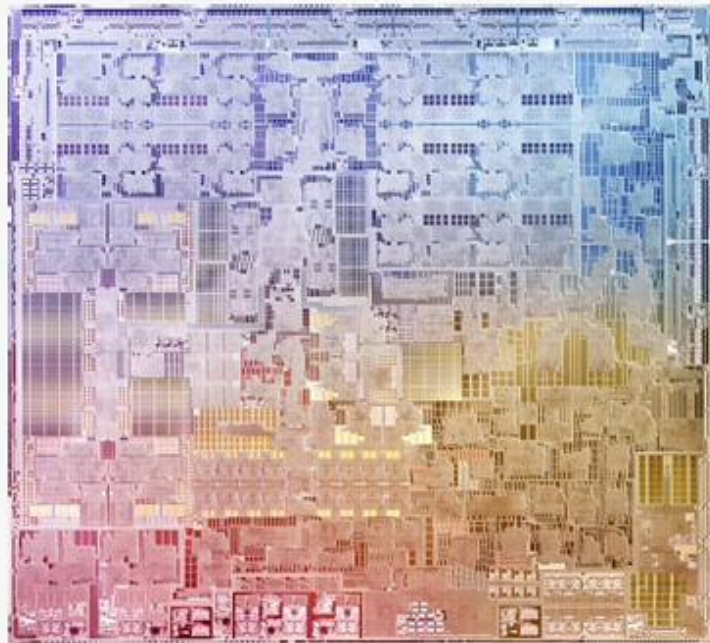
2017: Intel Kaby Lake



https://en.wikichip.org/wiki/intel/microarchitectures/kaby_lake

- 64-bit processor
- 4 cores, 8 threads
- 14-19 stage pipeline
- 3.9 GHz clock
- 1.75B transistors

Apple M2



Apple M2

- 8 cores
- 20B transistors
- up to 3.5 GHz

<https://www.apple.com/newsroom/2022/06/apple-unveils-m2-with-breakthrough-performance-and-capabilities/>

How to Deal with This Complexity?

- ❑ Hardware Description Languages!
- ❑ A fact of life in computer engineering
 - Need to be able to **specify complex designs**
 - communicate with others in your design group
 - ... and to **simulate** their behavior
 - *yes, it's what I want to build*
 - ... and to **synthesize** (automatically design) portions of it
 - have an error-free path to implementation
- ❑ Hardware Description Languages
 - Many similarly featured **HDLs** (e.g., **Verilog**, VHDL, ...)
 - if you learn one, it is **not hard to learn** another
 - mapping between languages is typically **mechanical**, especially for the commonly used subset

Purpose of HDLs

- ❑ Purpose of Hardware Description Languages:
 - Capture design in Register Transfer Language form
 - i.e. All registers specified
 - Use to simulate design so as to verify correctness
 - Pass through Synthesis tool to obtain reasonably optimal gate-level design that meets timing
 - Design productivity
 - Automatic synthesis
 - Capture design as RTL instead of schematic
 - Reduces time to create gate level design by an order of magnitude
- ❑ Synthesis
 - Basically, a Boolean Combinational Logic optimizer that is timing aware

Hardware Description Languages

- ❑ **Two well-known hardware description languages**

- ❑ **Verilog**

- Developed in 1984 by Gateway Design Automation
- Became an IEEE standard (1364) in 1995
- More popular in US

- ❑ **VHDL (VHSIC Hardware Description Language)**

- Developed in 1981 by the US Department of Defense
- Became an IEEE standard (1076) in 1987
- More popular in Europe

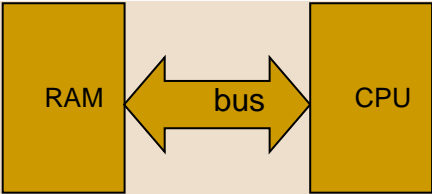
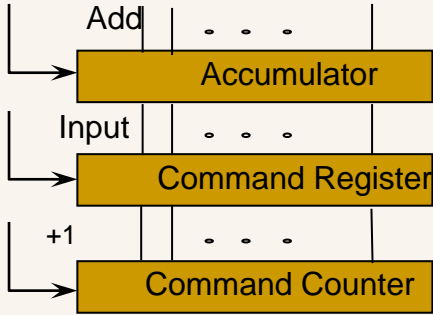
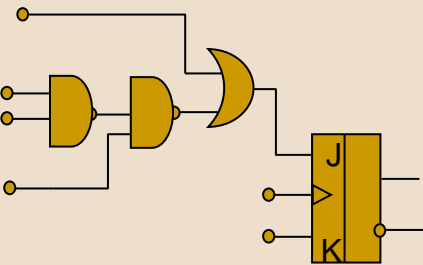
- ❑ In this course we will use Verilog

What this course is about

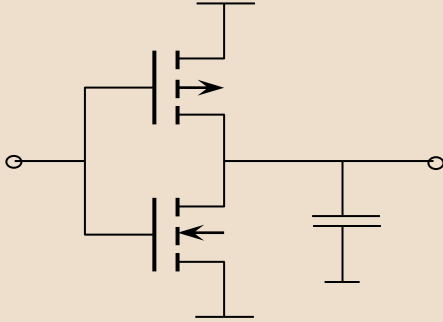
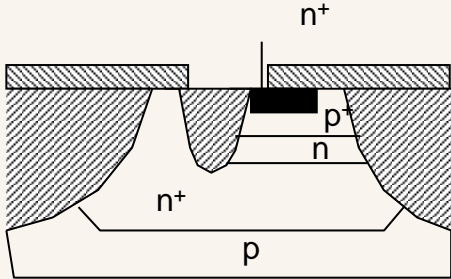
- ❑ HDL – Verilog
- ❑ Design Complex VLSI Systems
- ❑ Digital Design Flow
 - Synthesis, Simulation, and Evaluation

Digital Design Flow

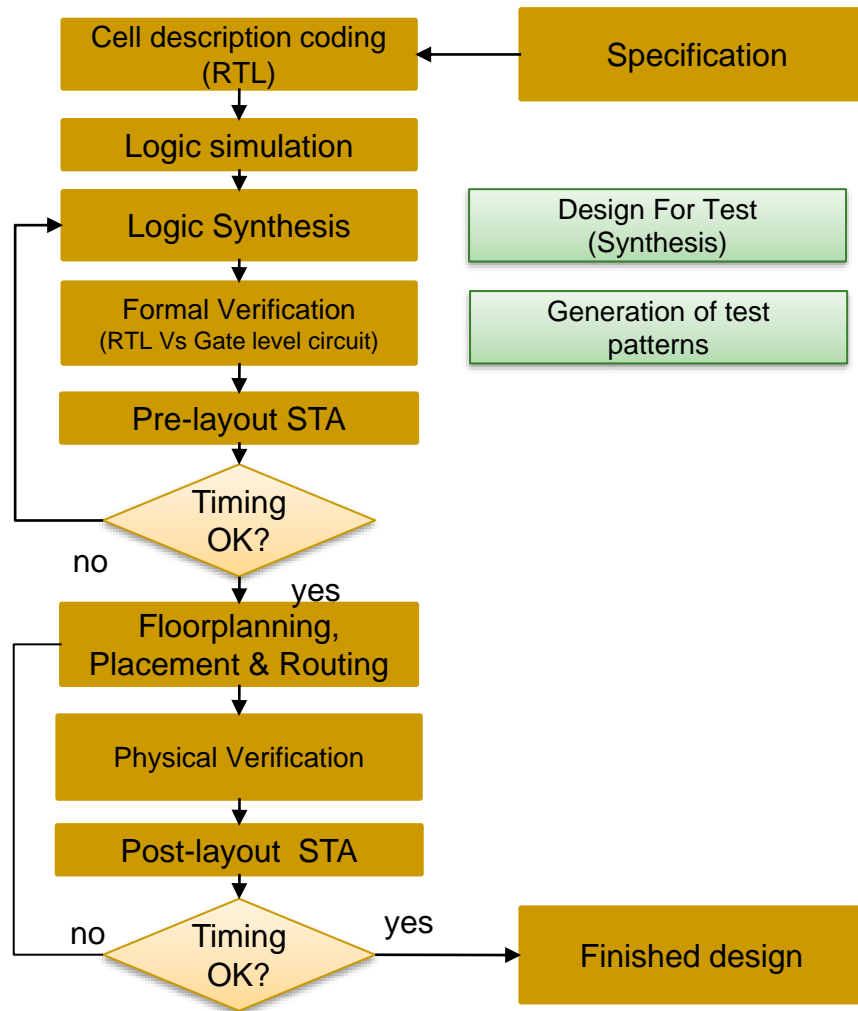
Design Levels

Level	Modeling Object	Example of Modeling Object	Mathematical Apparatus
System	Structural Circuit	 <p>A diagram showing a yellow box labeled 'RAM' on the left and a yellow box labeled 'CPU' on the right. They are connected by a double-headed yellow arrow labeled 'bus'.</p>	Queuing theory
Register-Transfer	Functional Circuits on the level of multi-bit devices	 <p>A diagram showing three yellow boxes stacked vertically. The top box is labeled 'Accumulator' and has an input labeled 'Add' and an output labeled 'Add'. The middle box is labeled 'Command Register' and has an input labeled 'Input' and an output labeled 'Input'. The bottom box is labeled 'Command Counter' and has an input labeled '+1' and an output labeled '+1'. Each box has three dots between its input and output lines, indicating multi-bit devices.</p>	Boolean Algebra
Gate	Circuit on the level of gates and flip-flops	 <p>A diagram showing a logic circuit. It starts with three inputs on the left. Two inputs go into an AND gate. The output of this AND gate goes into another AND gate along with a third input. The output of this second AND gate goes into an OR gate. The output of the OR gate goes into a J-K flip-flop. The flip-flop has inputs labeled 'J' and 'K' and an output line.</p>	

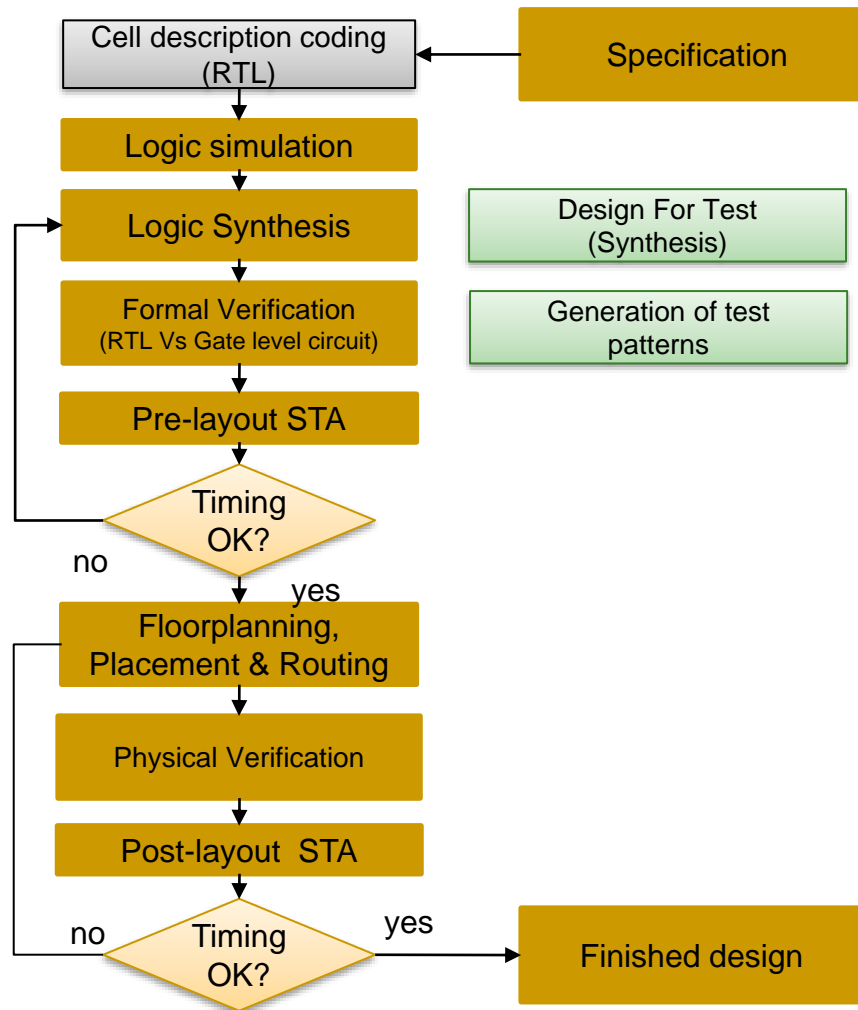
Design Levels (2)

Level	Modeling Object	Example of Modeling Object	Mathematical Apparatus
Circuit	Electrical Circuit		System of differential equations
Device	IC Components		System of differential equations with partial derivative

Digital Design Flow



Cell Description



Digital IC Specification

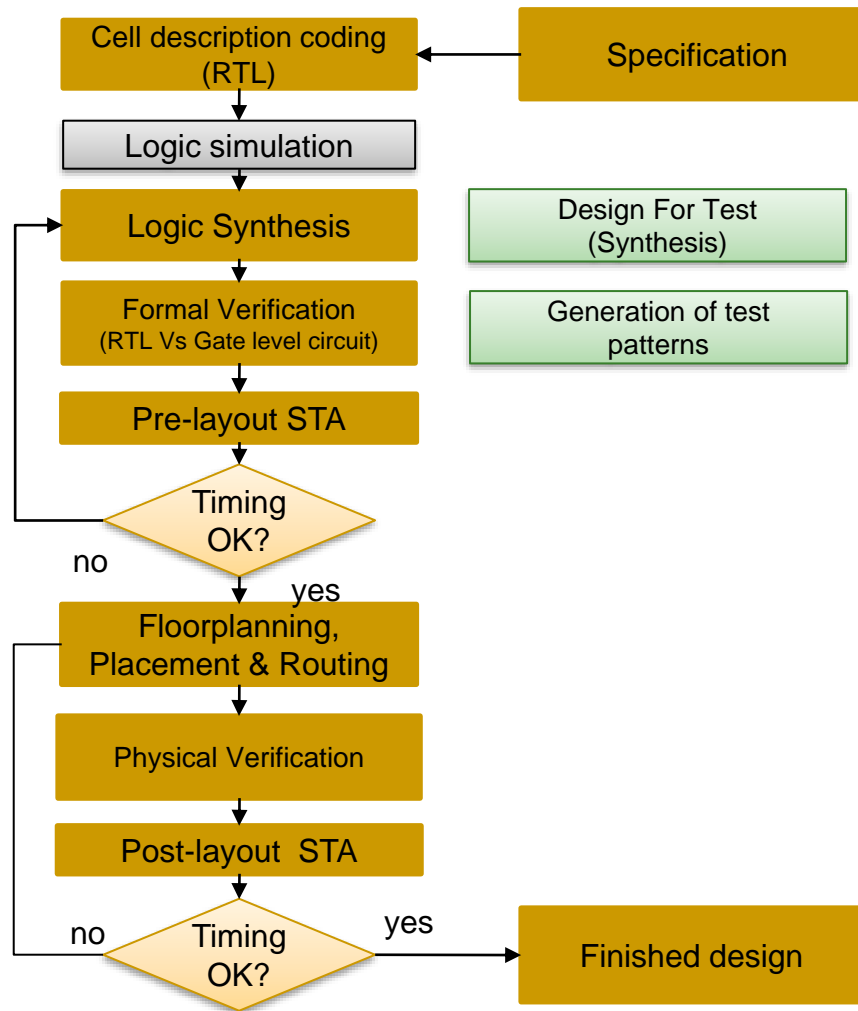
- ❑ Description of Digital IC functionality
- ❑ With the help of Verilog or VHDL in the specification
- ❑ An example of specification line:
 - if incoming_call AND line_is_available then RING;
- ❑ The specifications of contemporary Digital IC can contain millions of lines, can be created by a collective of numerous participants within a few months

Verilog Description Example

❑ 8-bit counter

```
module counter(out, reset, clock);  
  
    output [7:0] out;  
    input  clock, reset;  
    reg    [7:0] out;  
  
    always @(posedge clock or posedge resetn)  
  
        if (resetn)  
            out = 0;  
        else  
            out = out + 1;  
  
endmodule
```

Digital Design Flow

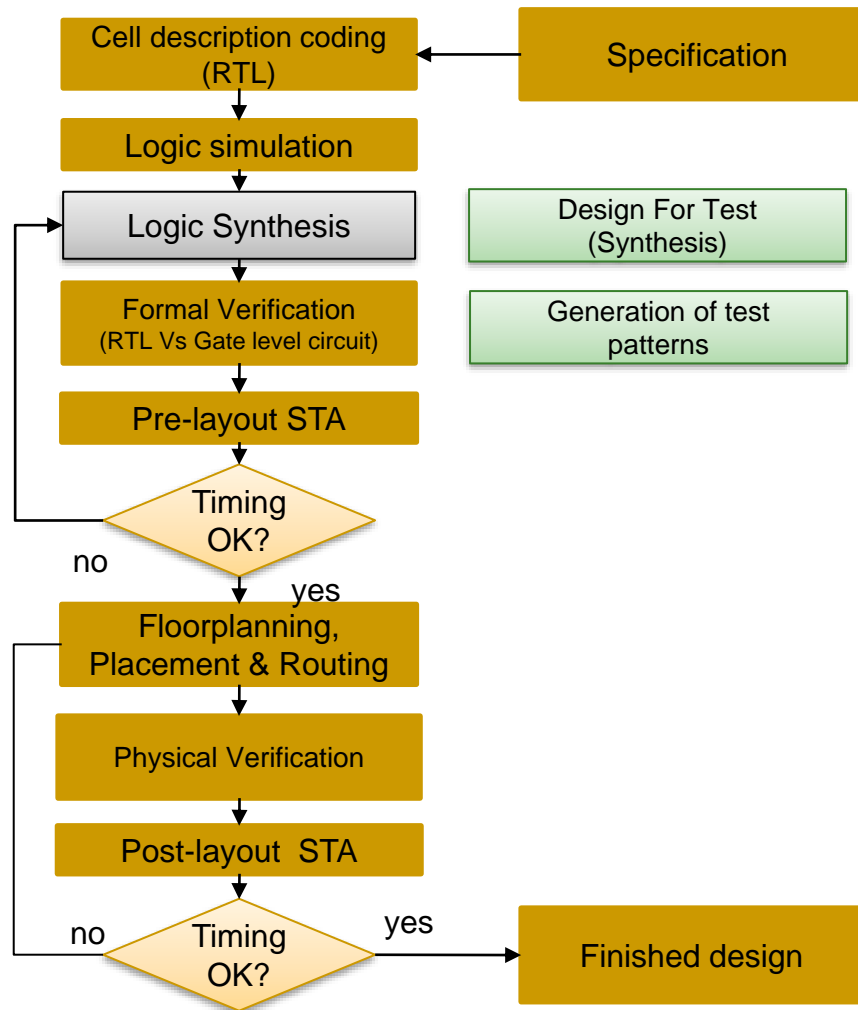


The left screenshot shows the 'DVE - TopLevel.1 - [Data.1]' window. It displays a Verilog module definition for 'jo'. The module has inputs 'clk', 'r', and 'size[31:0]', and an output 'out[0:7]'. The code includes a parameter 'size=7', a register 'reg [0:size] out;', and a logic block 'always @ (posedge clk or posedge r)'. The right screenshot shows the 'DVE - TopLevel.2 - [Wave.1]' window, which is a timing diagram. It displays signals 'clk', 'out[0:7]', 'r', and 'size[31:0]' over time. The 'out[0:7]' signal is shown with a red highlight and a 'match' status. The 'r' signal is shown with a red highlight and a 'match' status. The 'size[31:0]' signal is shown with a red highlight and a 'match' status. The 'clk' signal is shown with a red highlight and a 'match' status. The timing diagram includes a 'Group1' section and a 'New Group' section, with a 'match' status bar at the bottom.

SYNOPSYS®

VLSI Systems Design
SS23 | CEID, UPatras

Digital Design Flow



Logic Synthesis

❑ Synthesis

- The process which converts an abstract form of desired circuit behavior into a design implementation in terms of logic gates

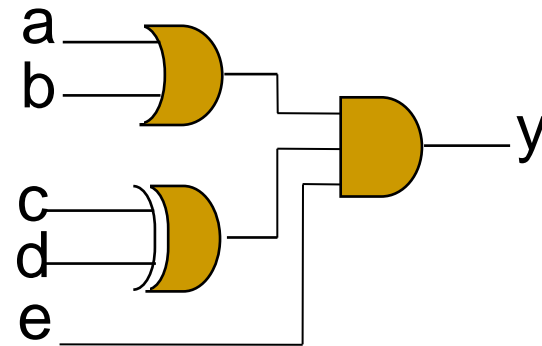
❑ Optimization

- Changing design to achieve design goal (required by specification)

Concept of Automated Design

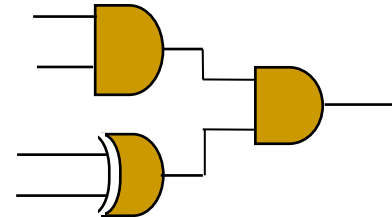
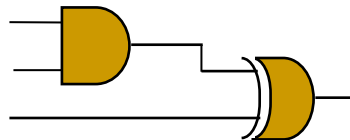
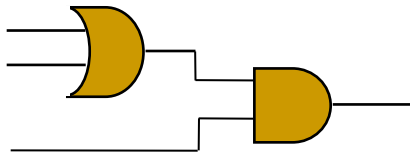
- Any digital function can be easily converted to a logic circuit. **Synthesis** tool uses this to automatically synthesize circuit from functional description.

$$Y = (a + b) \& (c \oplus d) \& e$$

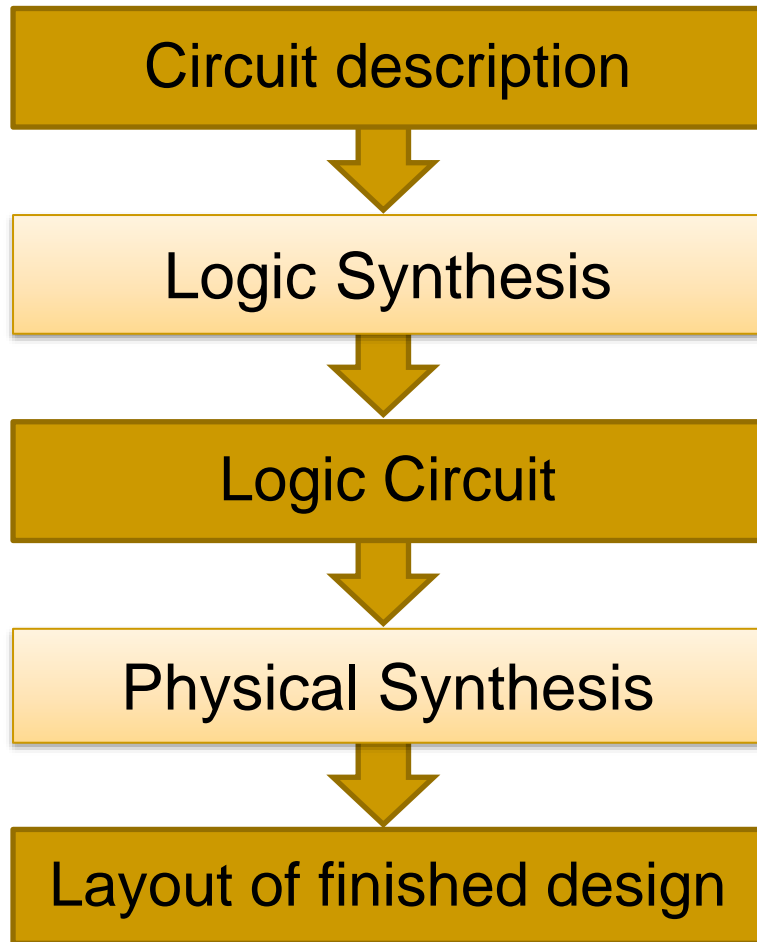


Concept of Automated Design (2)

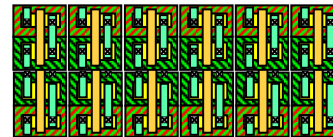
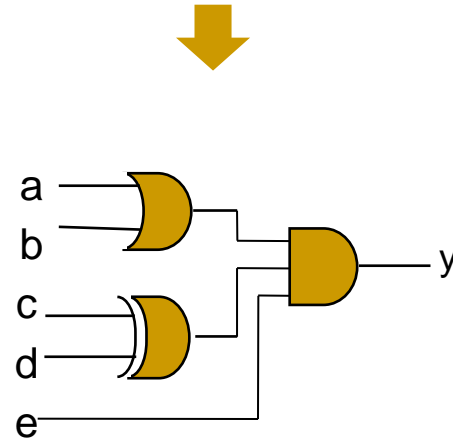
- ❑ If primitive (standard) cells are previously designed, large number of various digital circuits can be built using these parts



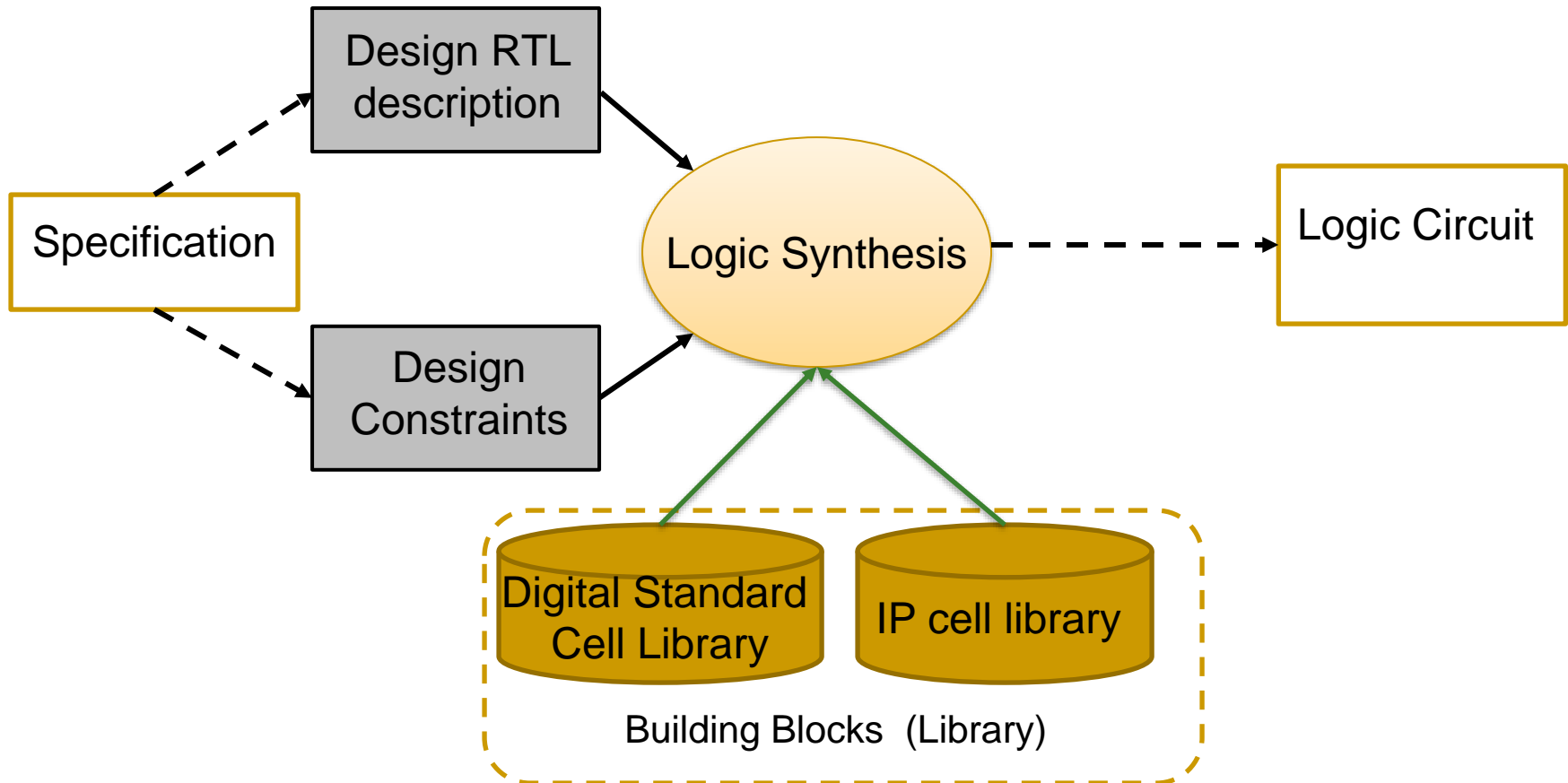
Basic Steps of Synthesis



$$y = (a + b) \& (c \oplus d) \& e$$



Logic Synthesis



Design Constraints

Specification

Design Description

Circuit must represent multiplication unit with 16bit word width and 4KB memory.

N0	Parameter description	Min	Typ	Max	Units
1.	Process	3.3V IO devices in TSMC 0.11			
2.	Input Clock Frequency	18		30	MHz
3.	Power Supply Voltage1	0.8	1.5	1.8	V
4.	Power Supply Voltage2	3	3.3	3.6	V
5.	Power Dissipation		125	180	mW
6.	Operating Temperature	0		125	°C
7.	Clock Jitter			28	Ps
8.	Die Area		2		um ²
9.					

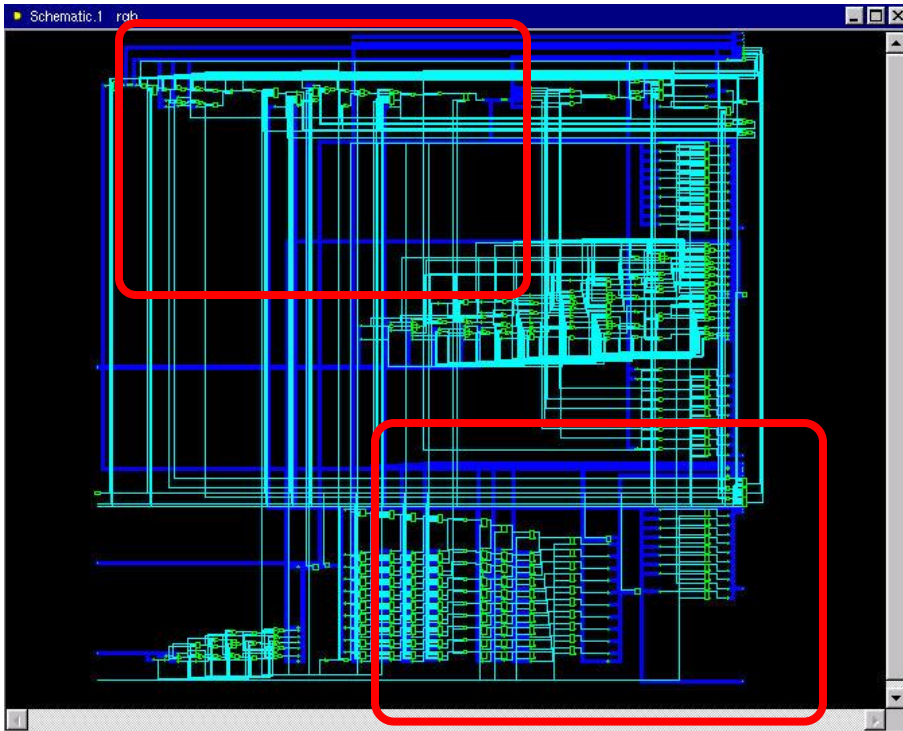
Design goals are specified as **constraints**.

Power \leq 100 mW
Area = 2 um²
18MHz < Frequency < 30 MHz

Design constraints are used as input for synthesis.

Logic Circuit

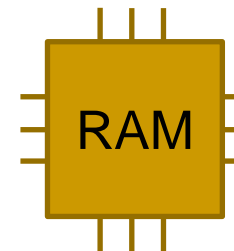
Digital standard cells



Intellectual property (IP) block

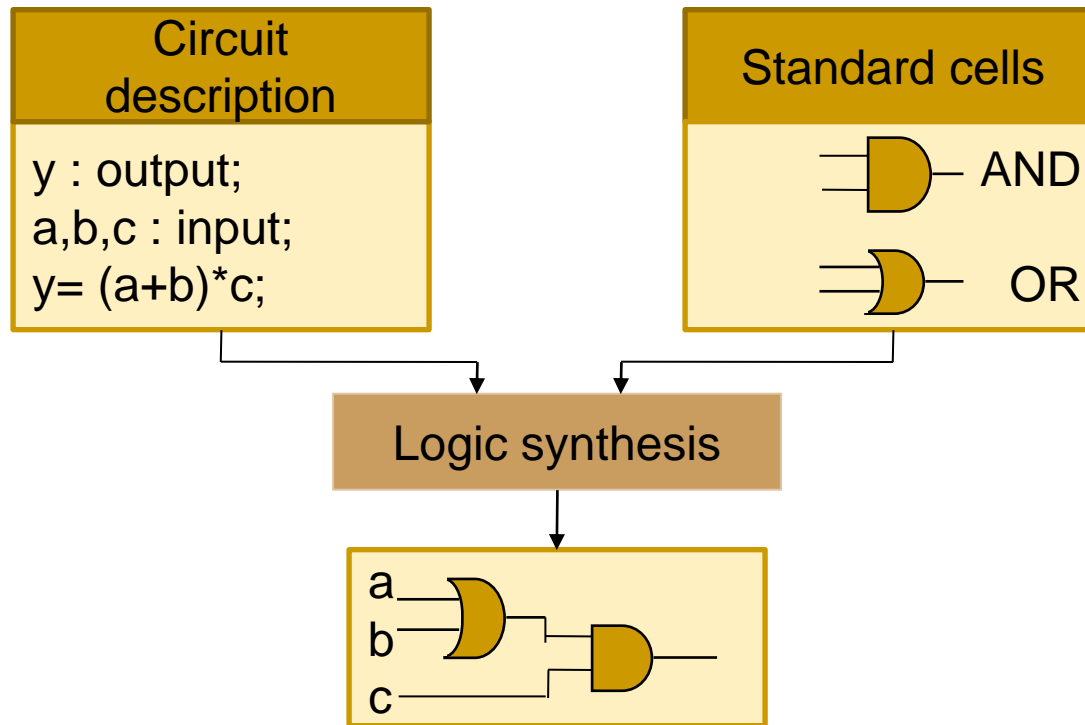
Final logic circuit consists of digital standard cells and Intellectual Property (IP) blocks.

Digital Standard Cell Library and IP cell library should be given as input to synthesis to be used as building blocks.



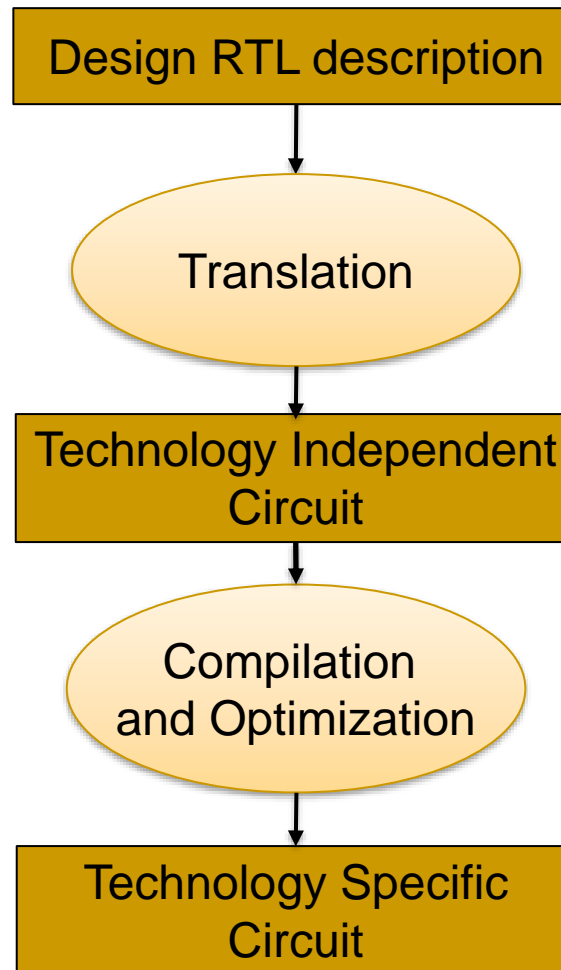
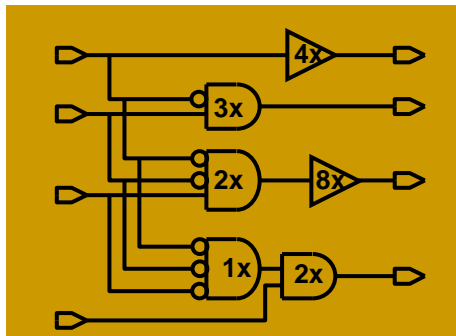
Logic Synthesis

Logic synthesis is the process which produces logic circuit from circuit description

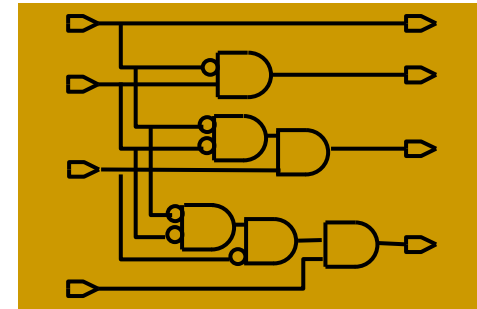


Logic Synthesis Steps

Technology Specific Circuit is obtained from independent one by replacing all components by real blocks (standard cells). This replacement process is also called **mapping**.

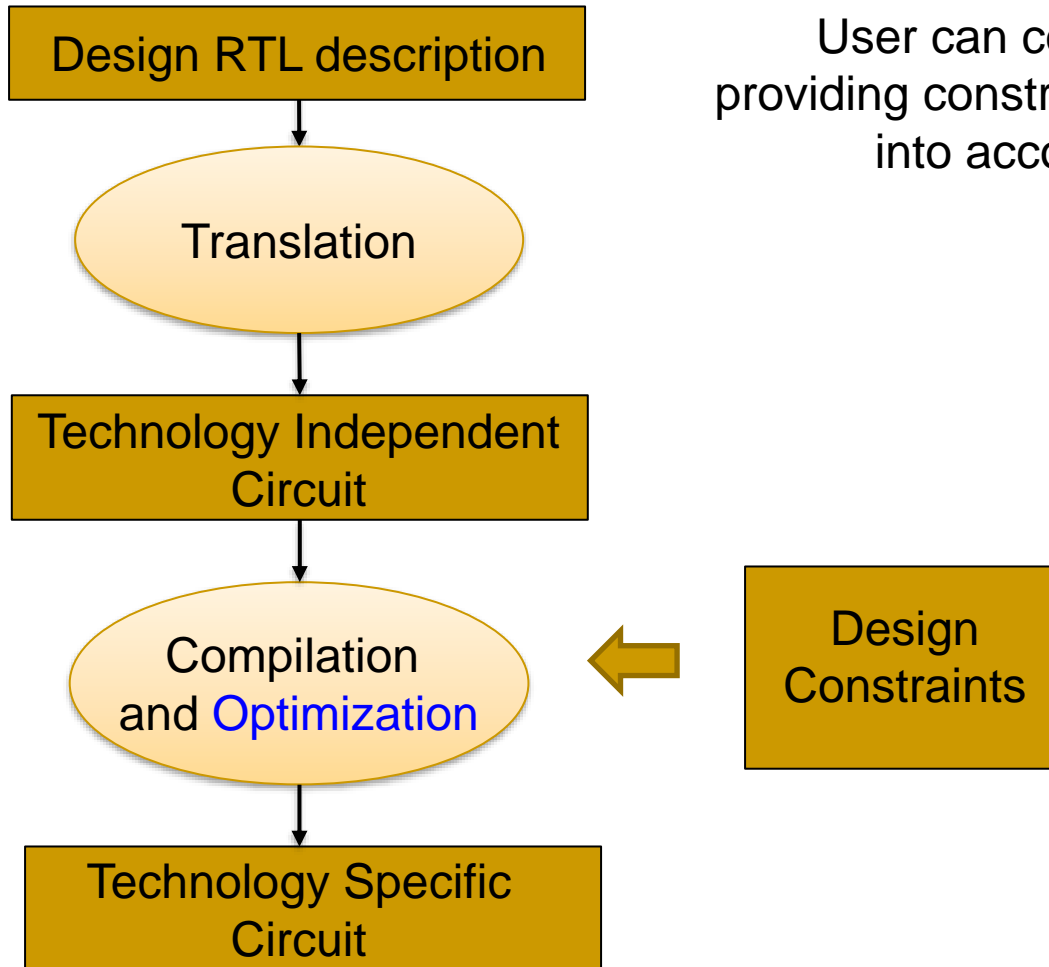


```
residue = 16'h0000;  
if (high_bits == 2'b10)  
    residue = state_table[index];  
else  
    state_table[index] = 16'h0000;
```



Technology Independent Circuit is logic circuit which fully implements function described but is built from **Generic Boolean Gates**.

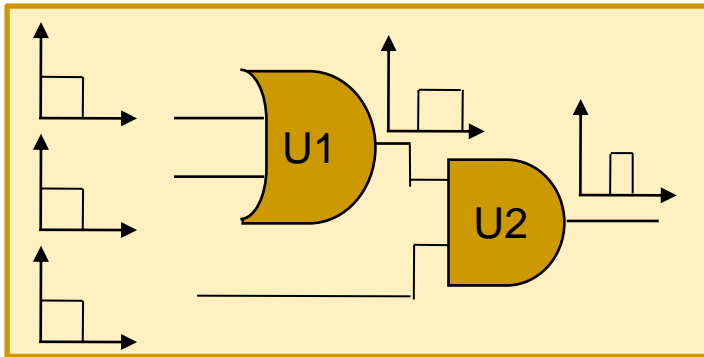
Constraint-Driven Synthesis



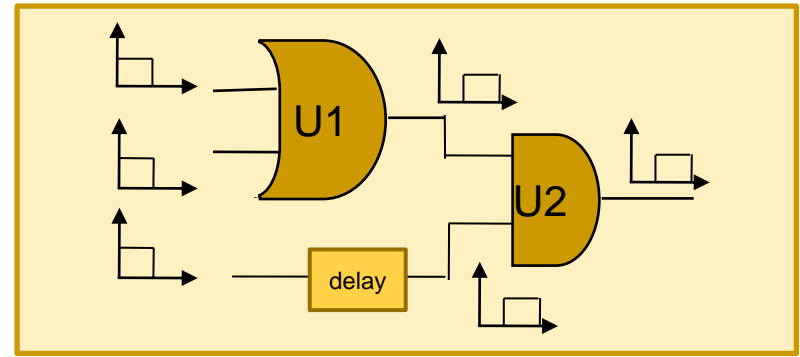
User can control synthesis process by providing constraints. Constraints are taken into account in the optimization step

Logic Synthesis (2)

- ❑ Logic synthesis also optimizes the circuit.
- ❑ The problem:
 - circuit simply created from function can possibly operate not as expected.



The delay of U1 element will affect final result

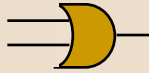




Additional elements should be added to the circuit to ensure correct operation

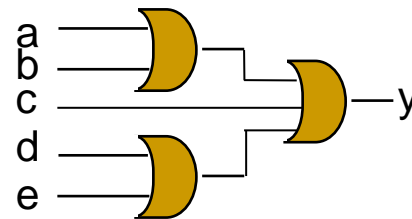
Main Optimization Trade-Offs

Circuit design is a trade-off of timing, power and area

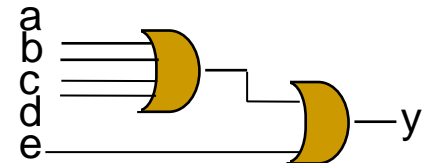
- ❑ Timing optimization
 - Goal: small delays
- ❑ Power optimization
 - Goal: low power consumption
- ❑ Area optimization
 - Goal: small area

Cell	Power
	2
	2.5
	3

Same function: $Y=a+b+c+d$



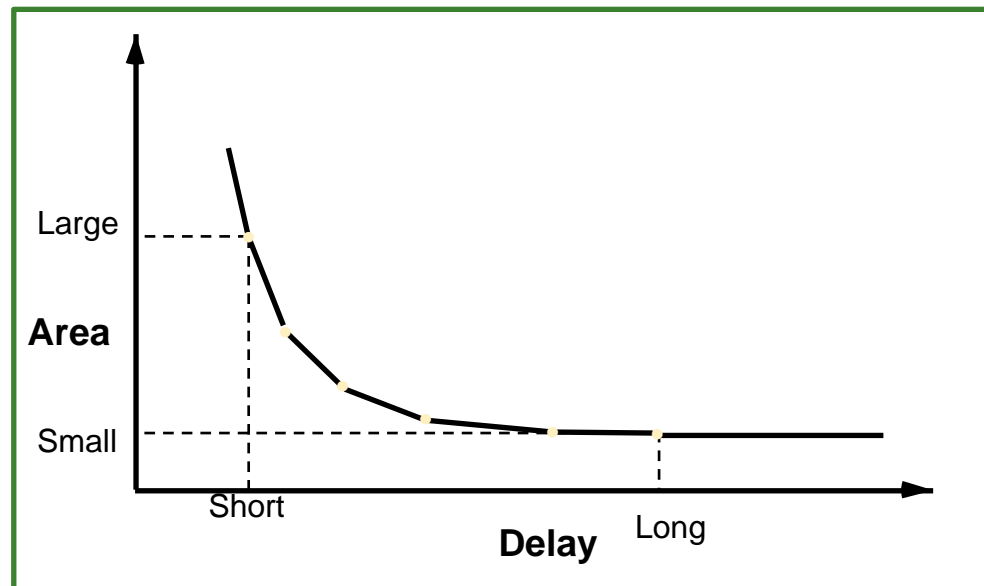
Total power: ~6



Total power: ~5

Parameter Trade-off

- ❑ Circuit is optimized to meet all constraints but timing has greater priority



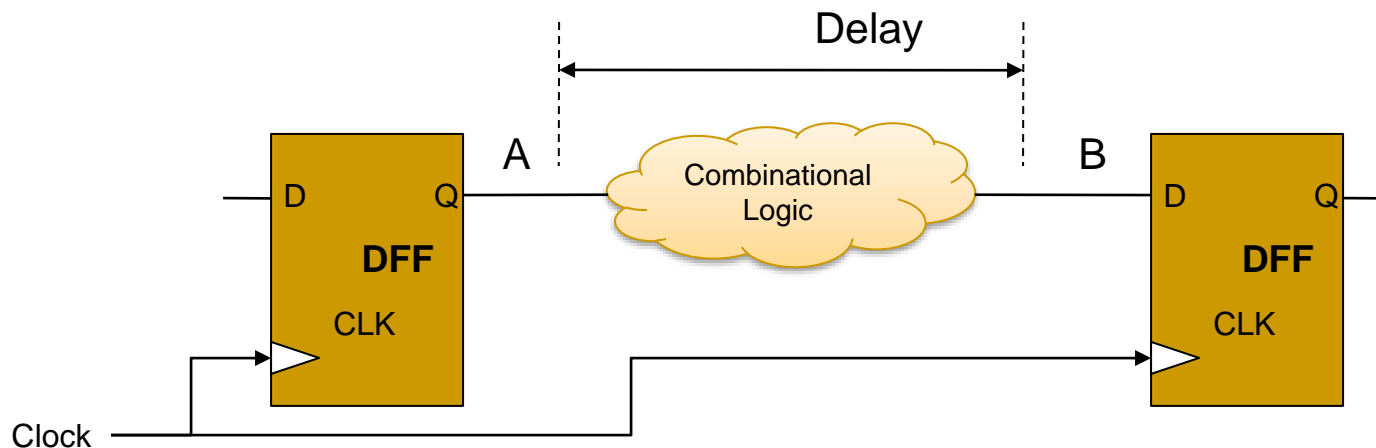
Area Constraints

- ❑ Area constraints are given by limiting maximum area value
- ❑ As timing has greater priority in Design Compiler it is used to set maximum area to zero, thus optimization achieves the best possible area with timings met

Timing Closure and Constraints

❑ Problem

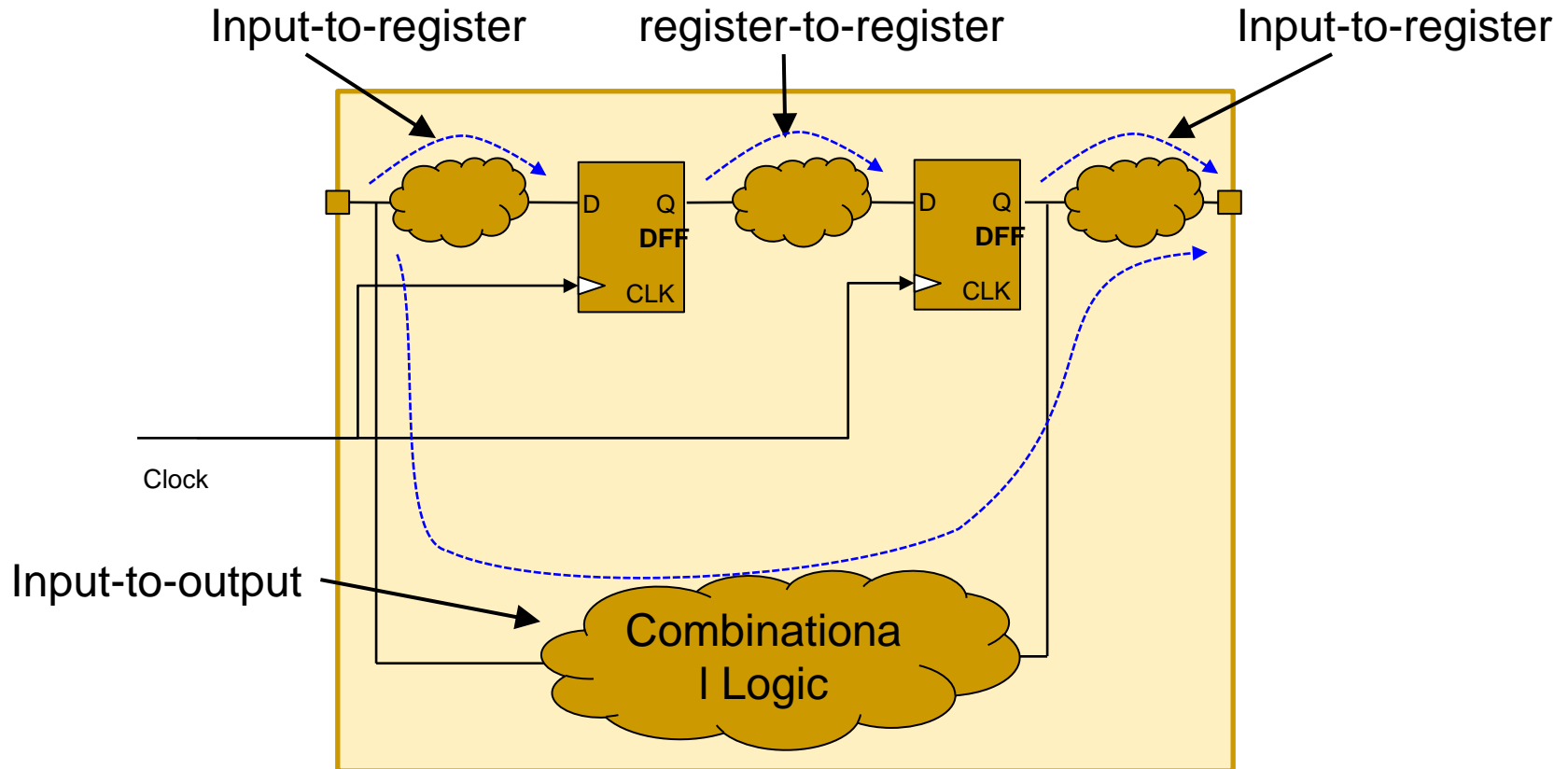
- In clocked environment signals on the register inputs must meet setup/hold requirements



If $\text{Delay} > T_{\text{clock}}$ setup time at point B is not met

Path Types

- There are 4 types of paths in a synchronous circuit



Timing Paths (Startpoint-Endpoint)

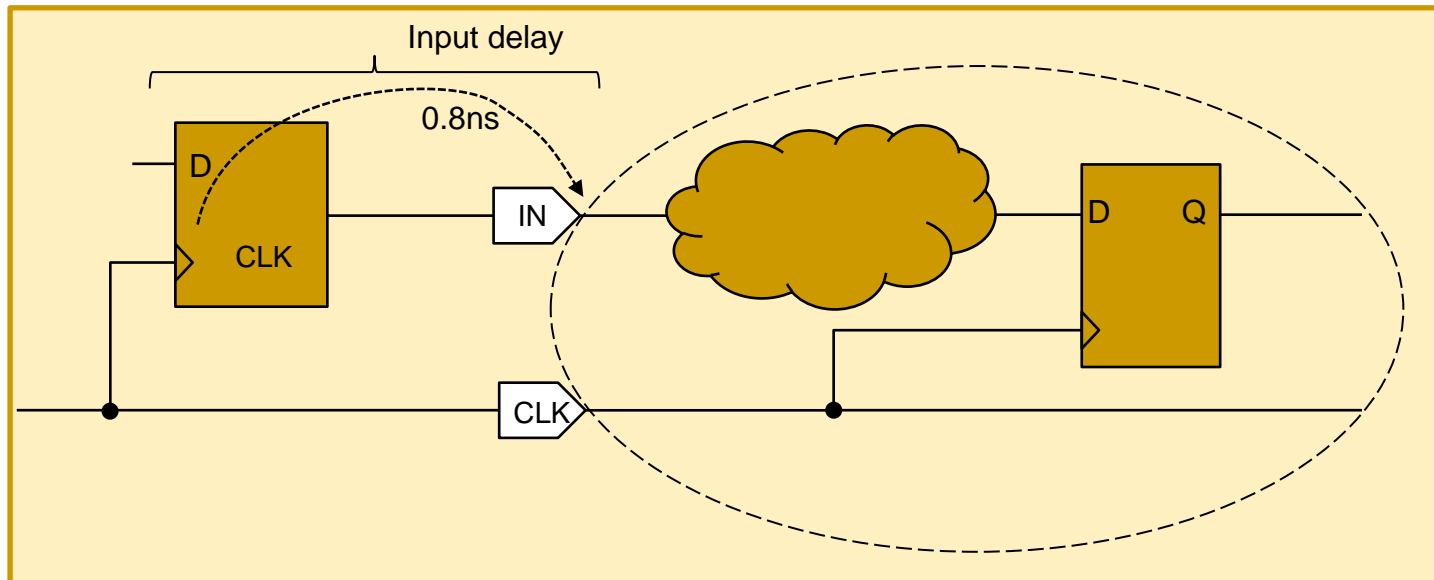
- ❑ Each timing path has a startpoint and an endpoint
- ❑ The startpoint is a place where data is **launched** by a clock edge:
 - ❑ a sequential element's clock pin
 - ❑ an input port of the design
- ❑ Data is propagated through the path and then **captured** at the endpoint by another clock edge:
 - ❑ a sequential element's data input pin
 - ❑ an output port of the design

Constraining Paths on Boundaries

- ❑ Clock set constraints on all reg-to-reg paths
- ❑ For boundaries paths additional data is required:
 - For the input paths the **arrival time** of the data is unknown
 - For an output path of the design the **external logic delays** are unknown
- ❑ In order to analyze the input-to-register and register-to-output timing, **external timing conditions** must be specified
 - **input delay** is defined as external delay before input
 - **output delay** is defined as delay of circuitry between output and next register

Input Delay

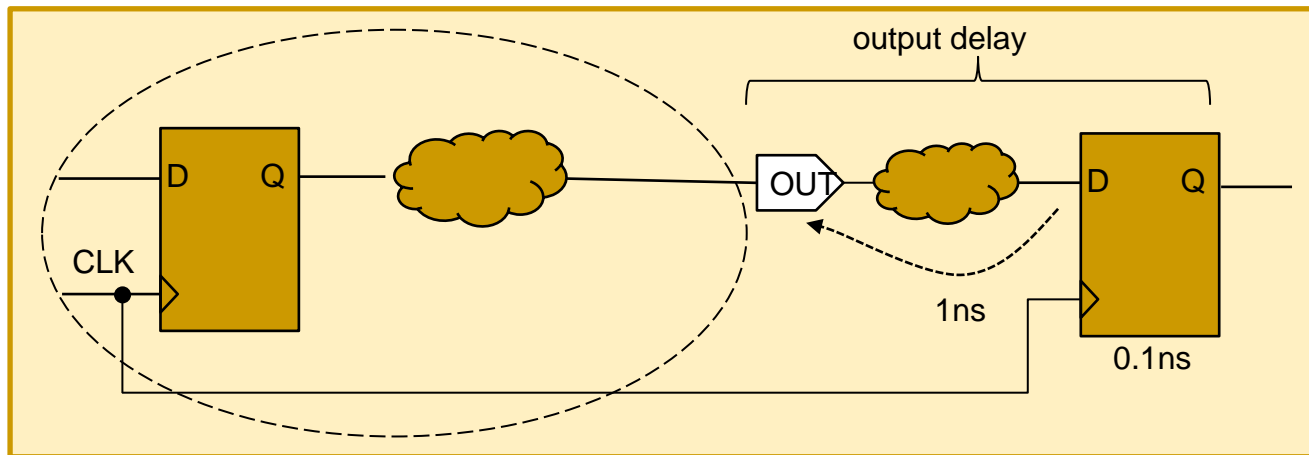
- ❑ The input delay is the CLK-to-IN external delay



- ❑ To define input delay `set_input_delay` command is used

Output Delay

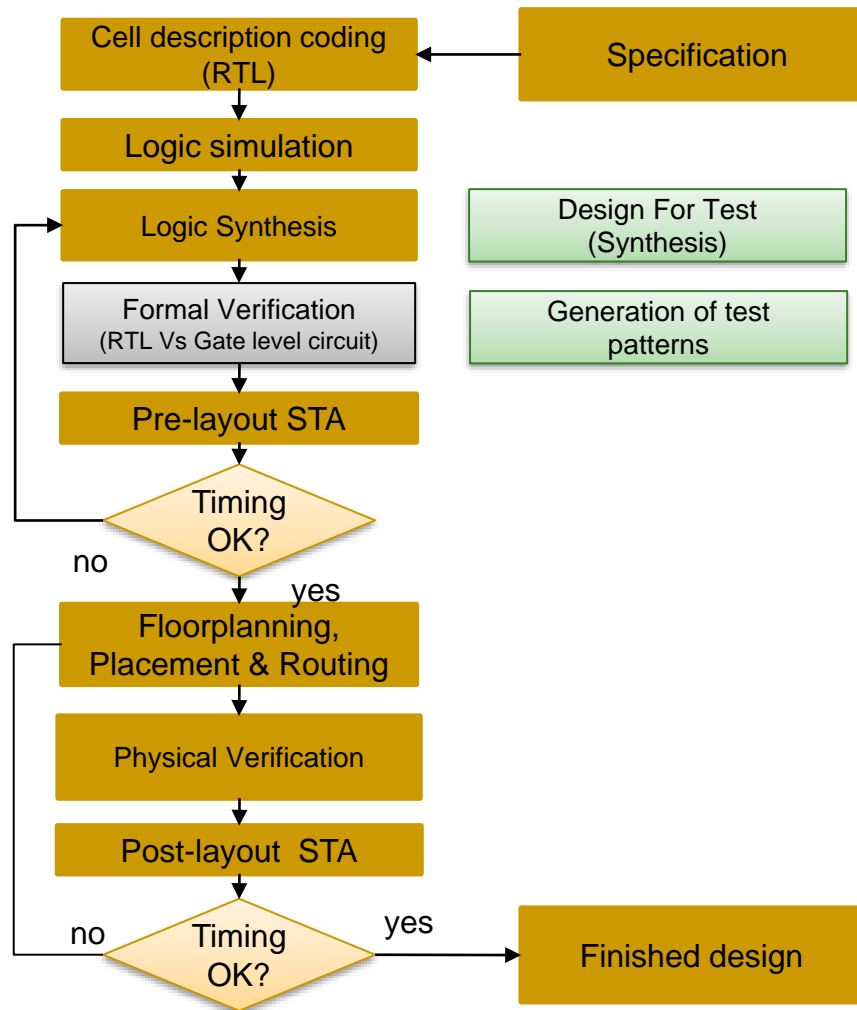
- ❑ The output delay includes the delay of the external buffer and the timing requirements for the external flip-flop



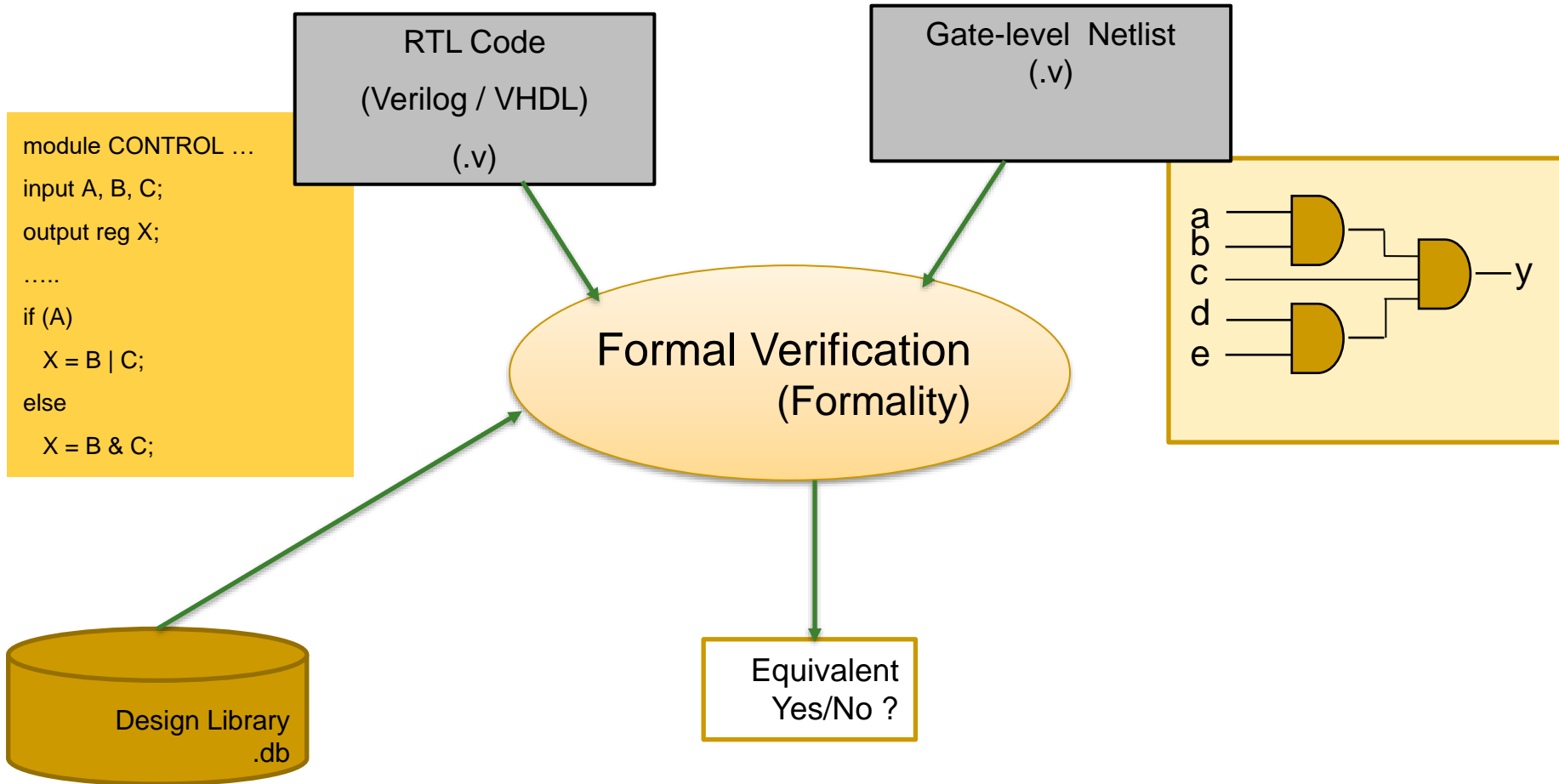
- ❑ To specify output delay *set_output_delay* command is used

$$\text{Output delay} = T_{\text{logic}} + T_{\text{setup}}$$

Digital Design Flow



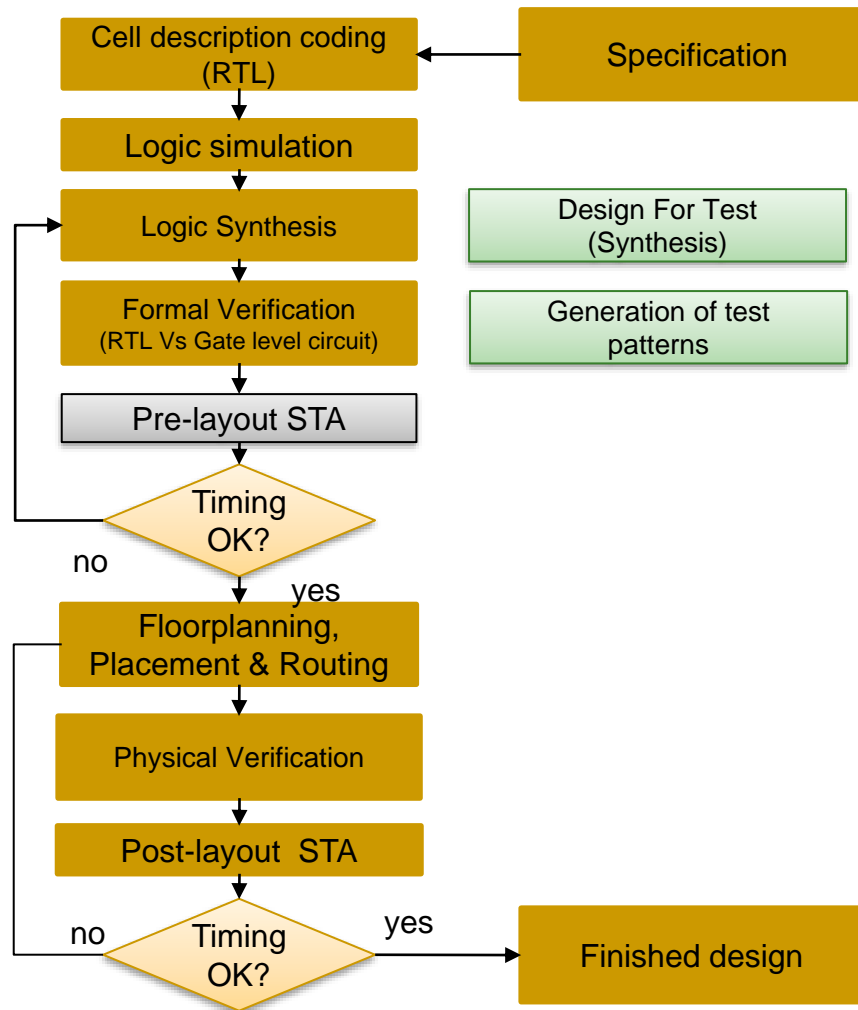
Formal Verification



Formal Verification (2)

- ❑ Checks whether two designs are functionally equivalent or not
- ❑ Its purpose is to detect unexpected differences that may have been introduced into a design during development

Digital Design Flow

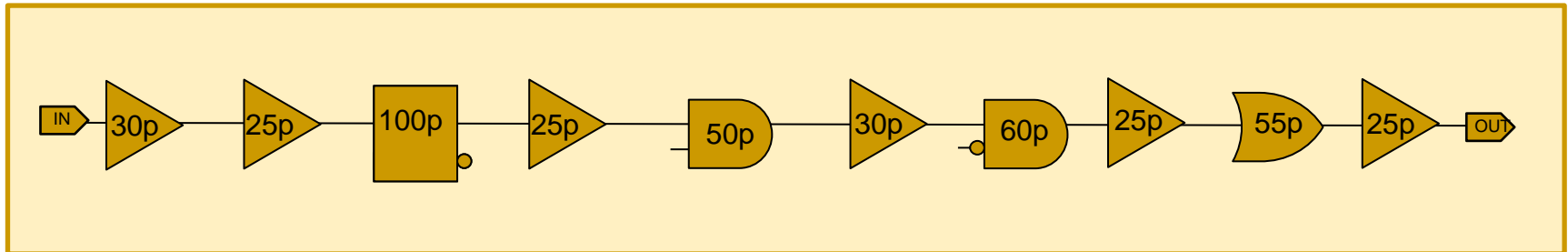


- ❑ The arrival time at the input is propagated through the gates at each level till it reaches the output



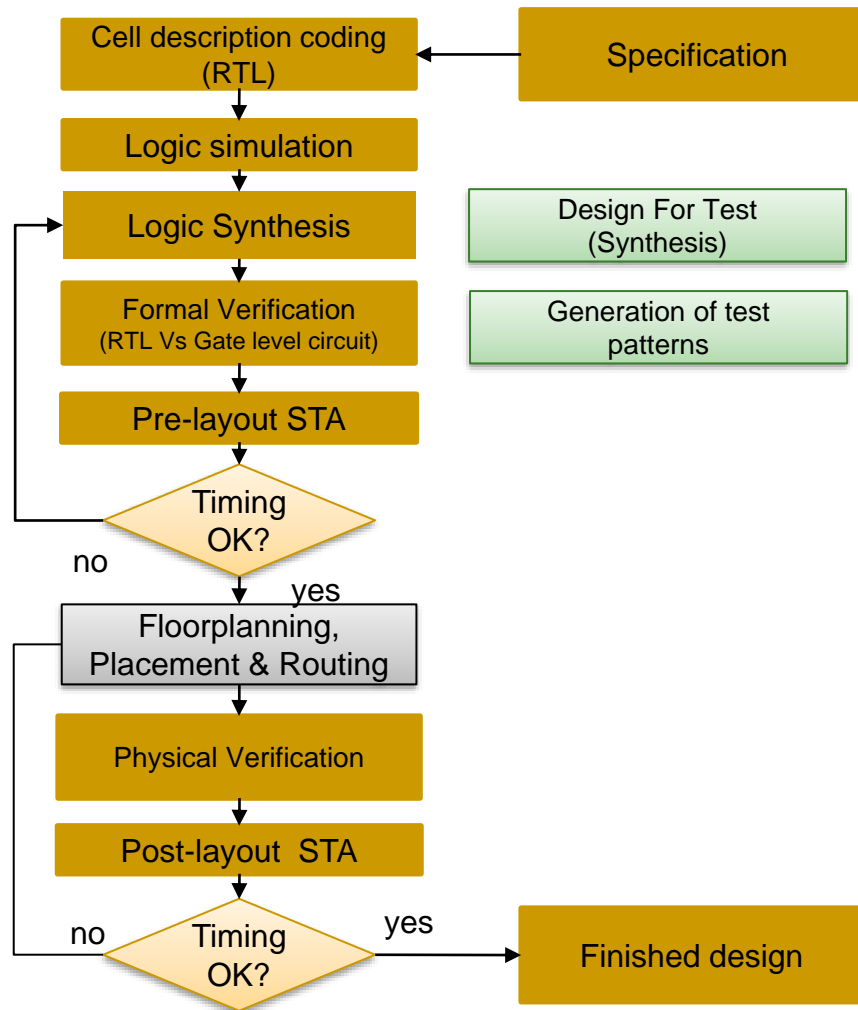
Example

- Static Timing Analysis (STA) is a method of computing the expected timing of a digital circuit without requiring simulation



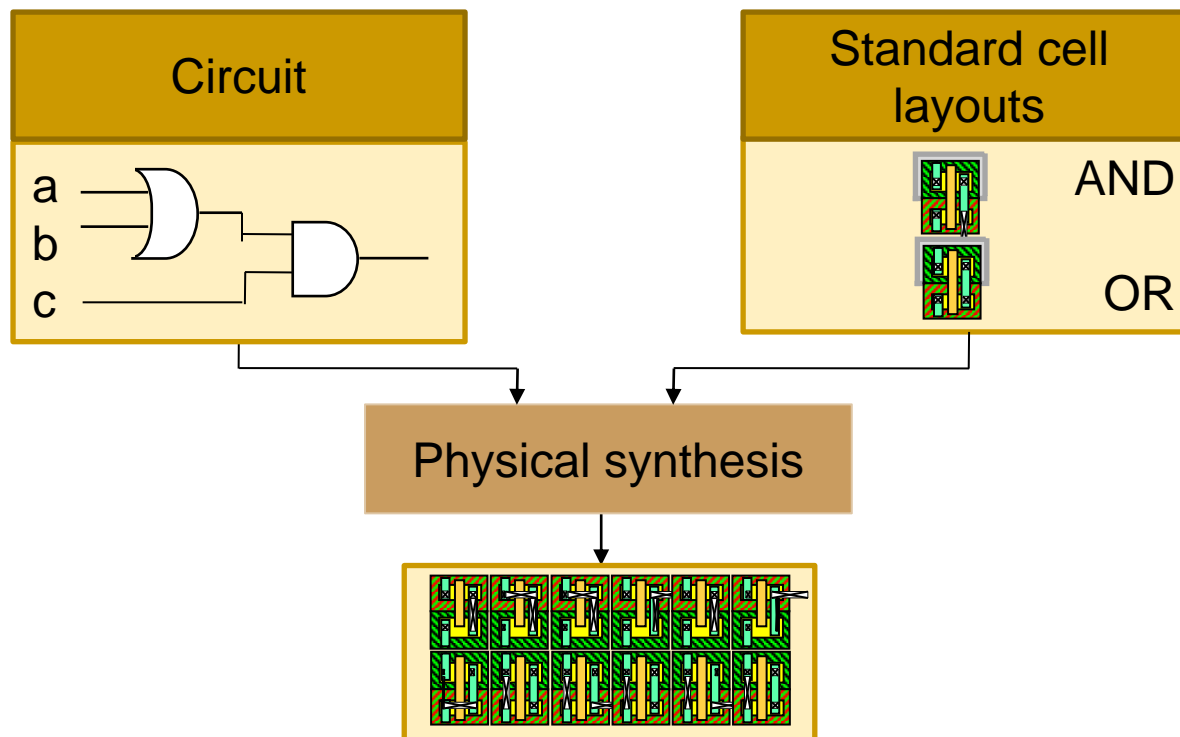
$$\text{Delay} = 30p + 25p + 100p + 25p + 50p + 30p + 60p + 25p + 55p + 25p = 425p$$

Digital Design Flow



Physical Synthesis

- Physical synthesis is the process that produces layout of logic circuit.

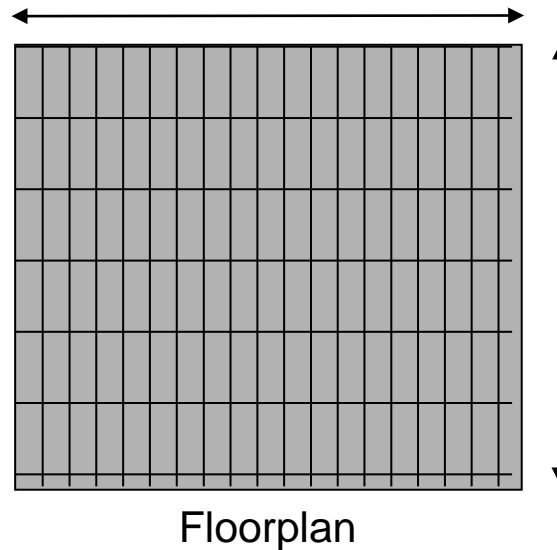


Physical Synthesis Steps

- ❑ Floorplanning
- ❑ Placement
- ❑ Routing

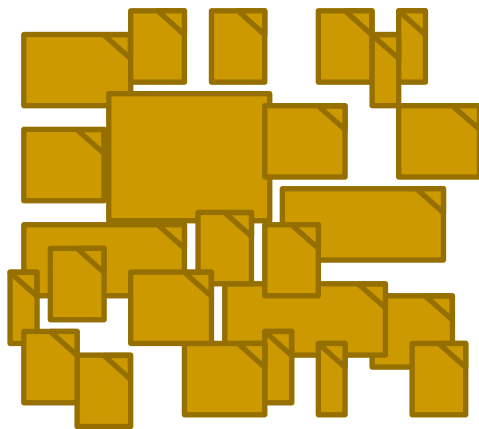
Floorplanning

- During the floorplanning step the overall cell is defined, including: cell size, supply network, etc.

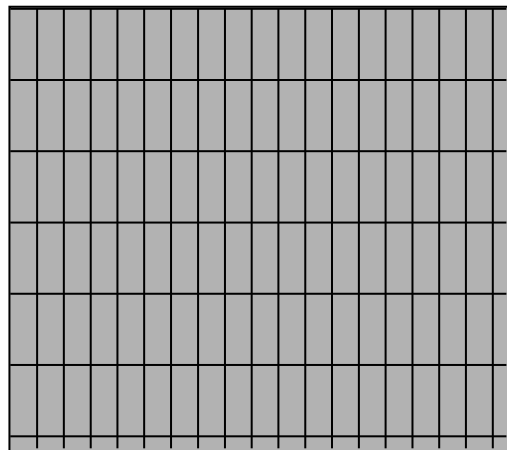


Placement

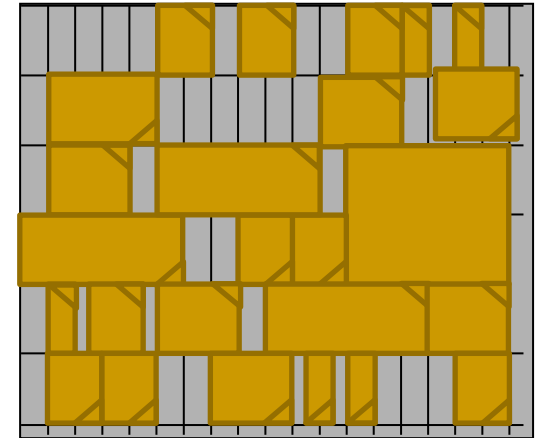
- ❑ **Placement** – exact placement of modules (modules can be standard cells, IPs).
- ❑ The goal is to minimize the total area and interconnect length



Cells from a circuit



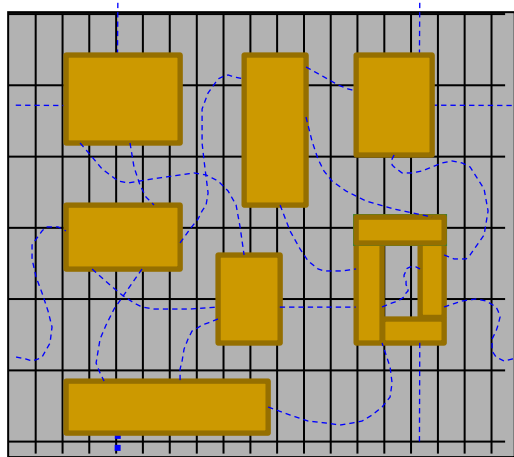
Floorplan



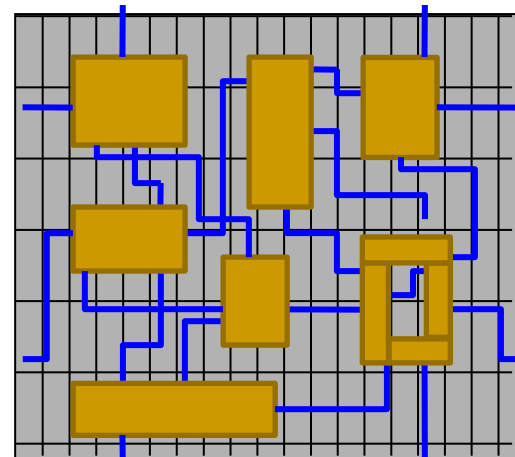
Placed design

Routing

- ❑ Routing connects placed cells according to schematic
- ❑ The goal is minimal impact of interconnects on circuit operation



Placed design



Routed design

Circuit Optimization During Physical Synthesis

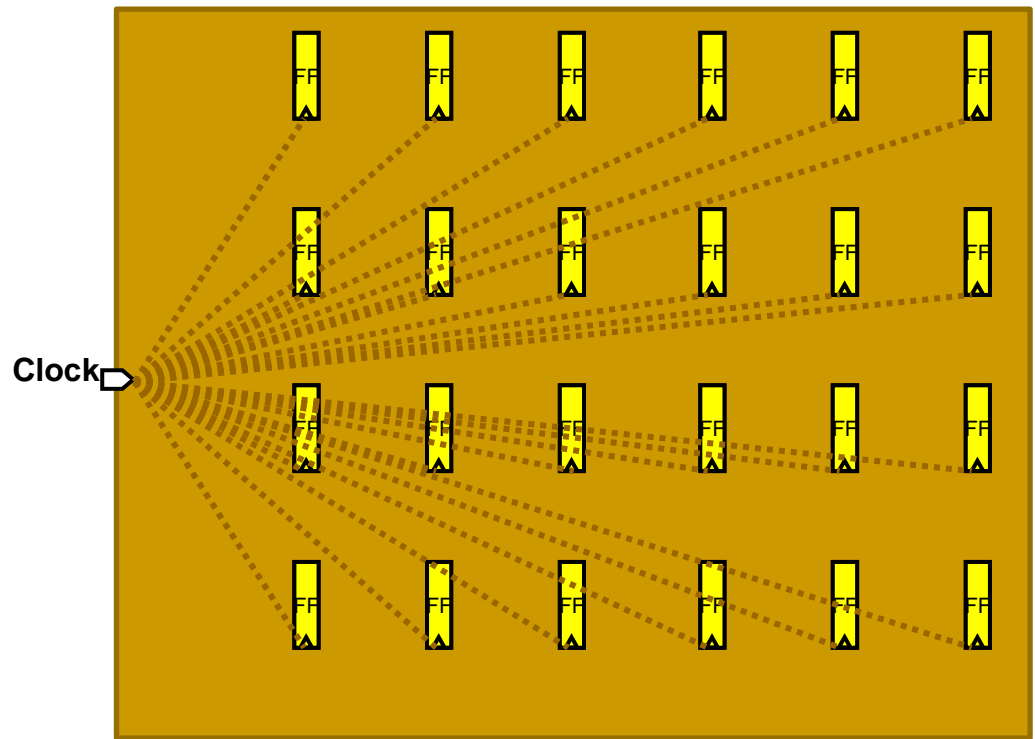
- ❑ Physical synthesis not only places and routes the cells of a circuit but also optimizes the cell as required by the designer

- ❑ Optimizations
 - Area
 - Interconnect length
 - Power density
 - Clock distribution
 - etc.

Physical Synthesis Circuit Optimization

Example: Clock Delay Problems

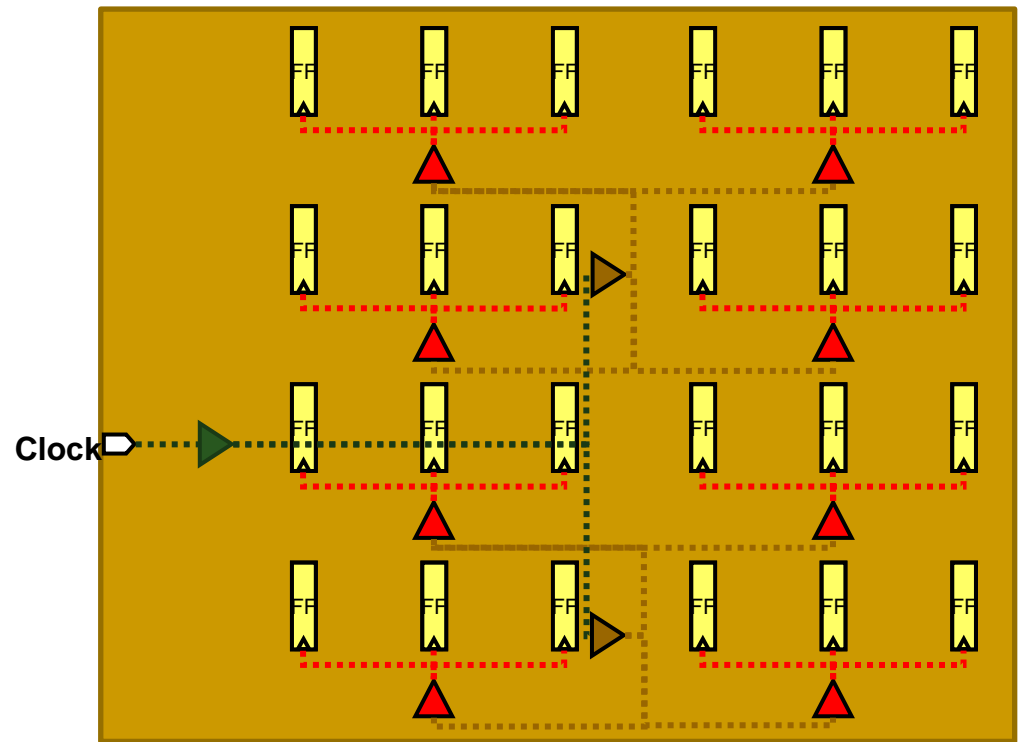
- ❑ All clock pins are driven by a single clock source



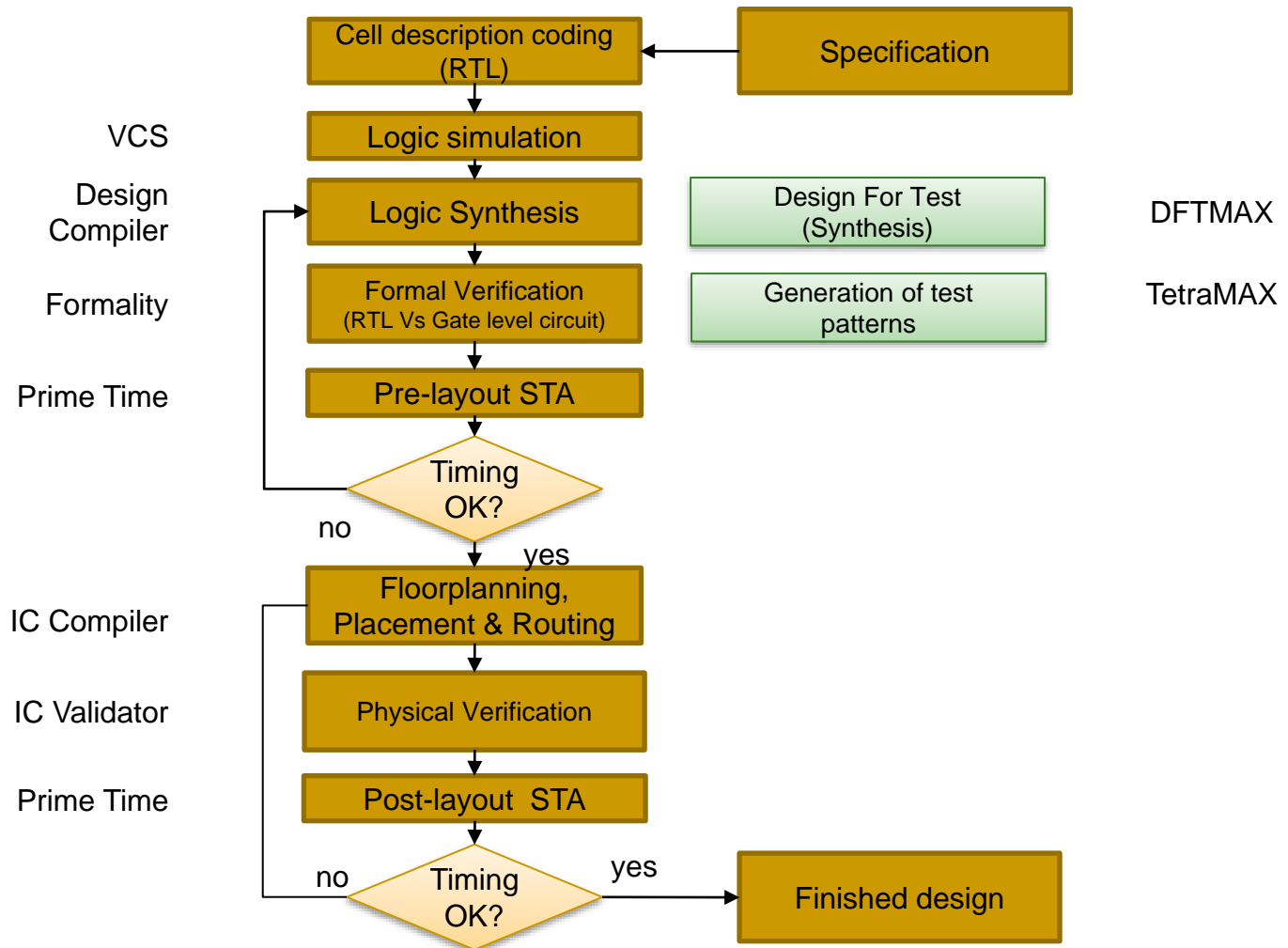
Physical Synthesis Circuit Optimization

Example: Clock Delay Problems

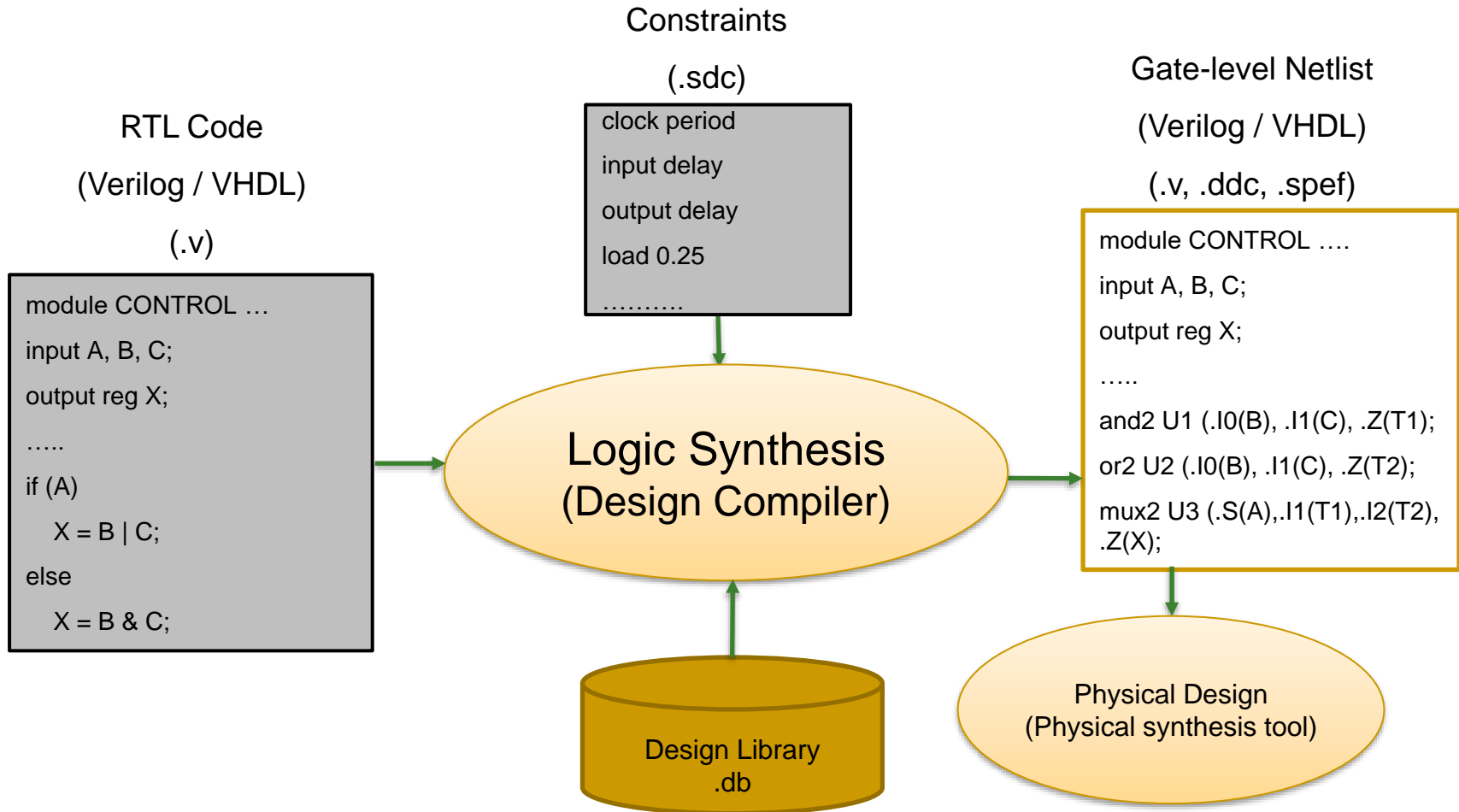
- A buffer tree is built to balance the loads and minimize the delays



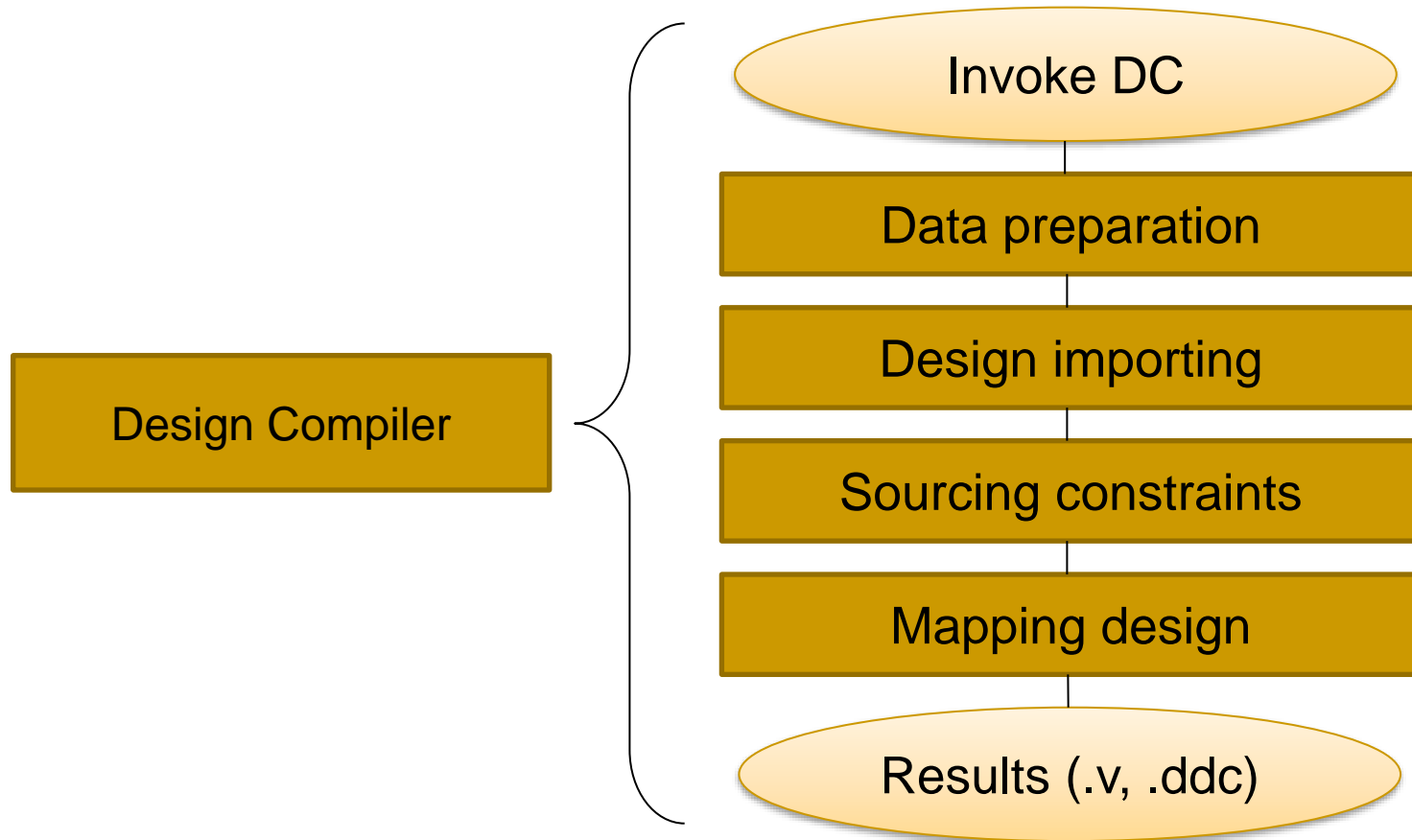
Digital Design Toolchain



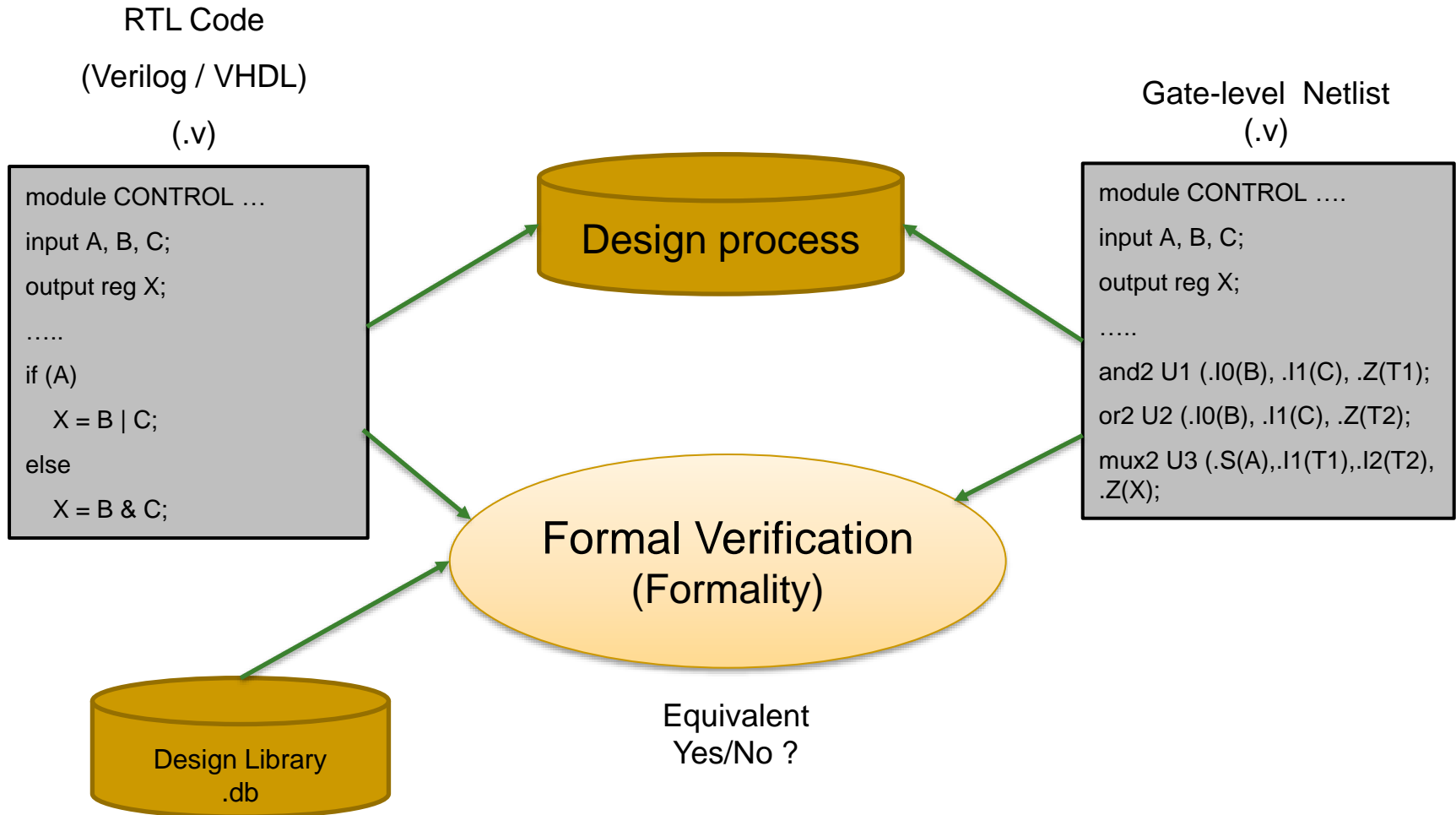
Design Environment of Logic Synthesis



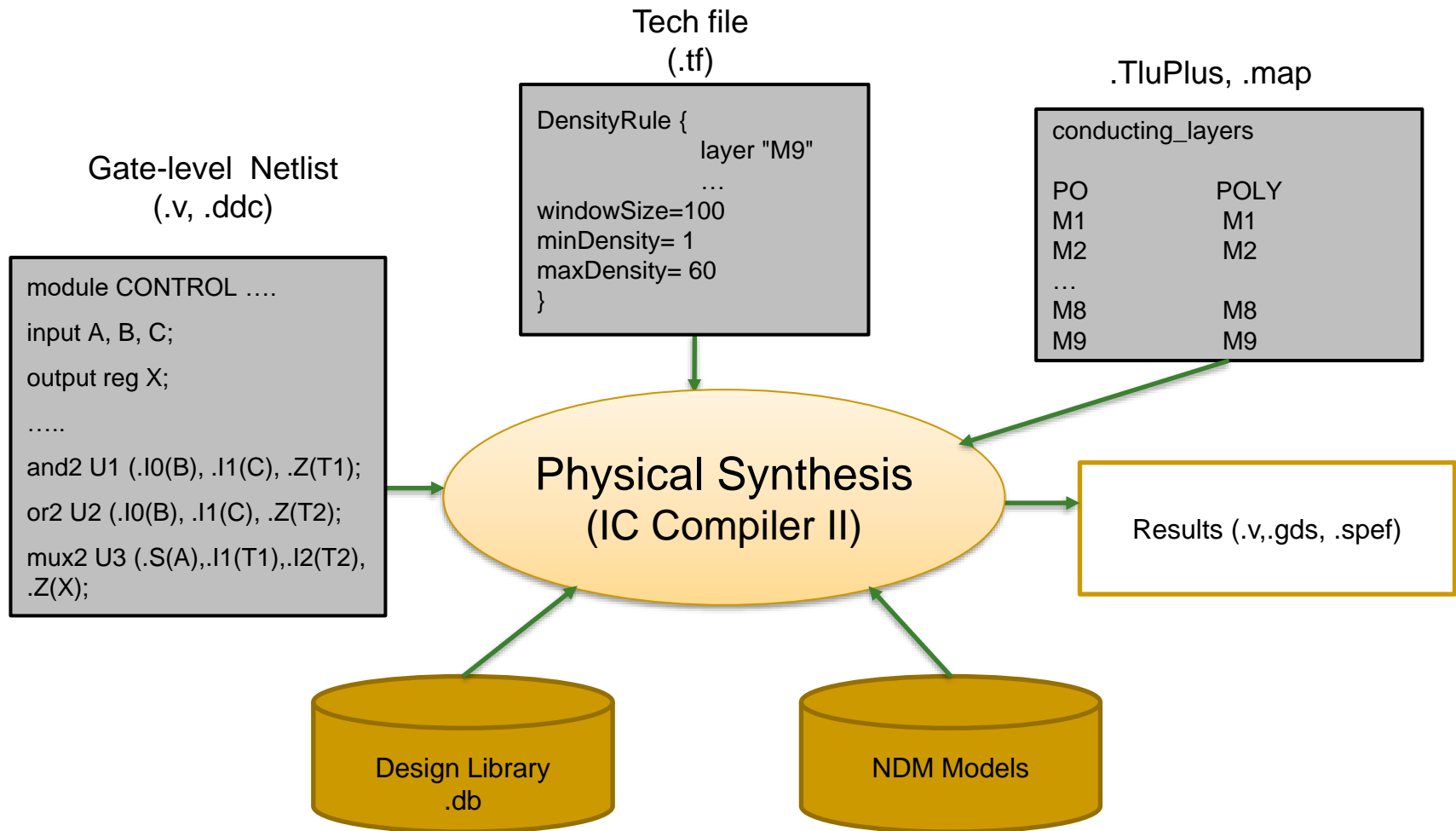
Design Compiler Steps



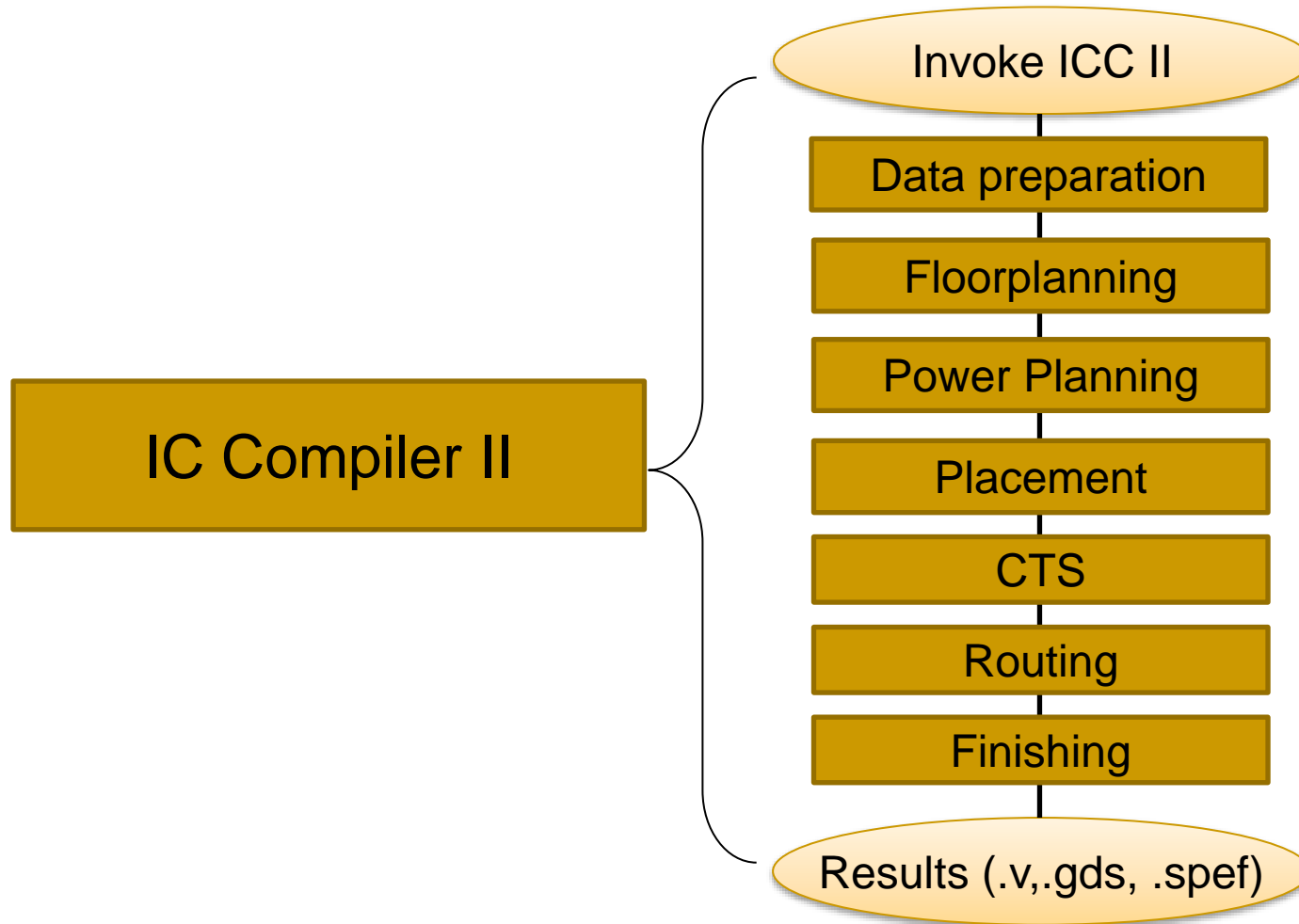
Formal Verification



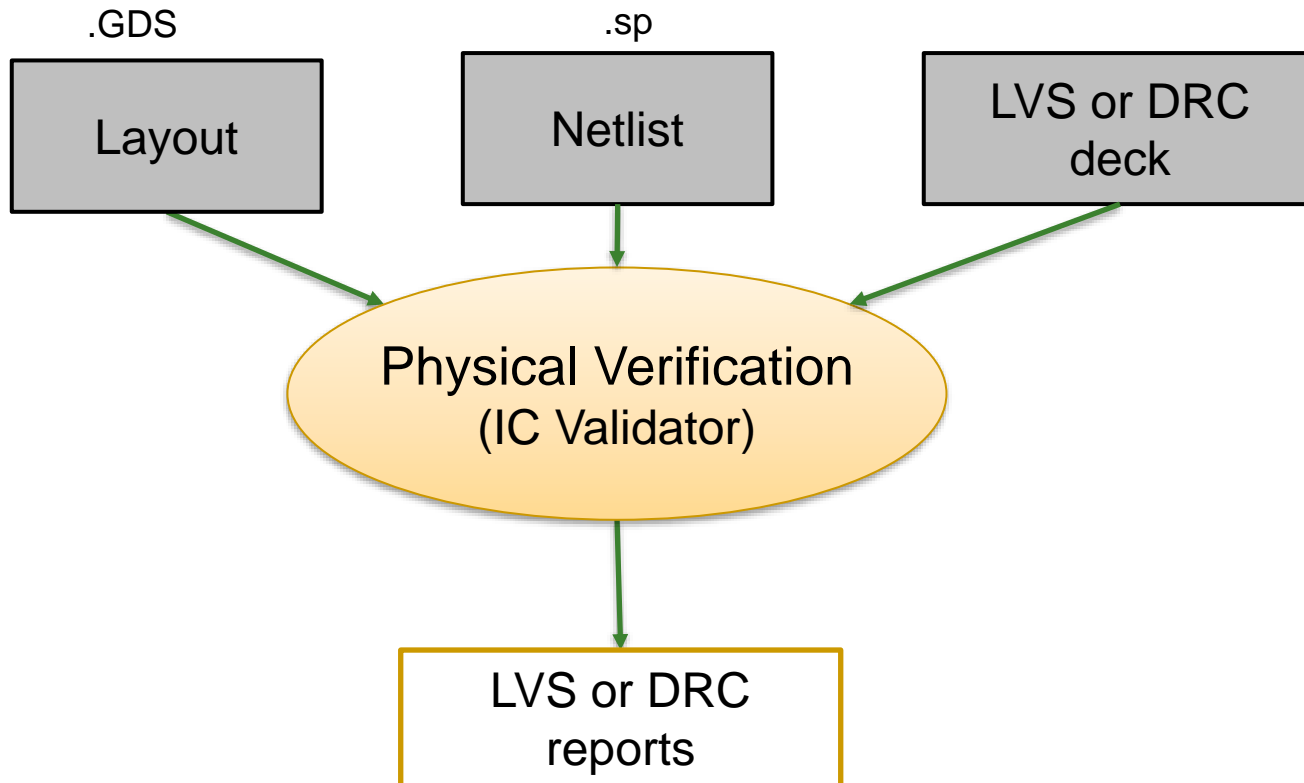
Design Environment of Physical Synthesis



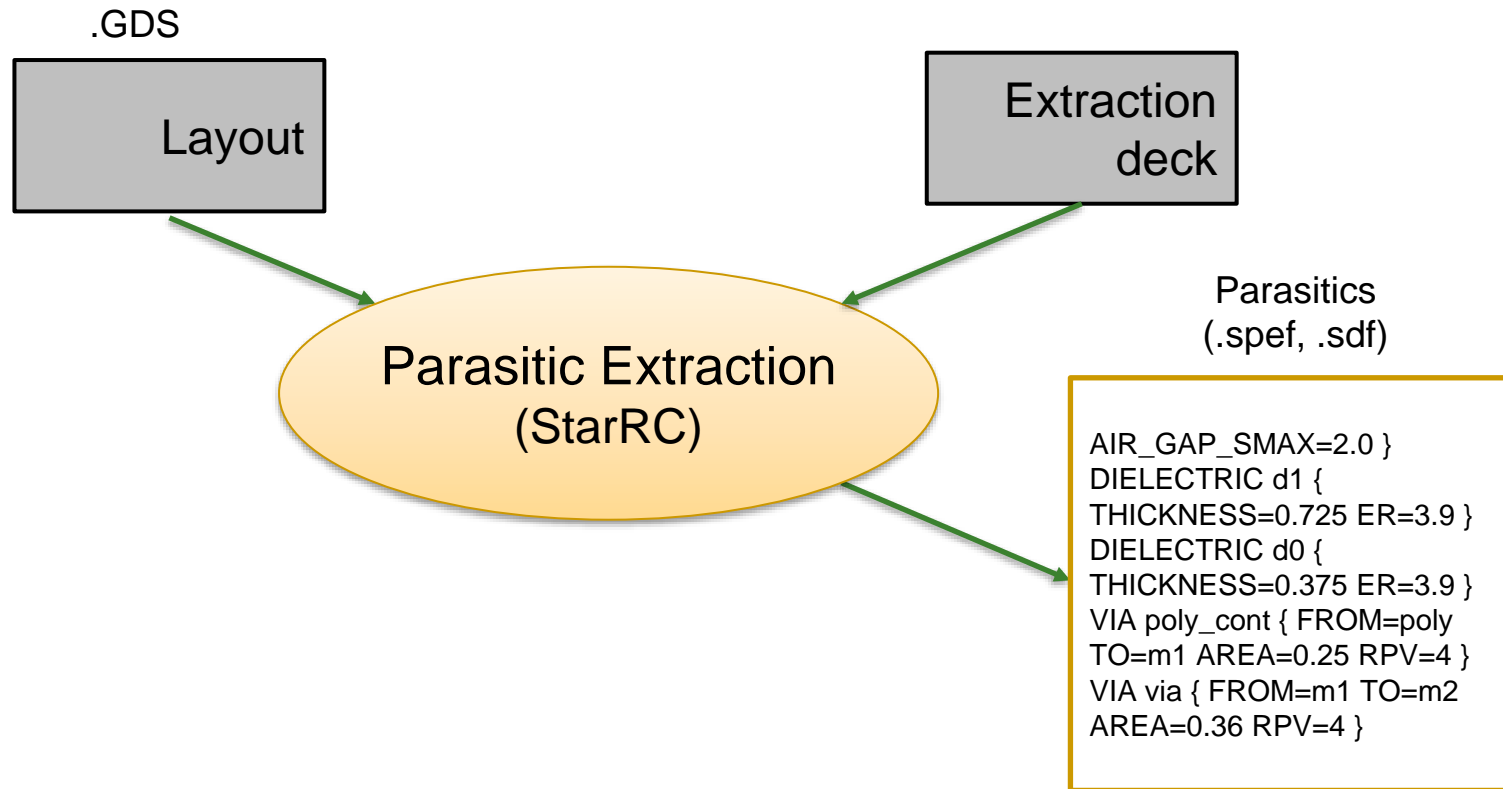
Physical Synthesis Steps



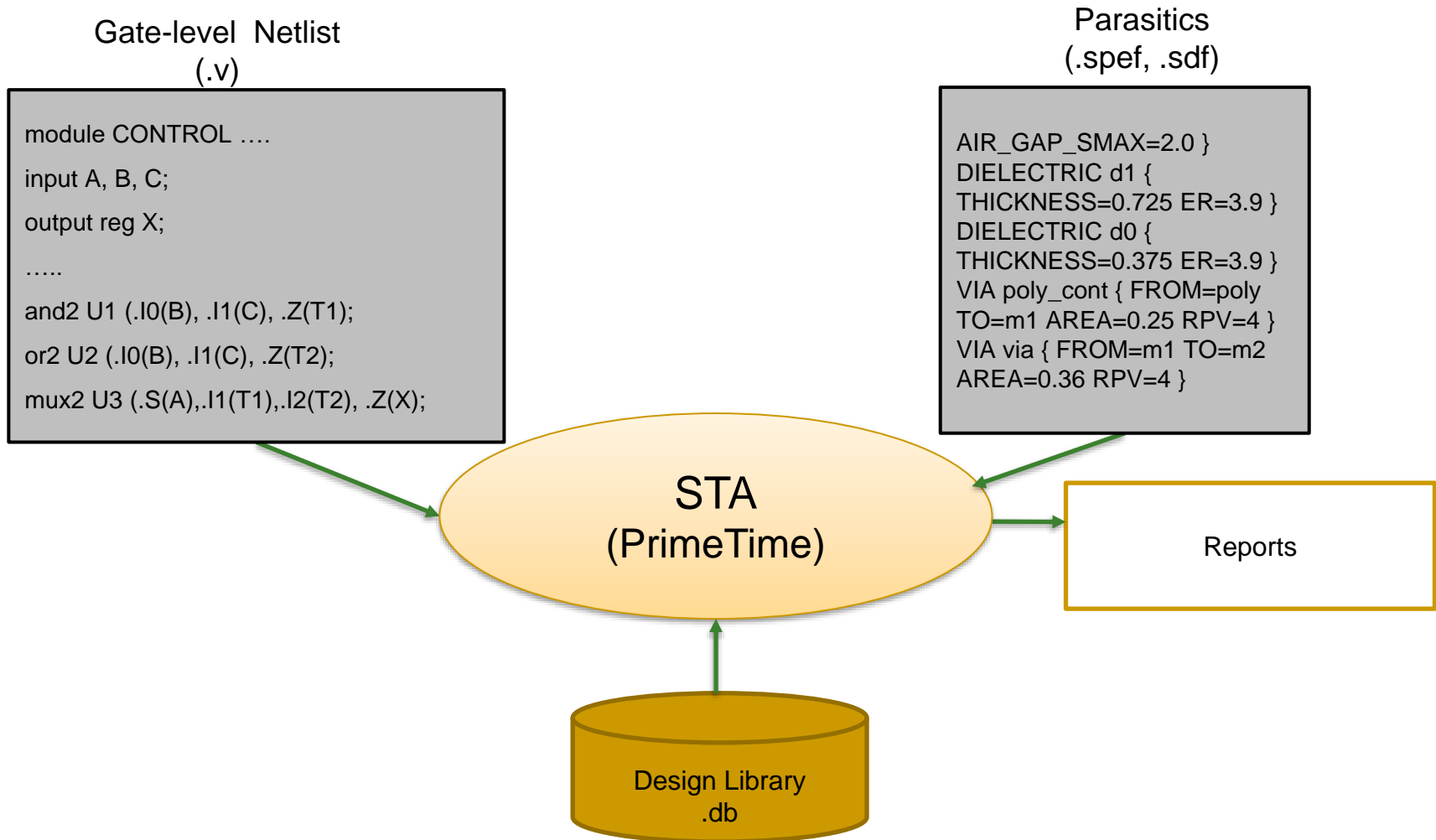
Physical Verification



Parasitic Extraction



Static Timing Analysis (STA)



Power Analysis

