Topic 7.1

ASIC Design Flow I

Xinfei Guo
xinfei.guo@sjtu.edu.cn

November 4th, 2021

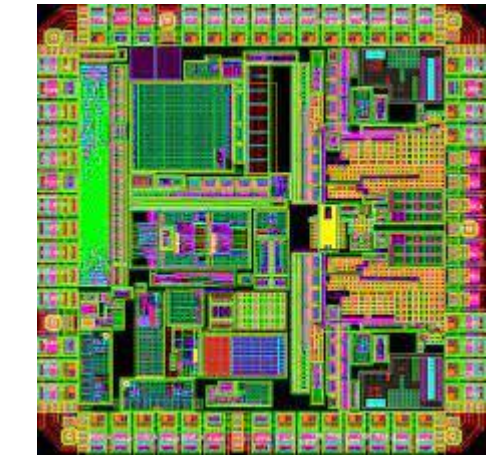
T7 learning goals

- How to design a Chip (SoC) from concept to silicon?
  - **Full design flow from RTL to Layout**
  - **How to make decisions at each step**

```
module PE (clock, R, S1, S2, S1S2mux, newDist, Accumulate, Rpipe);
input clock;
input [7:0] R, S1, S2;// memory inputs
input S1S2mux, newDist;// control inputs
output [7:0] Accumulate, Rpipe;
reg [7:0] Accumulate, AccumulateIn, Difference, Rpipe;
reg       Carry;

always @(posedge clock) Rpipe <= R;
always @(posedge clock) Accumulate <= AccumulateIn;

always @(R or S1 or S2 or S1S2mux or newDist or Accumulate)
  begin  // capture behavior of logic
   difference = R - S1S2mux ? S1 : S2;
   if (difference < 0) difference = 0 - difference;
// absolute subtraction
    {Carry,AccumulateIn} = Accumulate + difference;
    if (Carry == 1) AccumulateIn = 8'hFF;// saturated
    if (newDist == 1) AccumulateIn = difference;
// starting new Distortion calculation
  end
endmodule
```

Motion Estimator Processing Element (PE).

RTL



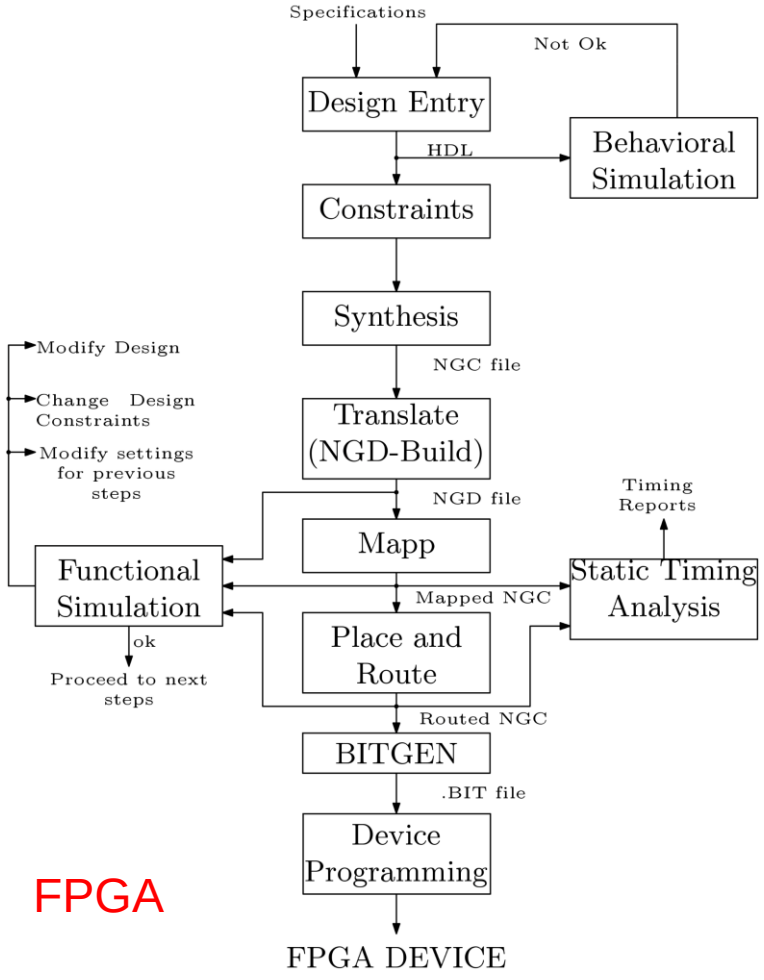
Layout

Figure: Synopsys

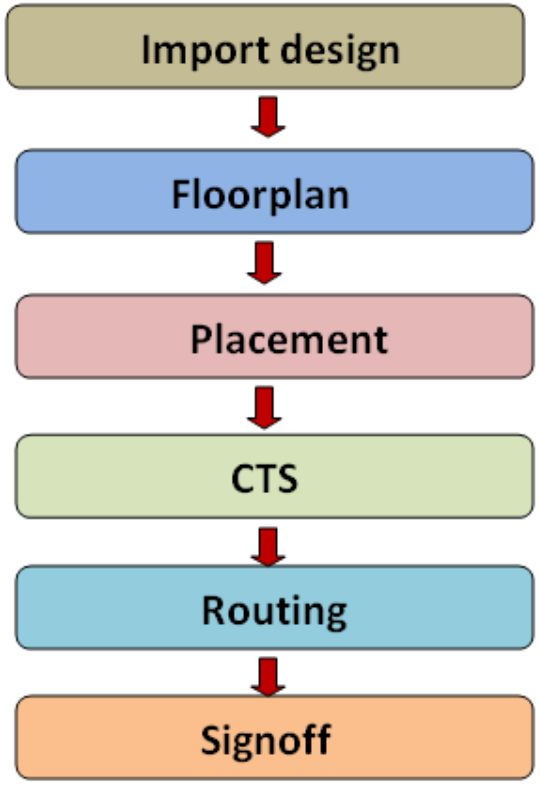
Disclaimer

- These materials are not usually taught at universities, they integrate a lot of my personal industry experiences, thus there is no good textbook for it. Slides, labs and internet are the best resources;
- This course is about the flow (tool usage), not about how the tool works (*VE527 Computer-Aided Design of Integrated Circuits* will teach algorithms behind the tools);
- Design flow is highly tool/design dependent, this course will focus on main ideas only, it will not provide any preferences on any tools;
- ASIC design flow (physical design) is never as easy as just “click a button”, it involves a lot of design decisions and experiences, this course can only provide basic ideas.

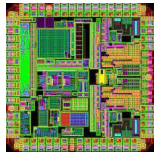
FPGA flow vs. ASIC flow



FPGA

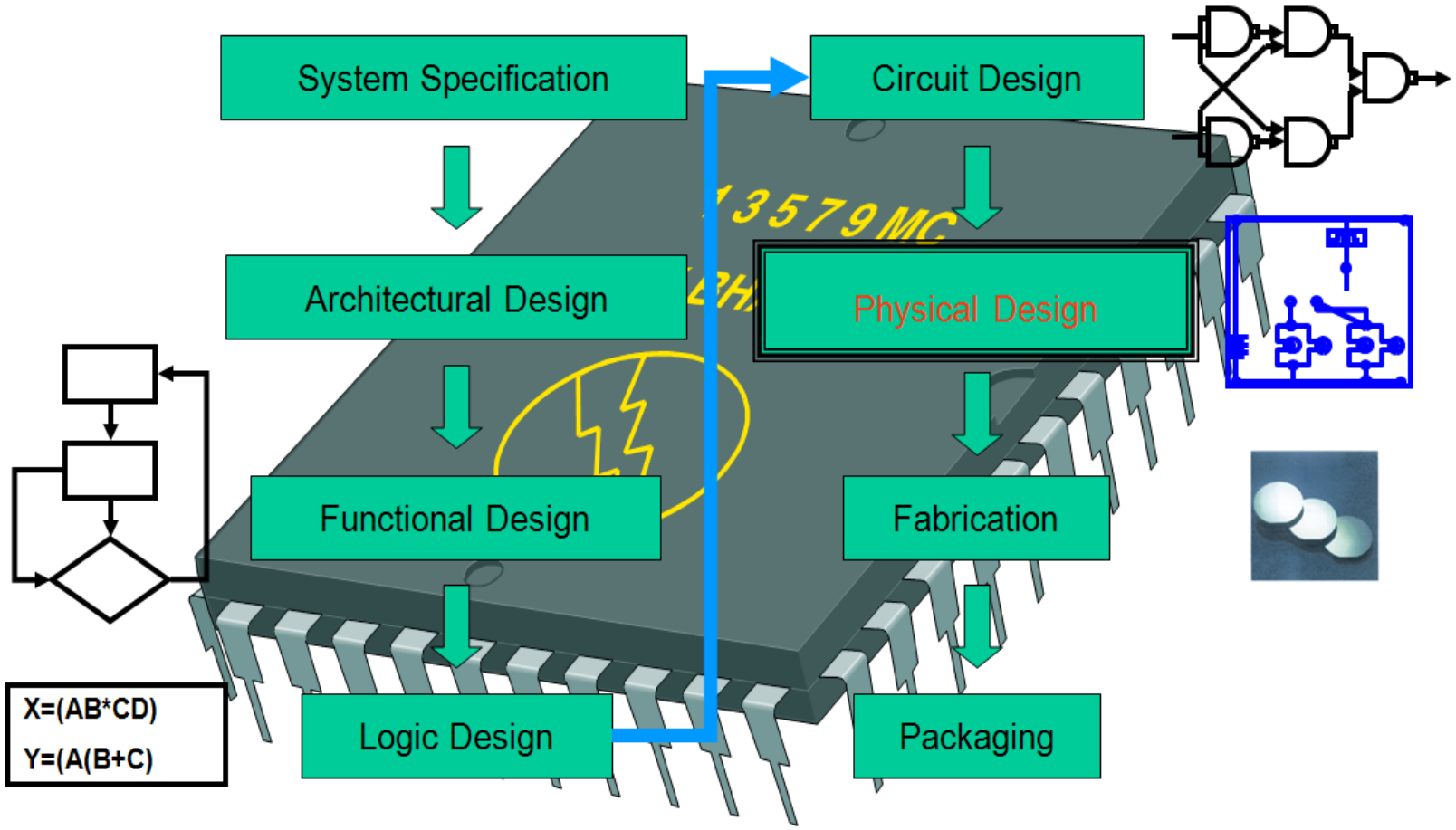


ASIC



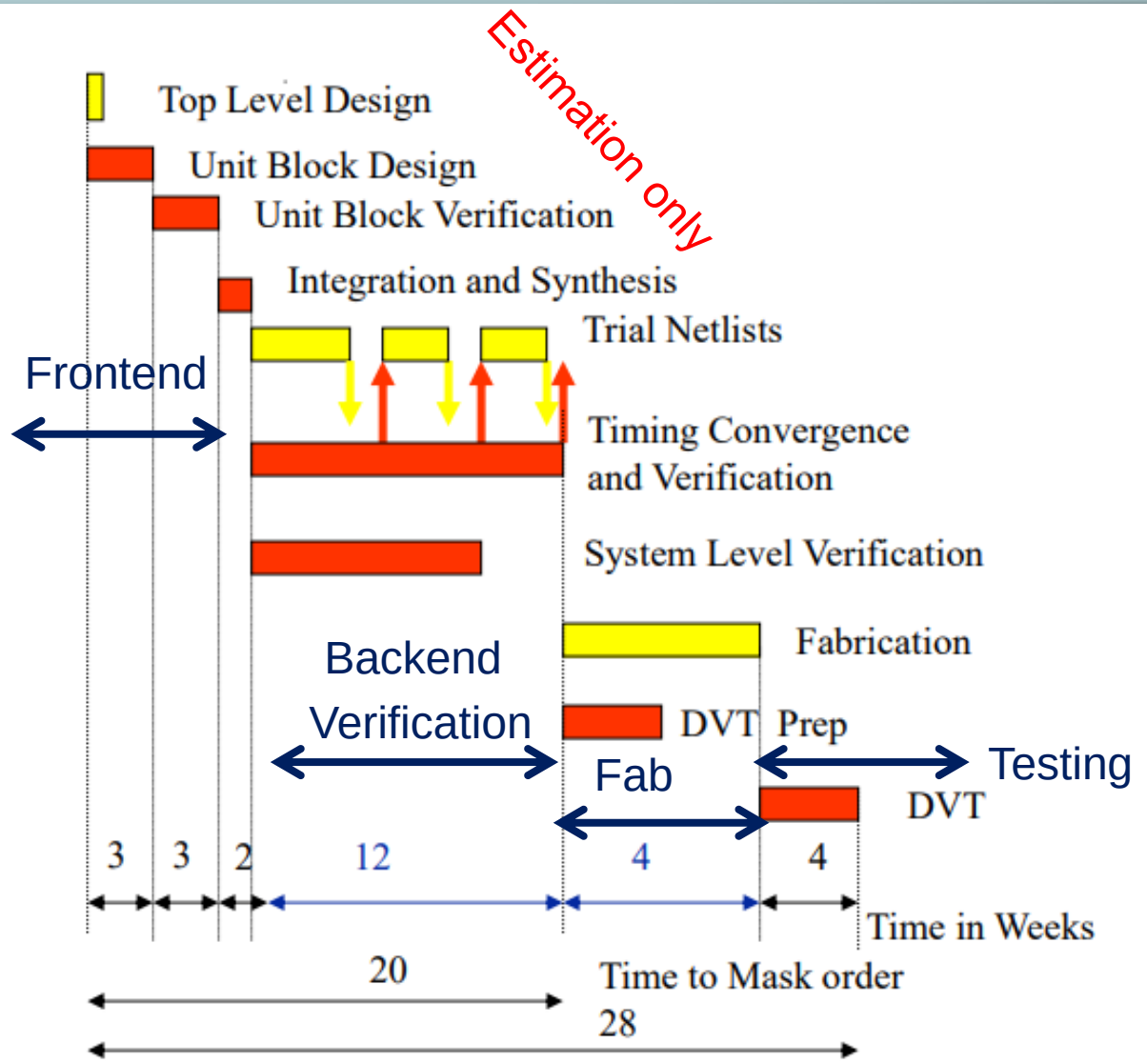
Images: https://digitalsystemdesign.in/fpga-implementation-step-by-step/
https://www.allaboutvlsi.in/2020/12/physical-design-flow.html

VLSI Design Cycle



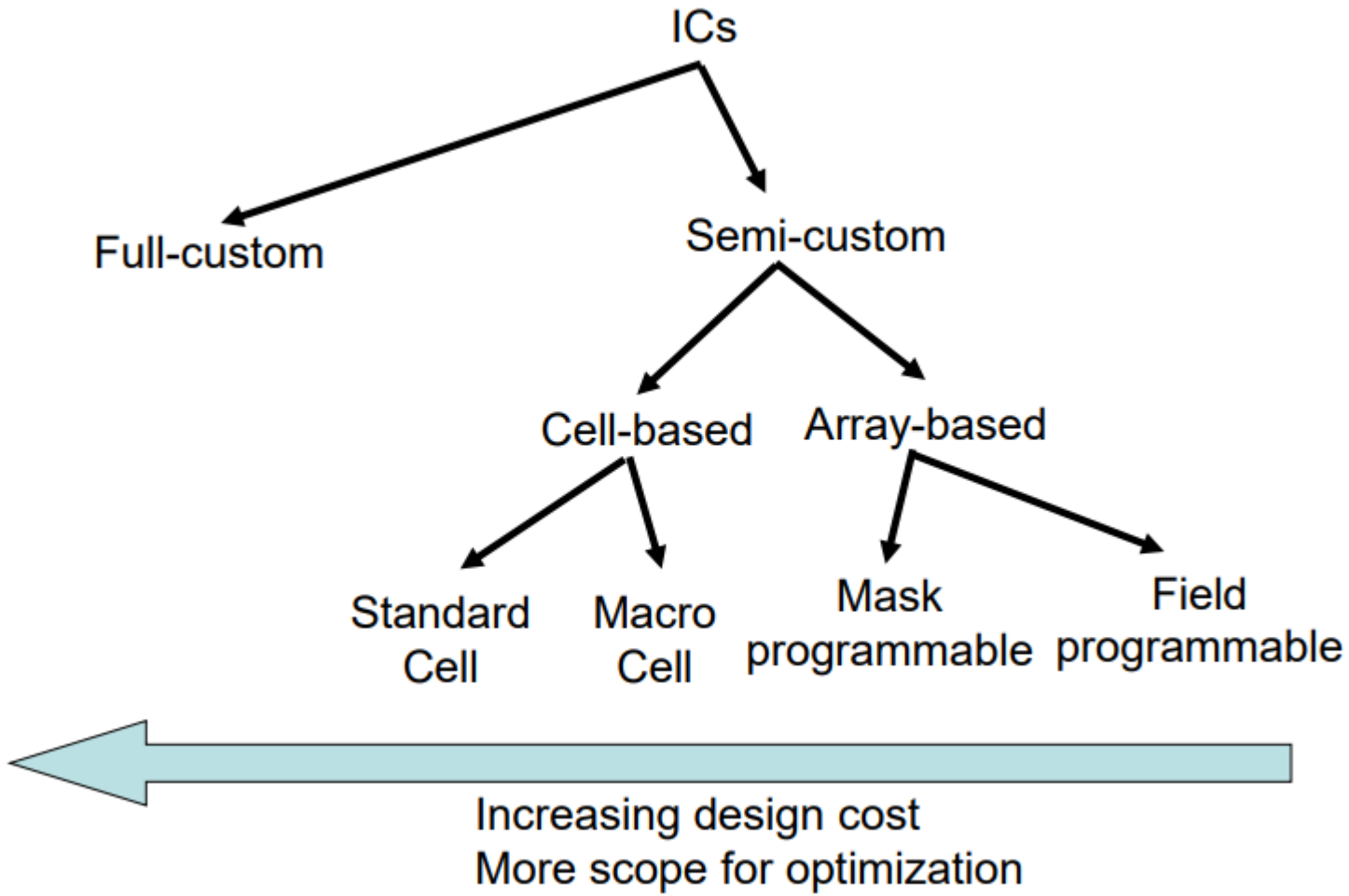
source: Synopsys

Recap: SoC Design Cycle



source: https://www.ee.ryerson.ca/~courses/coe838/

Taxonomy of Integrated Circuits



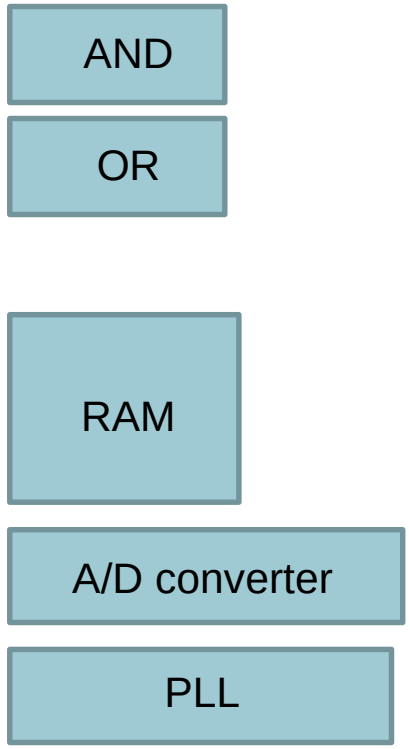
source: https://nanohub.org/courses/ECE695R/

Full-Custom Design

- Logic and physical design of every device is optimized

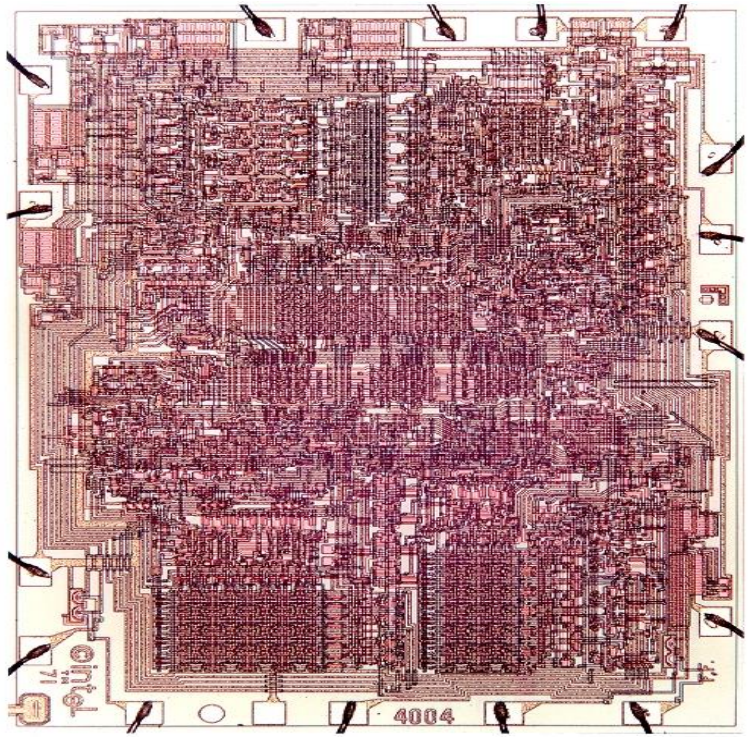
• Best speed/power/area tradeoff	• High design cost • Long time to market

- Reasons to use:
  - No means for semicustom design (PLL, A/D convertors)
  - Design will be reused (library cell, IP) in semicustom designs
  - Large volume production (memories)
  - Performance have larger priority than cost and time-to-market (supercomputers)



source: Synopsys

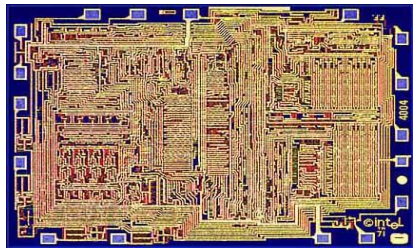
The Custom Approach



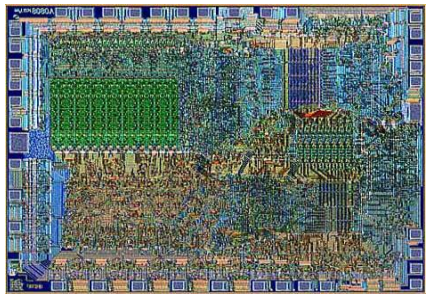
Intel 4004

Courtesy Intel

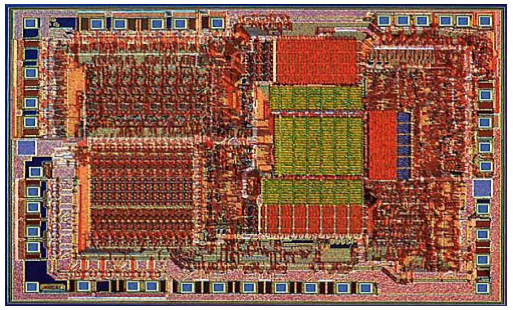
Transition to Automation and Regular



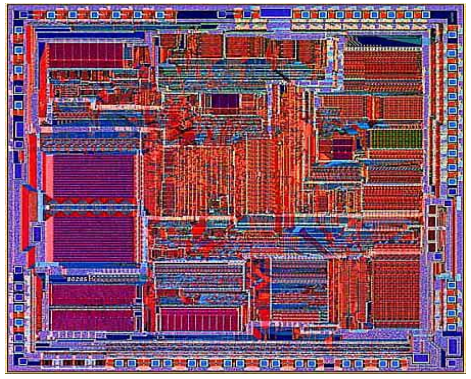
Intel 4004 (‘71)



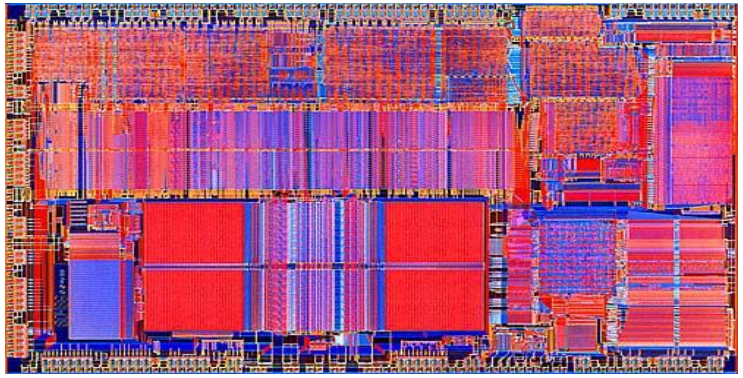
Intel 8080



Intel 8085



Intel 8286

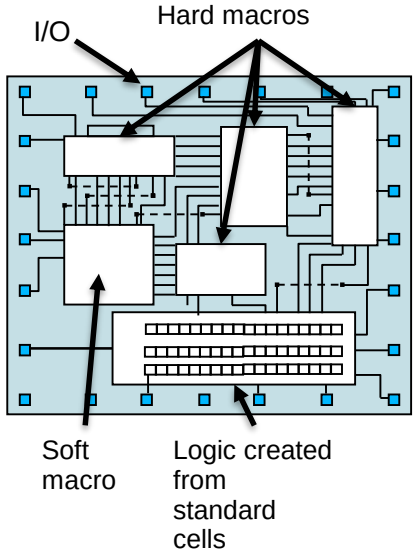


Intel 8486

Courtesy Intel

Cell-based Design (commonly used)

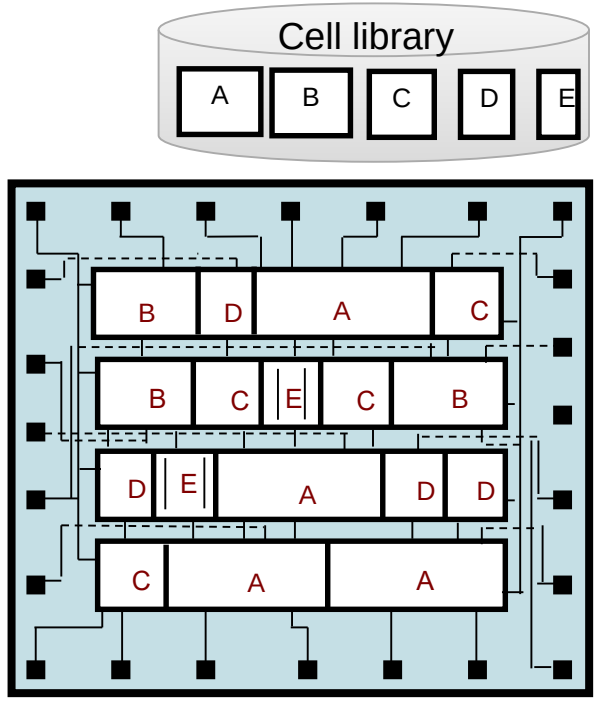
- Design is compiled using ready blocks:
  - Technology-dependent fixed size physical designs
    - Digital standard cells
      - Small cells performing simple logic used to build larger designs (Boolean primitives: and, or, xor; flip-flops; adders, etc.)
    - Hard macros (Hard IP)
      - Ready physical design (analog cells, I/O cells, etc.)
  - Technology-independent behavioural descriptions, synthezible to physical design using standard cells and if necessary hard-IPs
    - Soft macros (Soft IP),
      - Synthezible register transfer level descriptions of complex functions (processor cores, arithmetic units, etc.).
    - System-level macros (SLMs)
      - High level behavioural descriptions (DSP logic, digital filters, etc.)



source: Synopsys

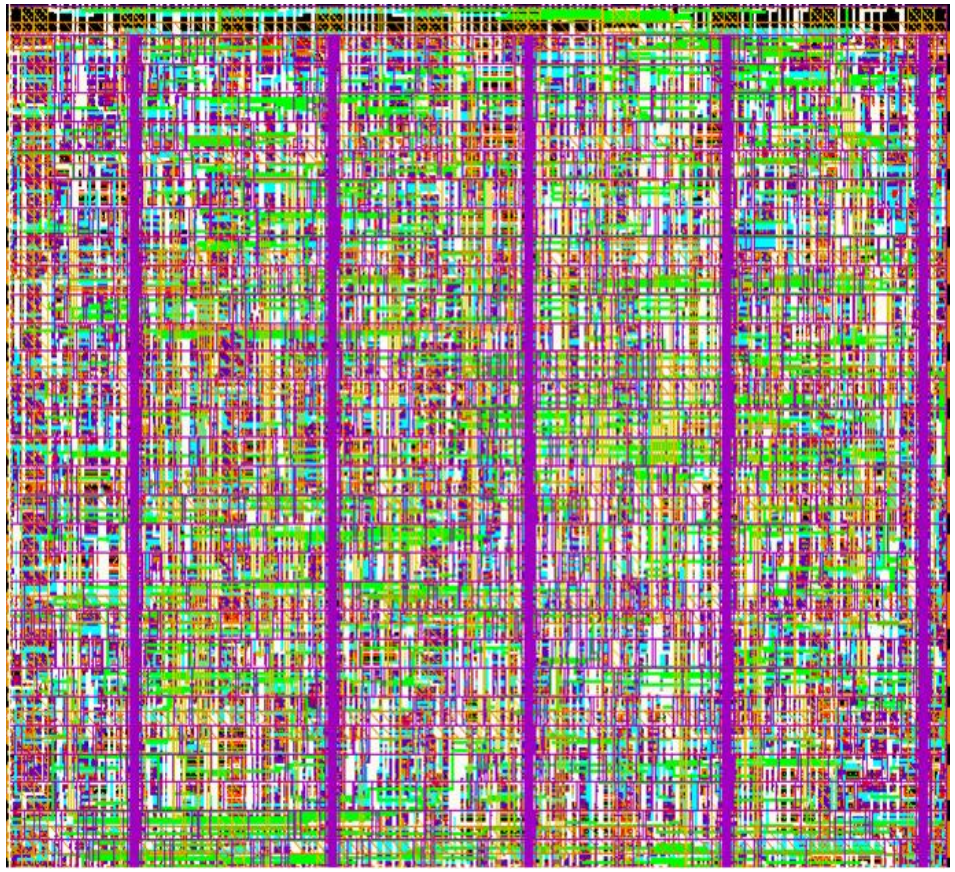
Cell-based Design: Standard Cells

- Design done at behavioral level
- Behavioral description automatically translated to logic circuit and layout by EDA tools
- Standard cells are used to build logic
- Use:
  - Large digital designs (arithmetic units, processors)



source: Synopsys

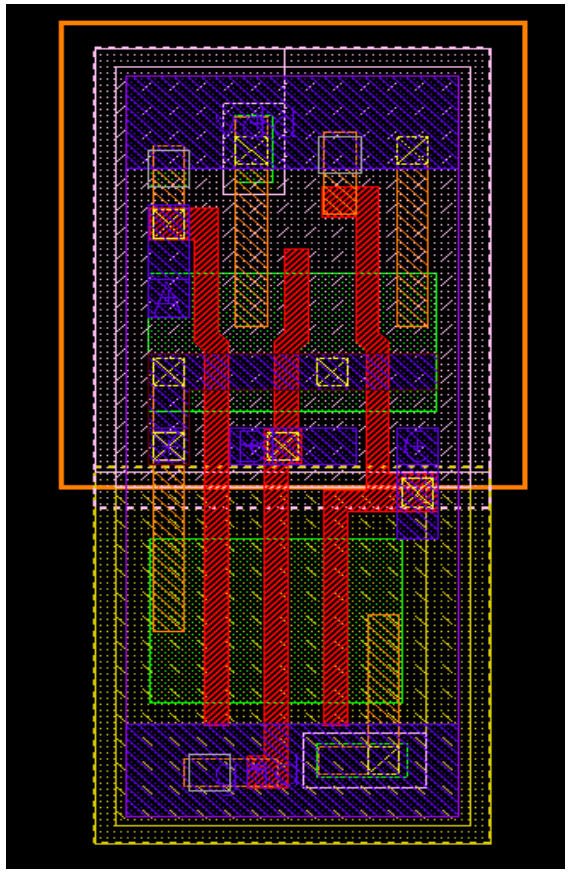
Standard Cell – The New Generation



Cell-structure
hidden under
interconnect layers

source: Synopsys

Standard Cell - Example

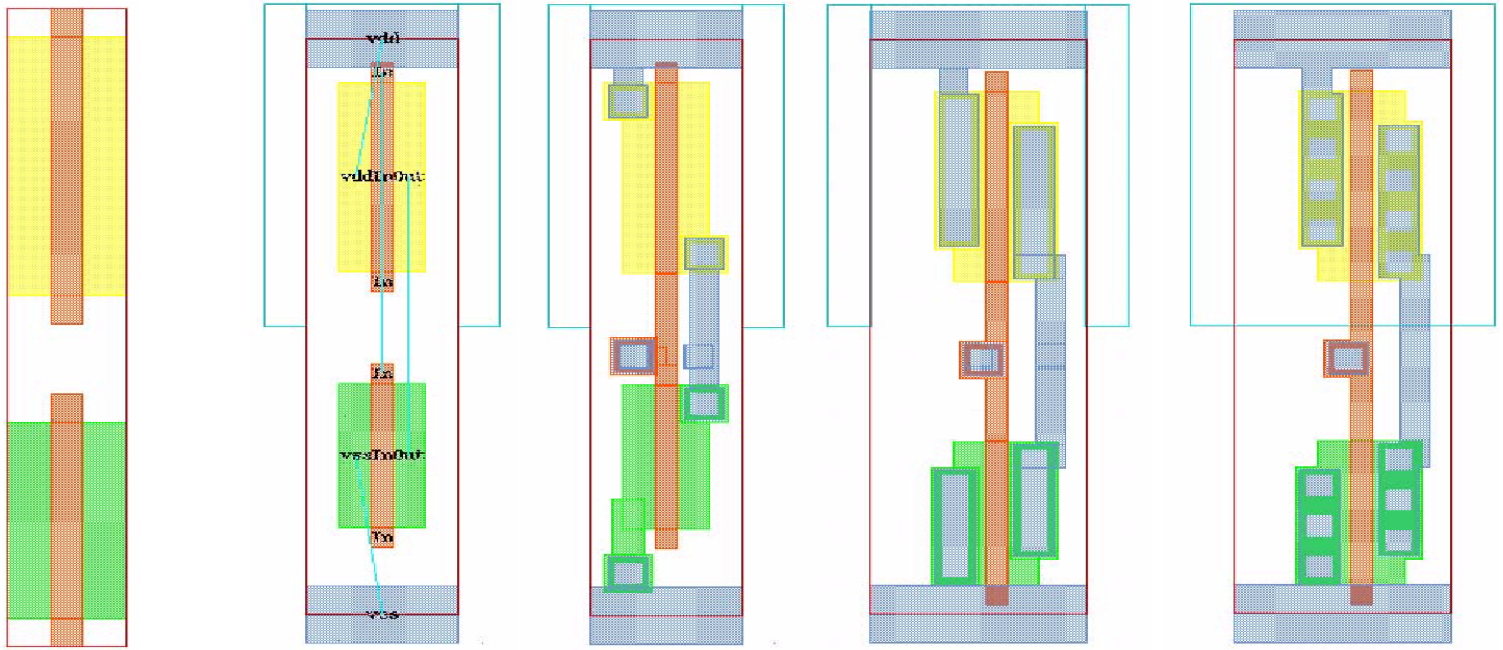


Path	1.2V - 125°C	1.6V - 40°C
$In1—t_{pLH}$	$0.073+7.98C+0.317T$	$0.020+2.73C+0.253T$
$In1—t_{pHL}$	$0.069+8.43C+0.364T$	$0.018+2.14C+0.292T$
$In2—t_{pLH}$	$0.101+7.97C+0.318T$	$0.026+2.38C+0.255T$
$In2—t_{pHL}$	$0.097+8.42C+0.325T$	$0.023+2.14C+0.269T$
$In3—t_{pLH}$	$0.120+8.00C+0.318T$	$0.031+2.37C+0.258T$
$In3—t_{pHL}$	$0.110+8.41C+0.280T$	$0.027+2.15C+0.223T$

3-input NAND cell
(from ST Microelectronics):
C = Load capacitance
T = input rise/fall time

source: Synopsys

Automatic Cell Generation



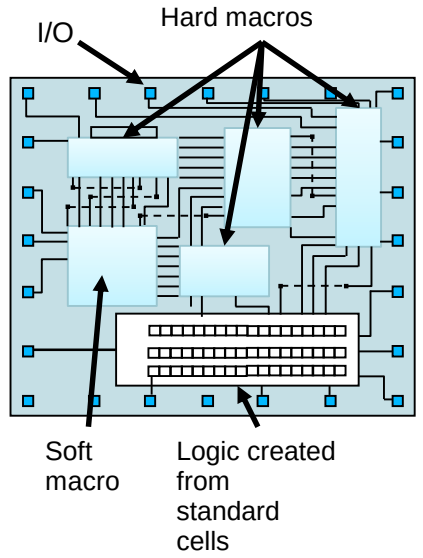
Initial transistor geometries | Placed transistors | Routed cell | Compacted cell | Finished cell

Courtesy Acadabra

source: Synopsys

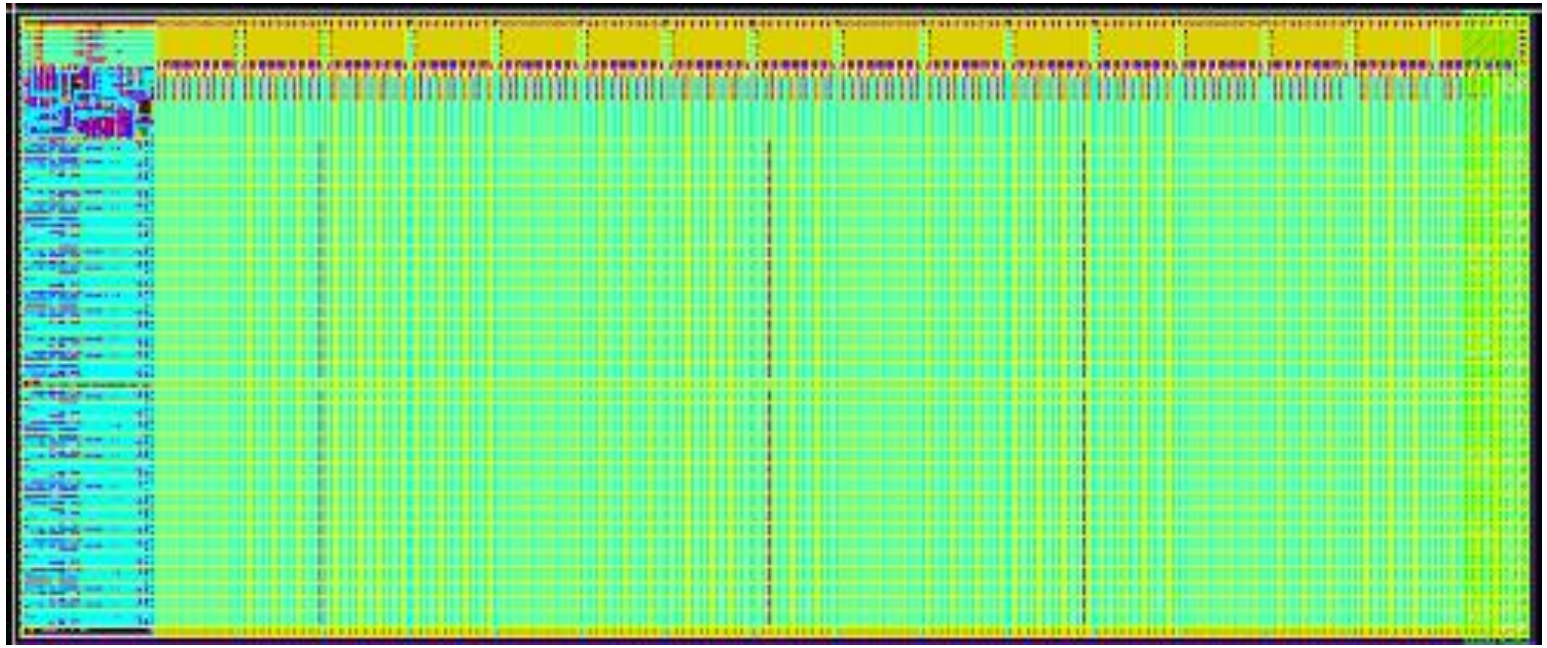
Cell-based Design: Macro Cells

- Design is created of ready macro cells
  - Hard macros like SRAMs, PLLs, Analog Macros
  - Soft macros (first synthesized for specific technology)
- Use:
  - Functionality that cannot be synthesized using only standard cells (analog functions, cache cells, I/Os, etc.)



source: Synopsys

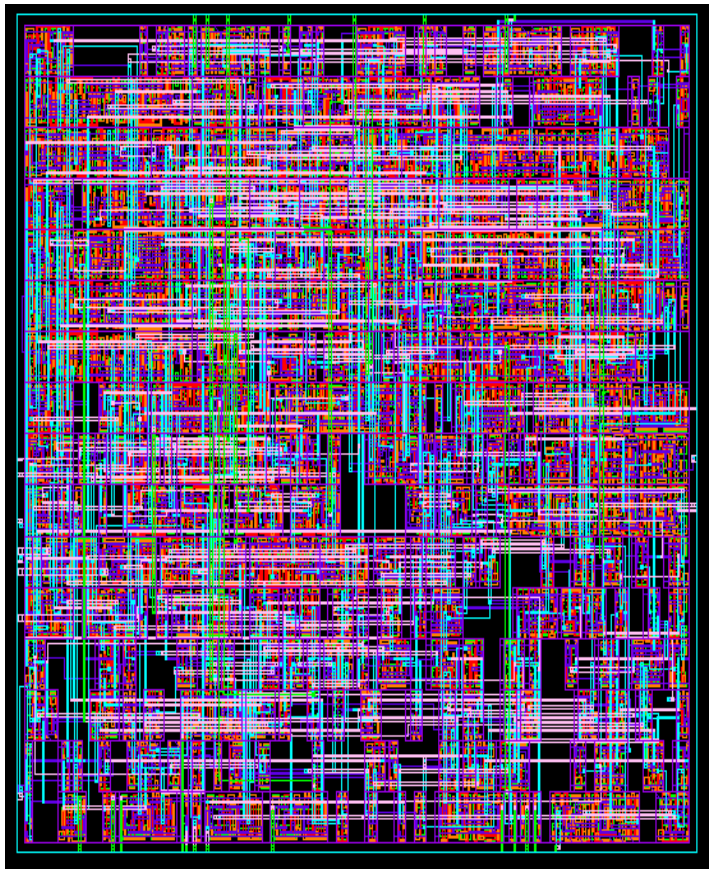
MacroModules

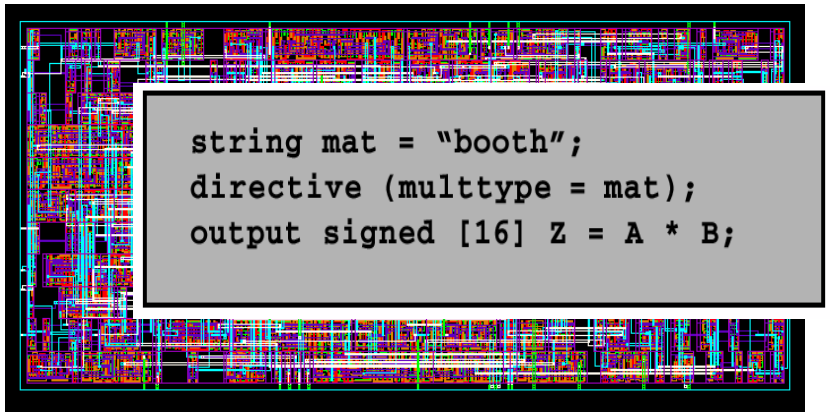


256×32 (or 8192 bit) SRAM
Generated by hard-macro module generator

source: Synopsys

“Soft” MacroModules

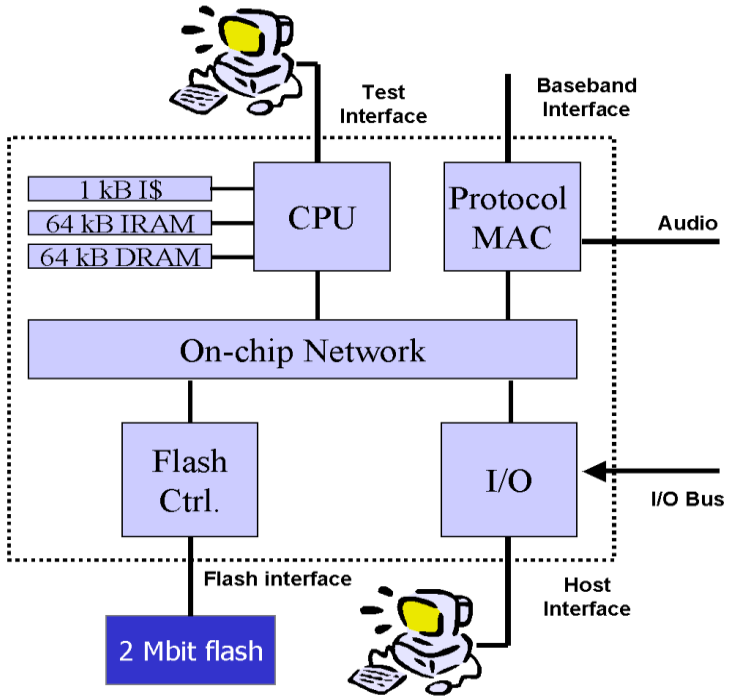


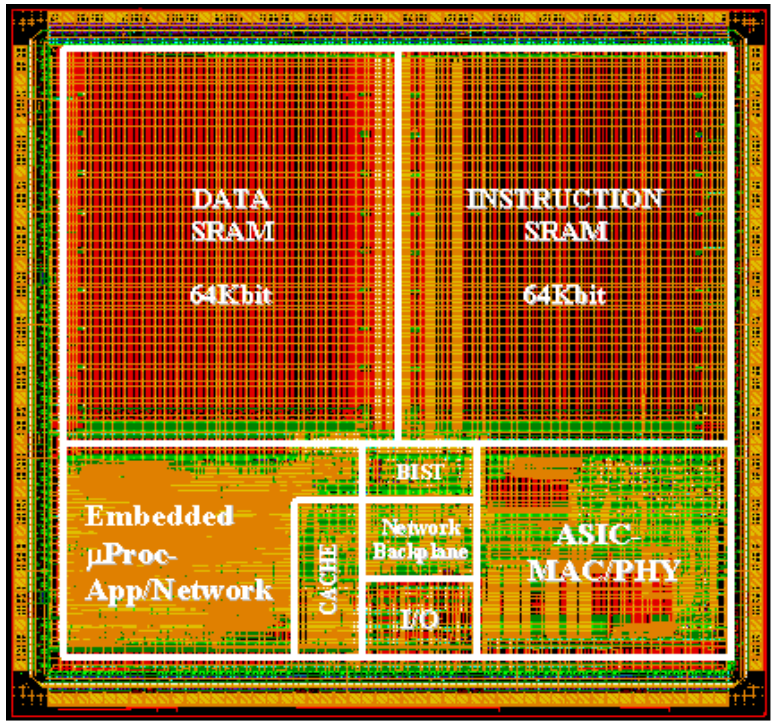


Synopsys DesignCompiler

source: Synopsys

“Intellectual Property”





A Protocol Processor for Wireless

source: Synopsys

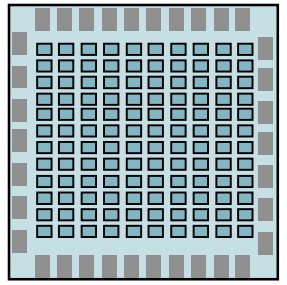
Array-based Design

- Design is created from prefabricated array of regular structures programmable by additional process steps (masks) to perform required functionality.
- Usually only interconnects are customized.
  - Sea-of-Gates
    - Gates fill all core area
    - Best cell density, hard interconnection
  - Channeled Gate Array
    - Special routing channels are left between gates
    - Not best cell density, easy interconnection
  - Structured Gate Array
    - Custom blocks are embedded (memory, PLL, A/D converter)

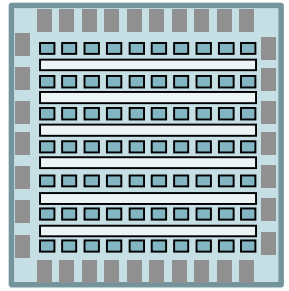
Gate Array and Sea-of-Gates

- Common advantages
  - Small number of masks used
  - Wide range of package options
  - Low NRE costs
  - Fast time to market
  - Relatively low minimum order quantity
- Use:
  - communication, industry and office automation products.

- Gate Array
  - Only top metal layers used
  - Low gate denisty

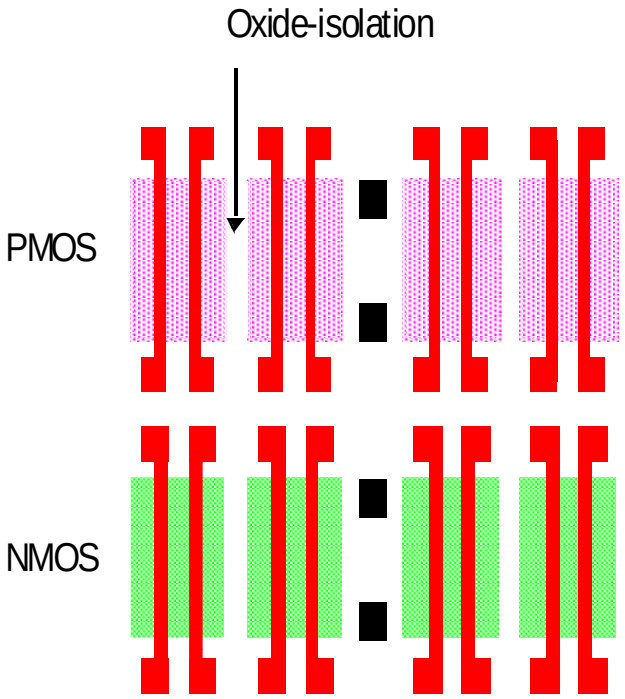


- Sea-of Gates
  - More metal layers required (routing done over gates)
  - High gate density

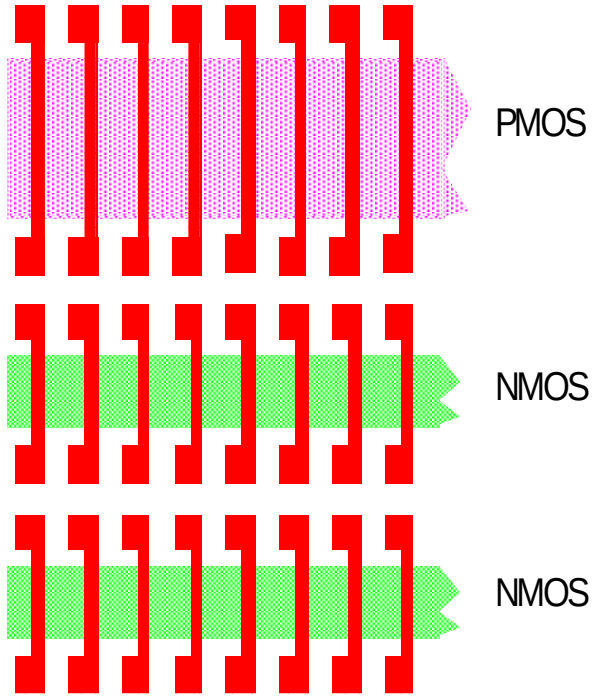


source: Synopsys

Sea-of-gate Primitive Cells



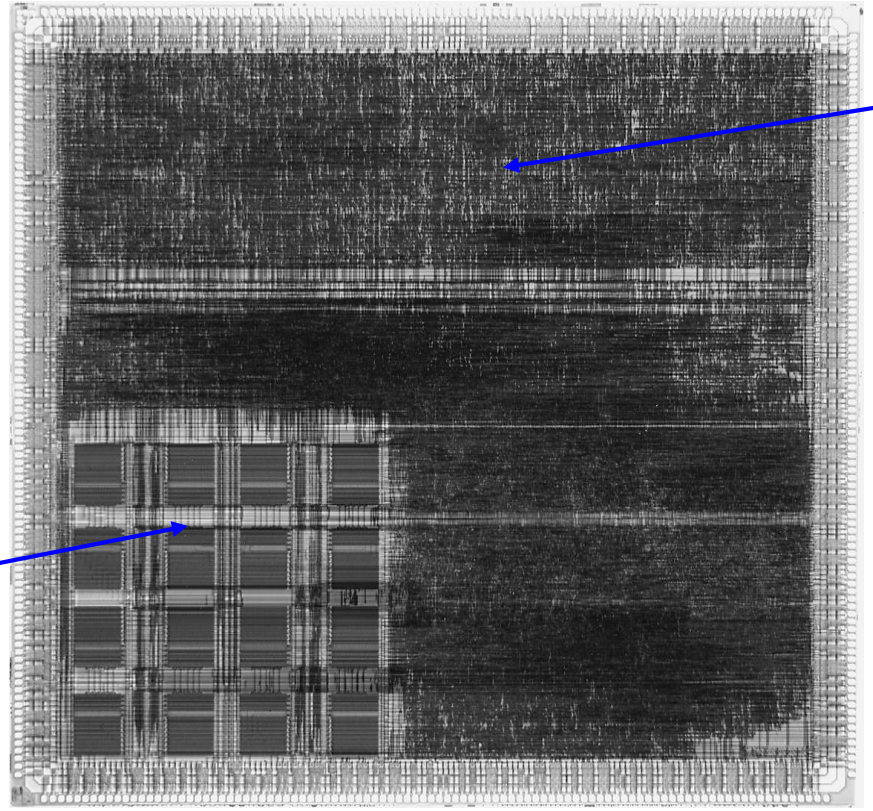
Using oxide-isolation



Using gate-isolation

source: Synopsys

Sea-of-gates



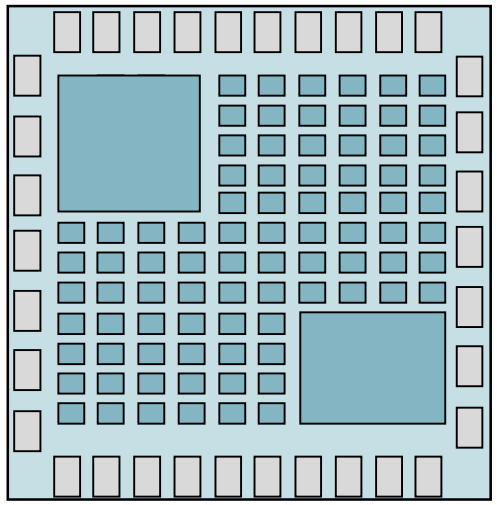
LSI Logic LEA300K
(0.6 μm CMOS)

Courtesy LSI Logic

source: Synopsys

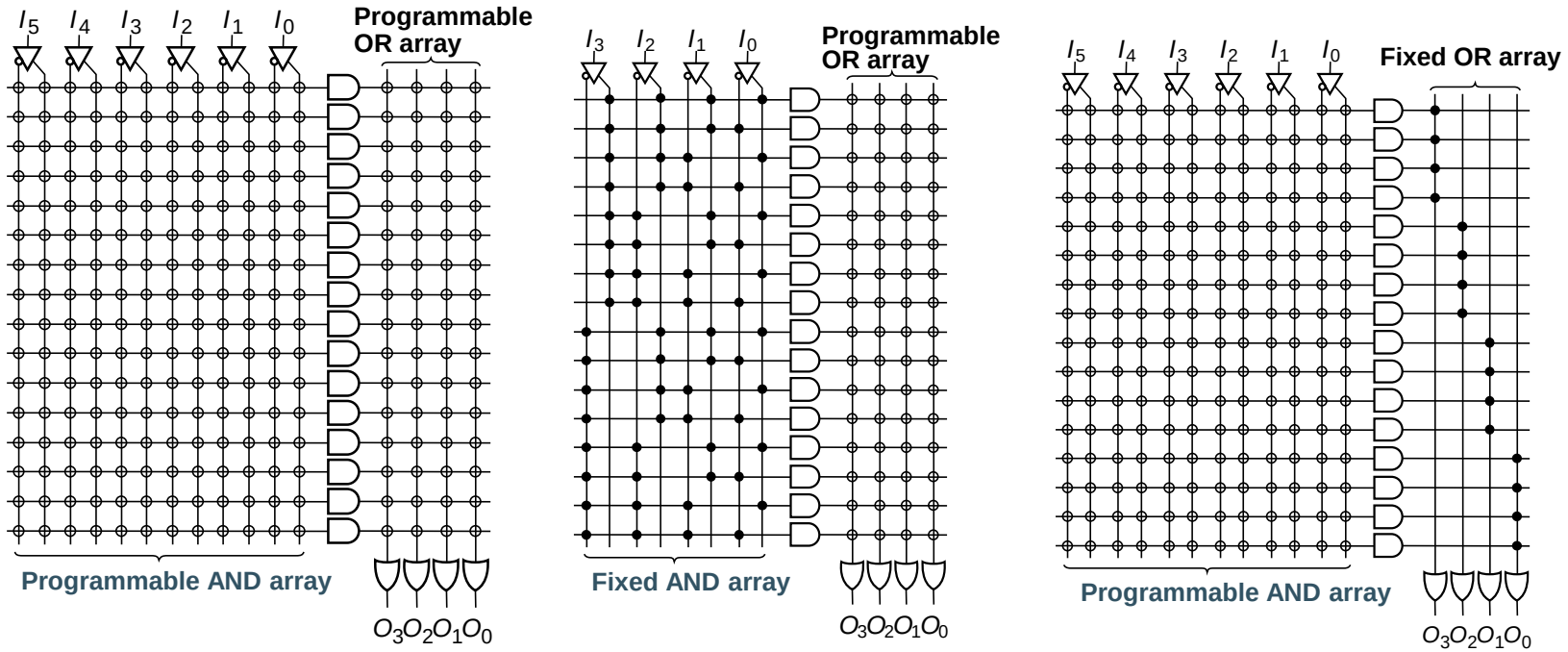
Structured Gate Array

- Gate Arrays are not suitable for some circuits
  - RAMs, PLL, etc.
- Structured Gate Array can contain
  - Embedded memories
  - Various functional blocks
- Disadvantages
  - Custom blocks are fixed
  - Unneeded functions waste area



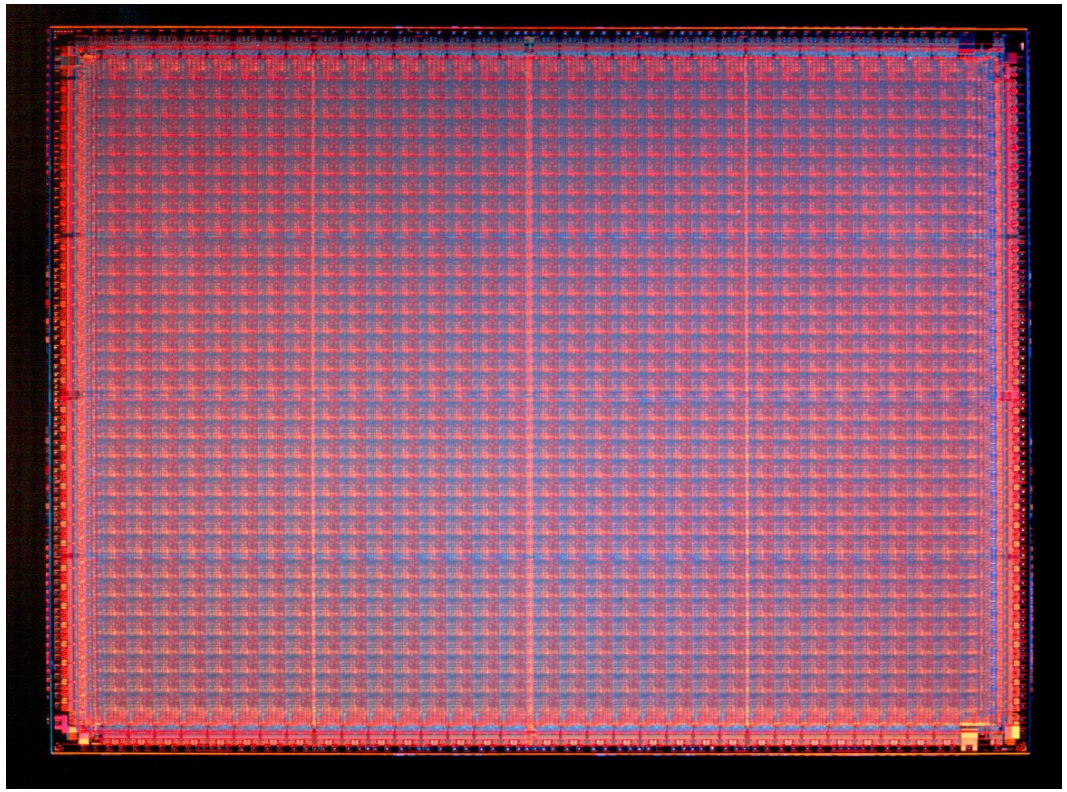
source: Synopsys

Array-Based Programmable Logic



source: Synopsys

RAM-based FPGA

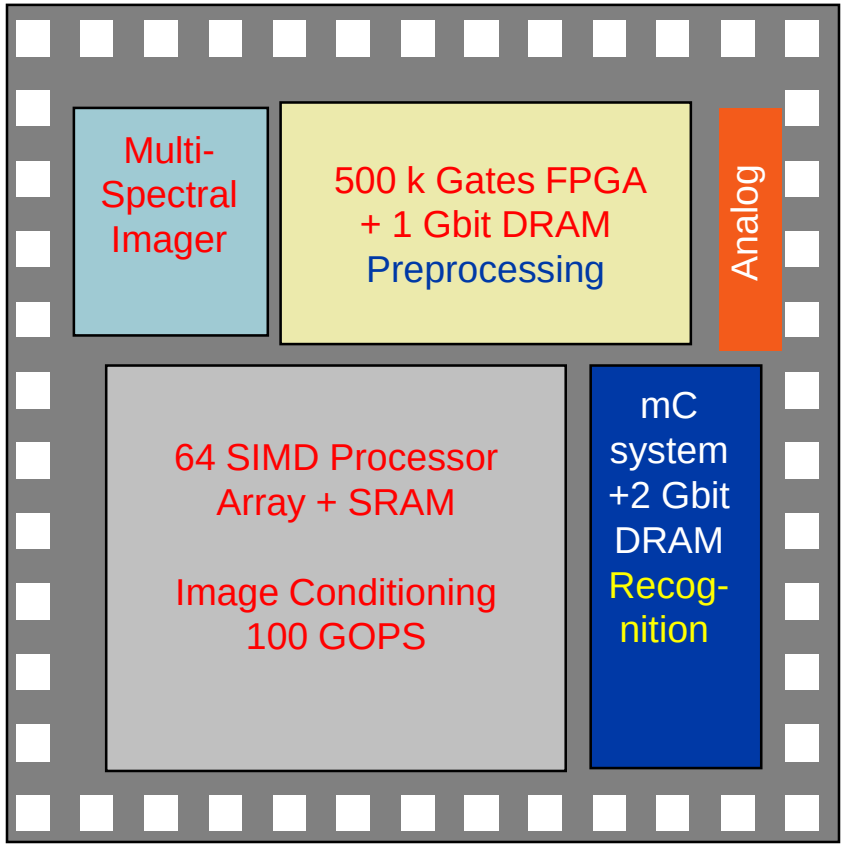


Courtesy Xilinx

Xilinx XC4000ex

source: Synopsys

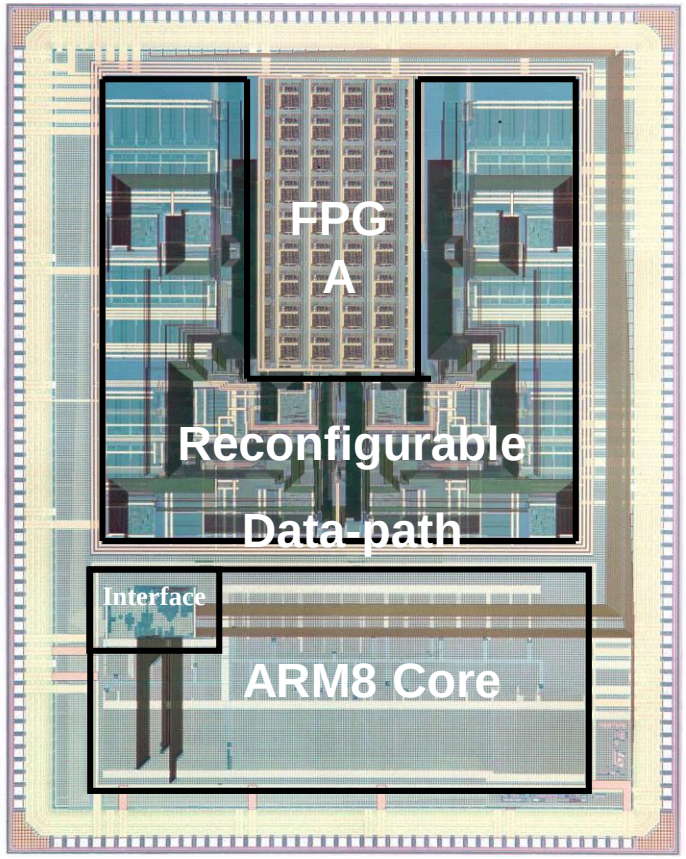
Design at a crossroad: System-on-a-Chip



- Embedded applications where cost, performance, and energy are the real issues!
- DSP and control intensive
- Mixed-mode
- Combines programmable and application-specific modules
- Software plays crucial role

source: Synopsys

Berkeley Pleiades Processor

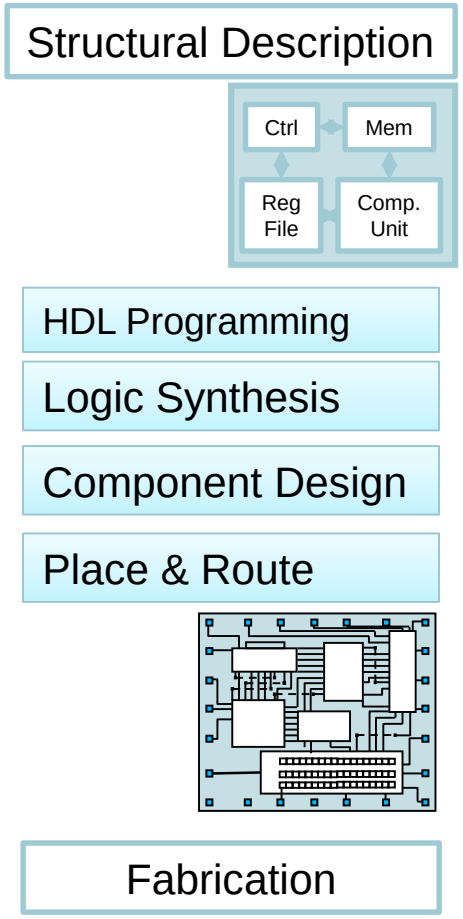


- 0.25um 6-level metal CMOS
- 5.2mm x 6.7mm
- 1.2 Million transistors
- 40 MHz at 1V
- 2 extra supplies: 0.4V, 1.5V
- 1.5~2 mW power dissipation

source: Synopsys

Full Custom Design Cycle

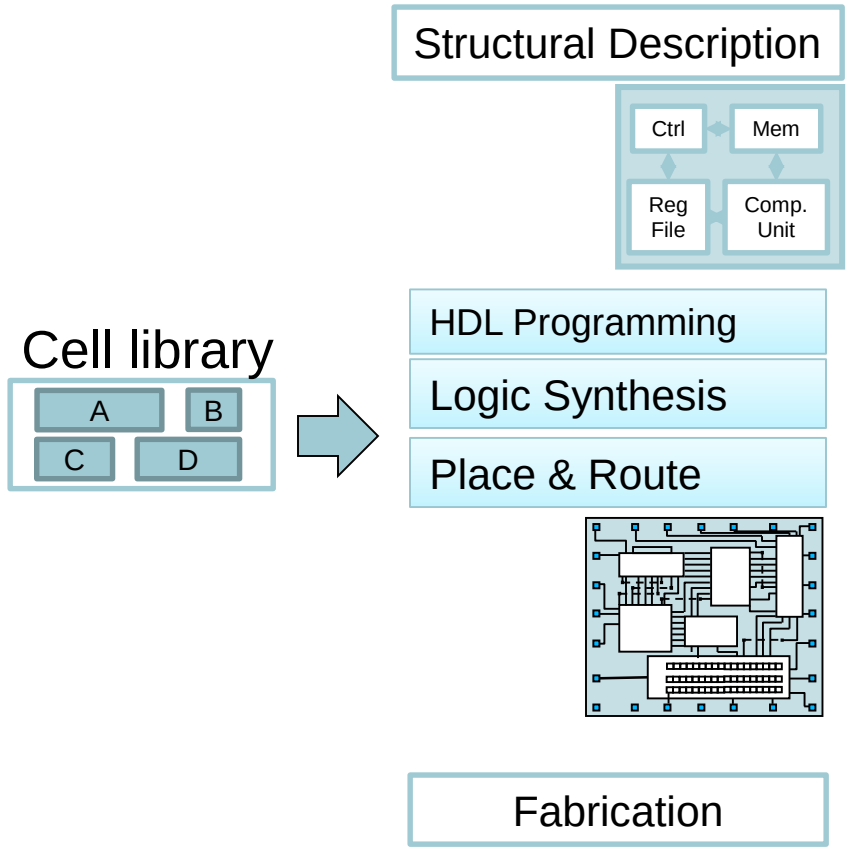
- Structural description
  - Design of required circuit at architectural level
- HDL Programming
  - Coding required behavior in HDL
- Component Design
  - Custom design of non-synthisable components and standard cells
- Logic Synthesis
  - Synthesis of logic circuit form HDL description
- Place & Route
  - Physical design



source: Synopsys

Semi Custom: Cell Based

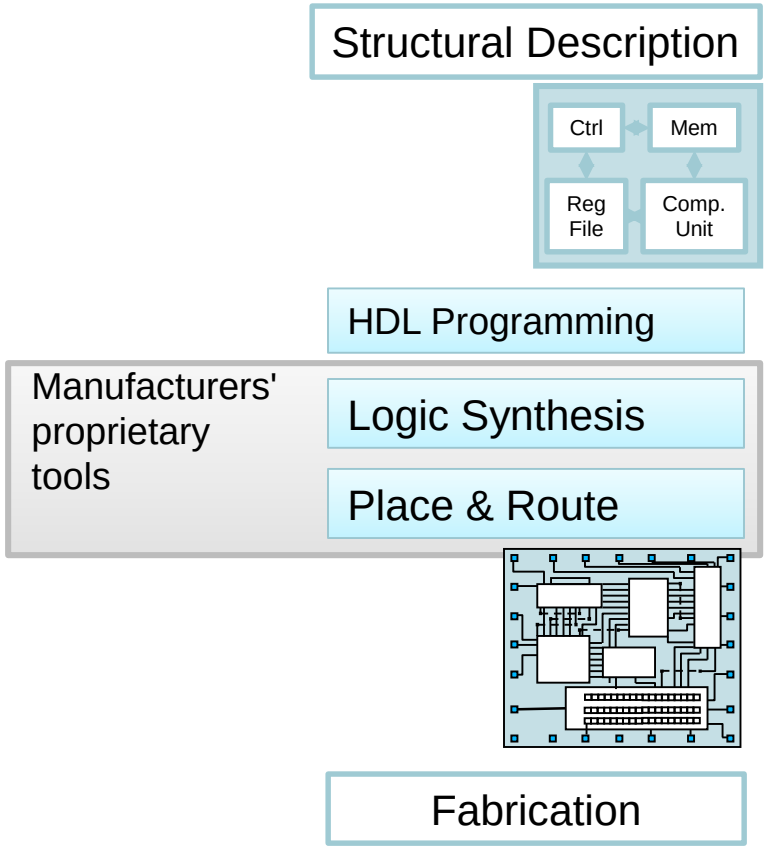
- Standard cells and another components are predesigned



source: Synopsys

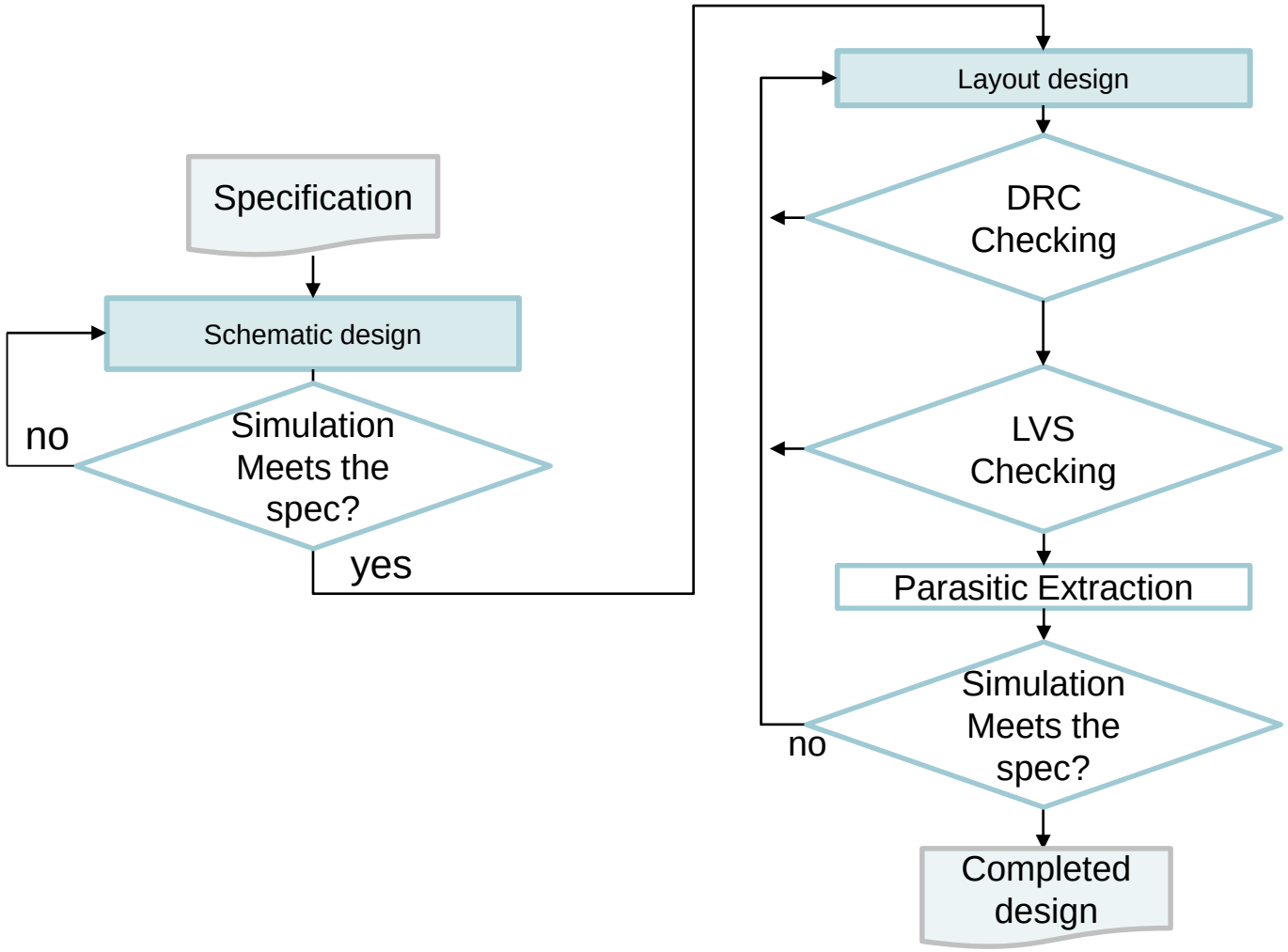
Semi Custom: Array Based

- Some structures are prefabricated
- Manufacturer provides proprietary synthesis and place & route tool for physical design



source: Synopsys

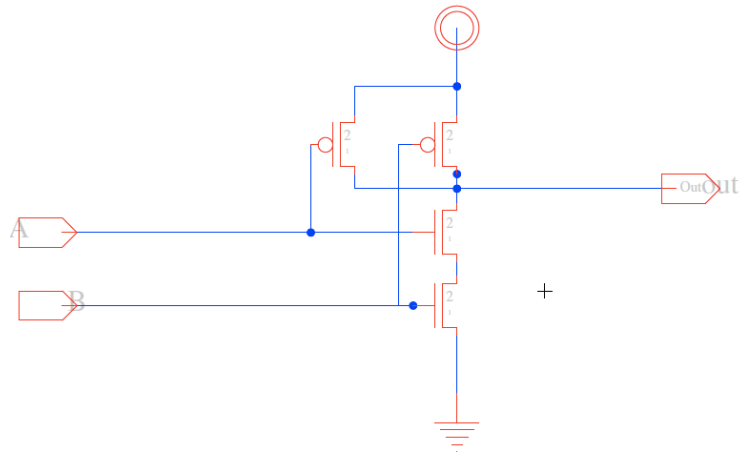
Custom Design Flow

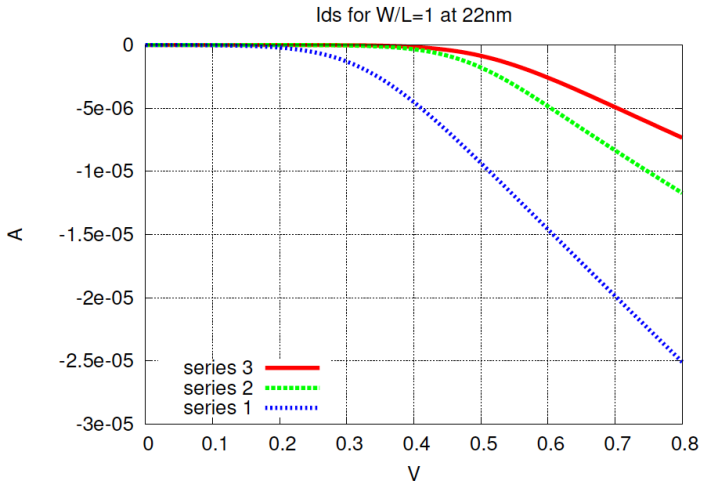


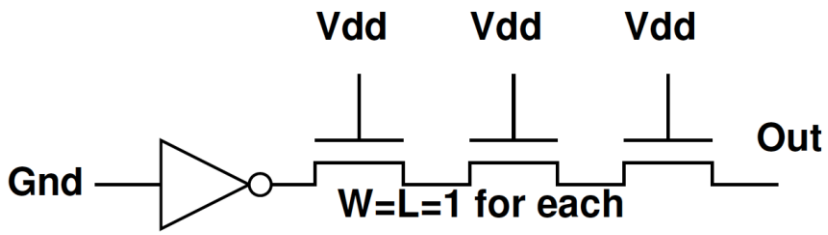
source: Synopsys

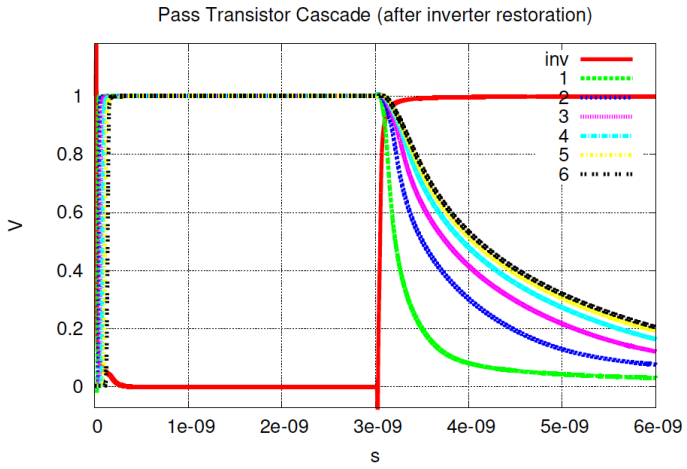
SPICE Circuit Simulator

Usually used to simulate custom designs.



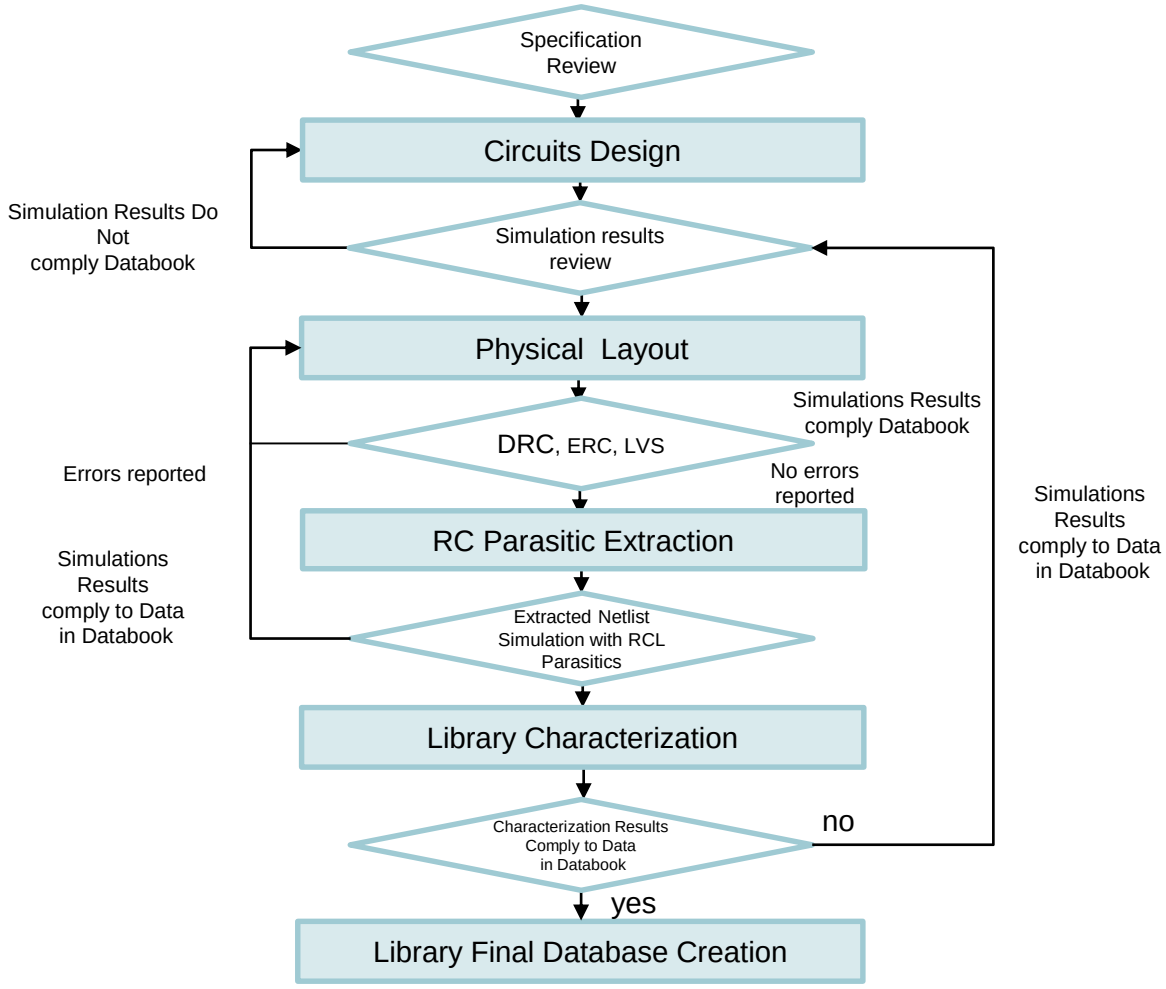






source: Upenn ese532/fall2021

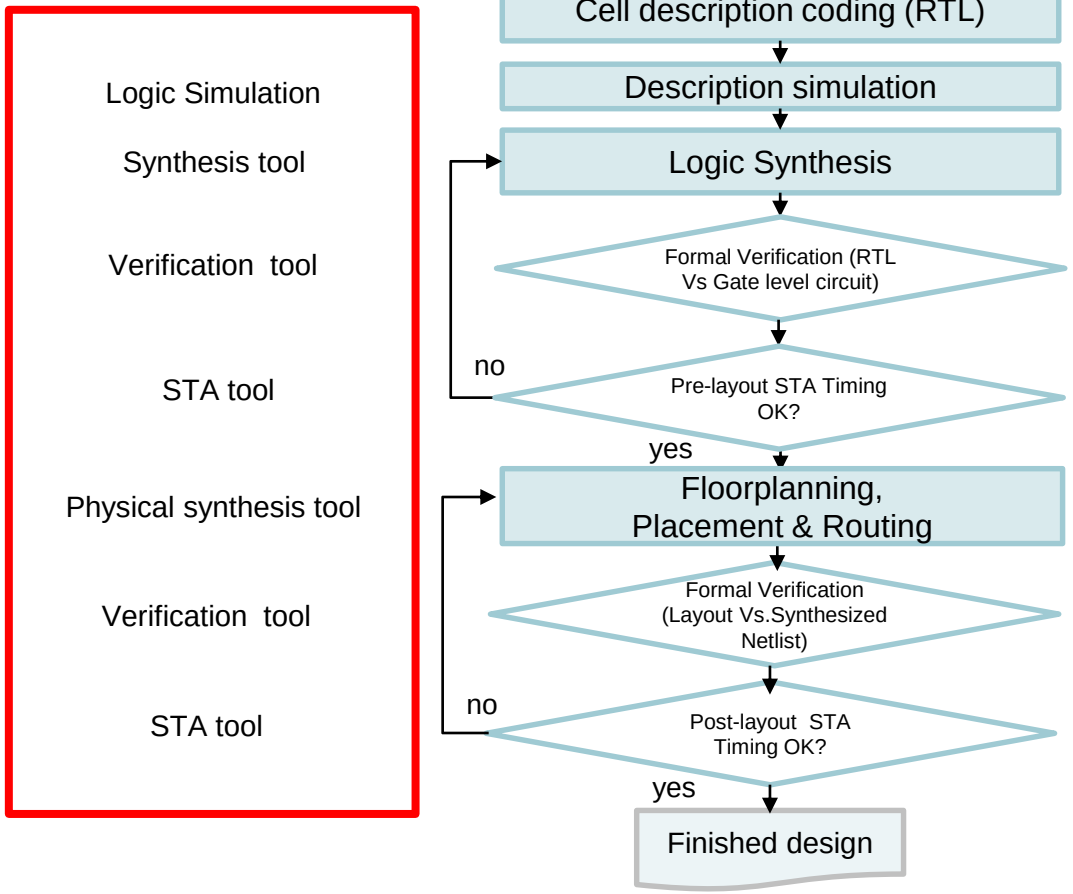
Digital Standard Cell Library Design Flow



source: Synopsys

Digital IC Design Flow

Need tools! Electronic Design Automation (EDA) Tools



source: Synopsys

EDA

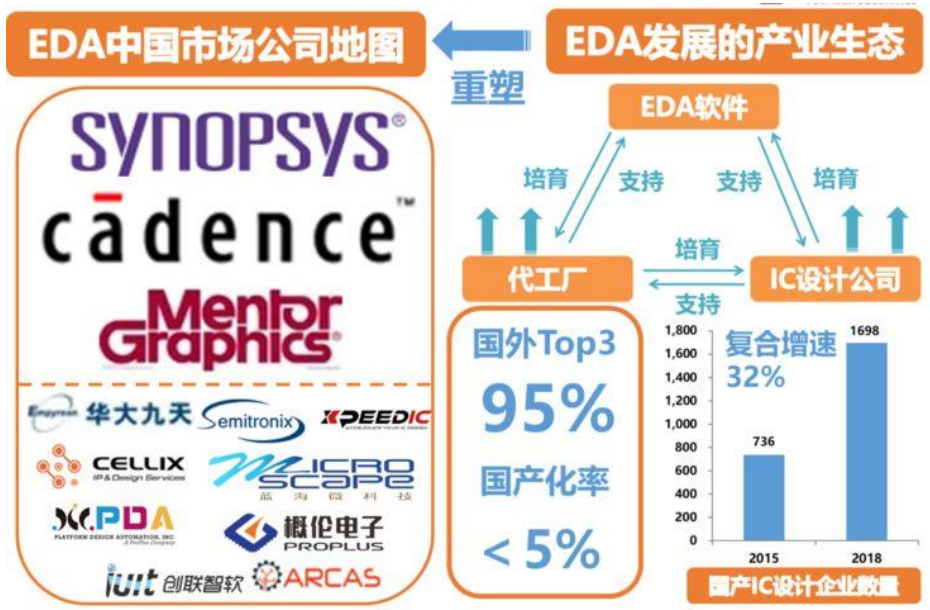
- The enabler of billion-transistor chips
- A lot of algorithms behind these tools
- EDA evolves by a series of paradigm shifts
  - From Computer Aided Design (CAD) to EDA

XILINX cadence

SYNOPSYS® KEYSIGHT TECHNOLOGIES

ANSYS® Mentor Graphics®

Part of Siemens



A list of Chinese EDA vendors

EDA软件开发领域的中国公司一览（不完全统计）

公司名称	所在地	成立时间（年）	注册资本（万）	研发方向
华大九天	北京	2009	17217	数模设计验证、后端时序分析等
博达微	北京	2012	293	器件建模、测试
芯华章科技	南京	2020	8446	数字验证：硬件仿真器，FPGA原型验证，智能验证，形式验证，逻辑仿真
国微思尔芯	上海	2004	1687	原型验证、云原型验证
奥卡思	成都	2018	1200	形式验证设计
广立微	杭州	2003	1000	良率测试
行芯	杭州	2018	600	IP低功耗
珂晶达	苏州	2011	1000	器件工艺仿真
芯禾科技	苏州	2010	1992	仿真、IPD
鸿芯微纳	深圳	2018	110000	数字布局布线、时序分析、功耗分析
国微集团	深圳	2002	22000	系统硬件仿真、逻辑综合、DFT等
贝思科尔	深圳	2011	8000	模拟混合验证
九同方微	武汉	2011	693	器件建模仿真
芯愿景	北京	2002	5000	集成电路分析设计平台、IP核和EDA软件
概伦电子	济南	2010	4250	仿真、噪声、测试
蓝海微	天津	2009	100	定制化服务

Source：公开报道及天眼查等企业数据库，由钛媒体App编辑统计制表。

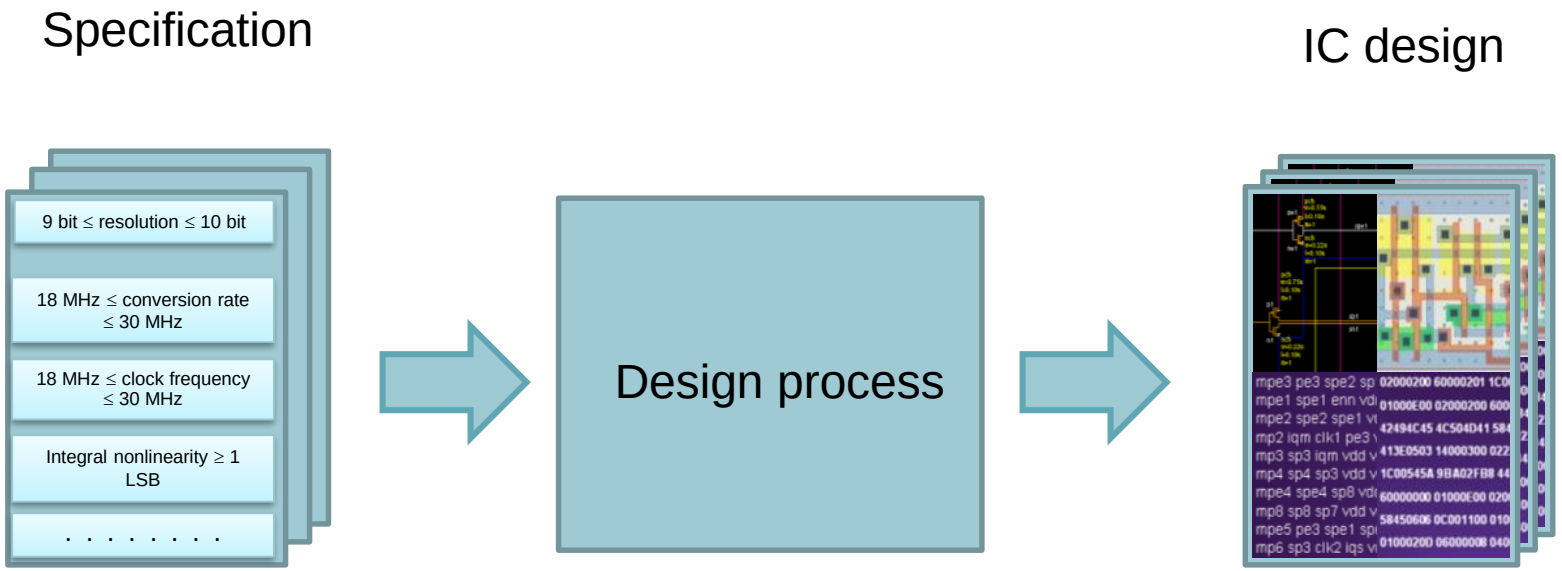
TMTPOST 钛媒体

EDA Tools

- Implementation Tools (Synthesis, Design for Test, Place & Route)
- Verification Tools (Functional Simulation, Formal Verification, etc.)
- Signoff Tools (DRC/LVS check, Power Integrity, Signal Integrity, Static Timing Analysis, etc.)

Electronic Design Process

- Design process is the process of getting IC design from specification.



Why is it so hard?

source: Synopsys

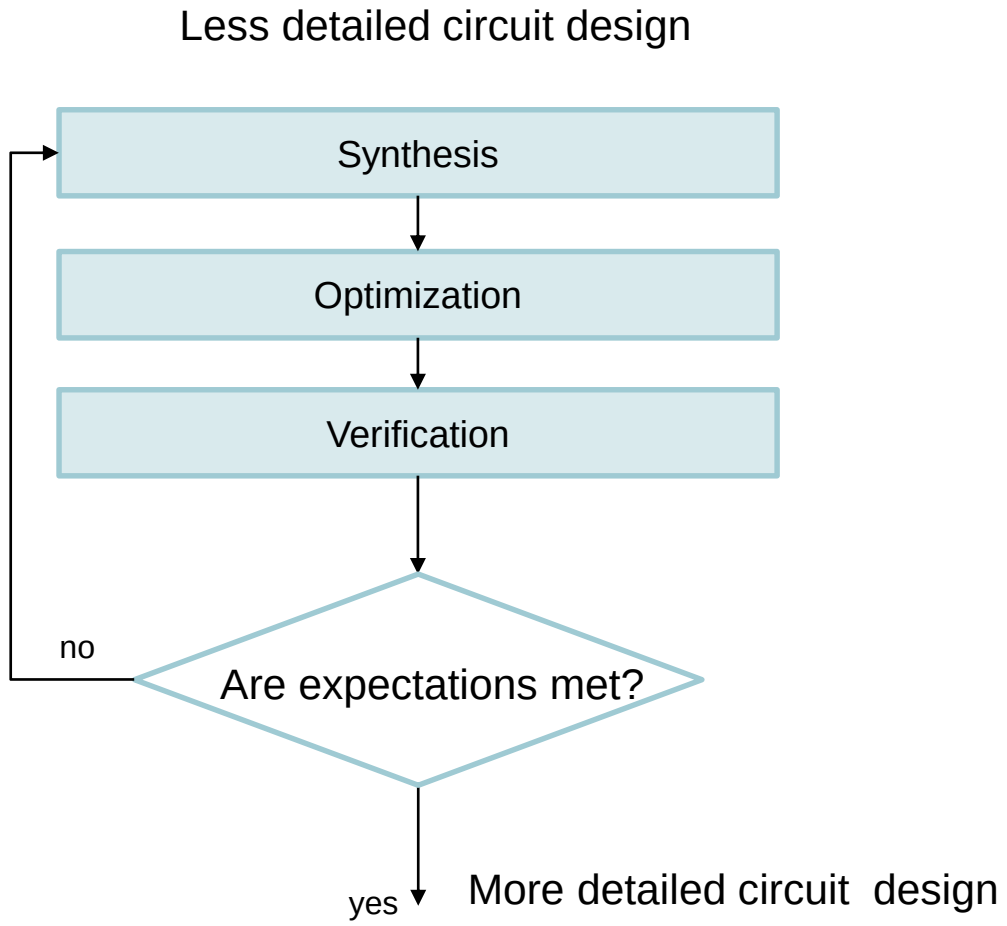
Isn’t it just “click button”?

- Design a chip in one day?
  - If you ask a student, it might be a “Yes”, sure, just click the button…I can synthesize it on FPGA in few minutes…
  - If you ask a physical design engineer, it is like “are you kidding”, why?
    - High quality chip that meets all requirements requires a lot of efforts
    - EDA tools are not mobile Apps
    - EDA tools are only “tools”, they can’t make decisions
    - Even the RTL is ready, but the way you implement it into silicon can lead to huge differences in PPA
    - …

VLSI Design Problem

- IC Characteristics
  - Area
  - Speed
  - Power dissipation
  - Design time
  - Testability
  - Power integrity
  - …
- It is impossible to design a VLSI circuit at once while having required characteristics
- The complexity is simply too high

Problems Solved in One Design Level



source: Synopsys

The “Design Closure” Problem

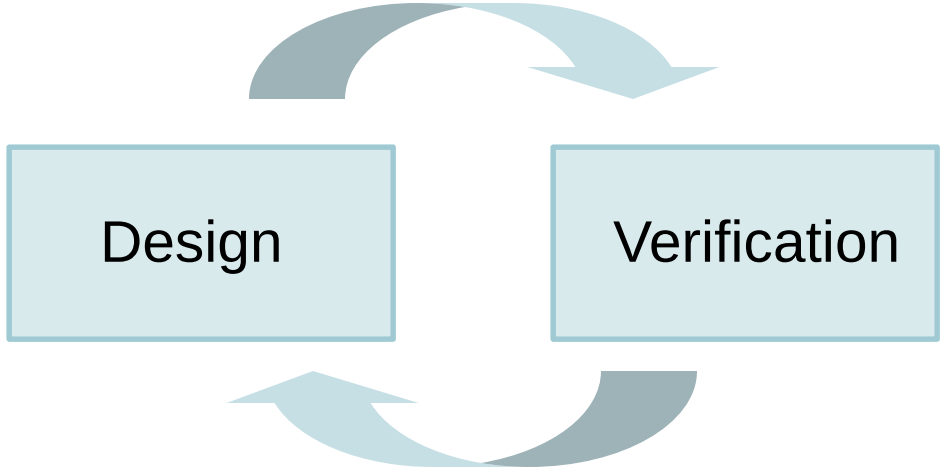
Iterative Removal of Timing Violations (white lines)

Courtesy Synopsys

source: Synopsys

Design Process

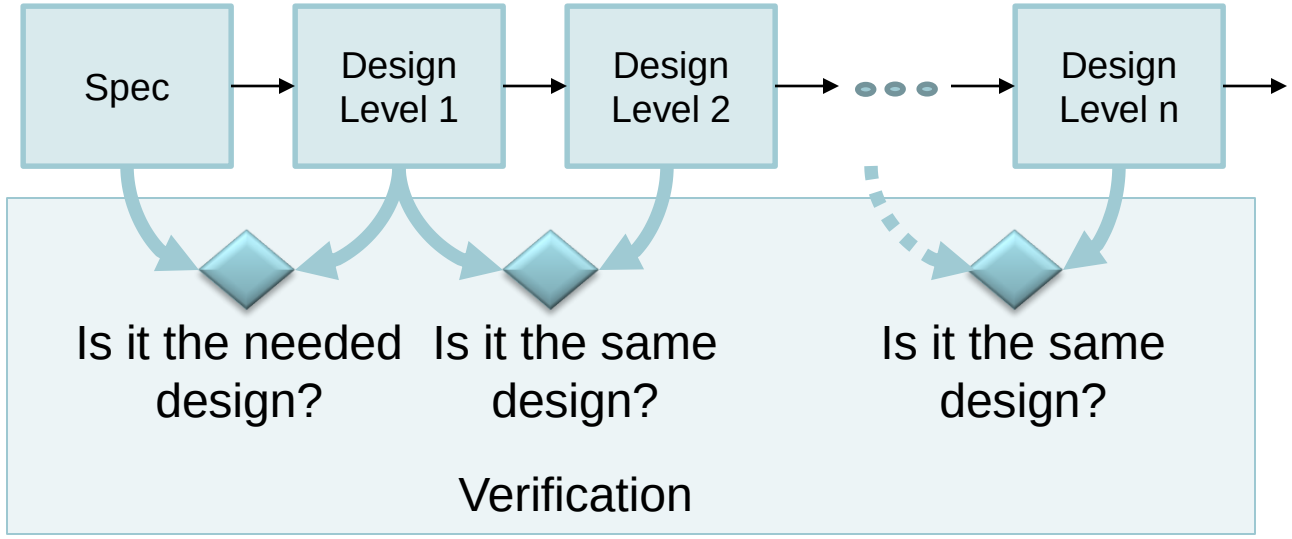
- Design process cycles between synthesis and verification



source: Synopsys

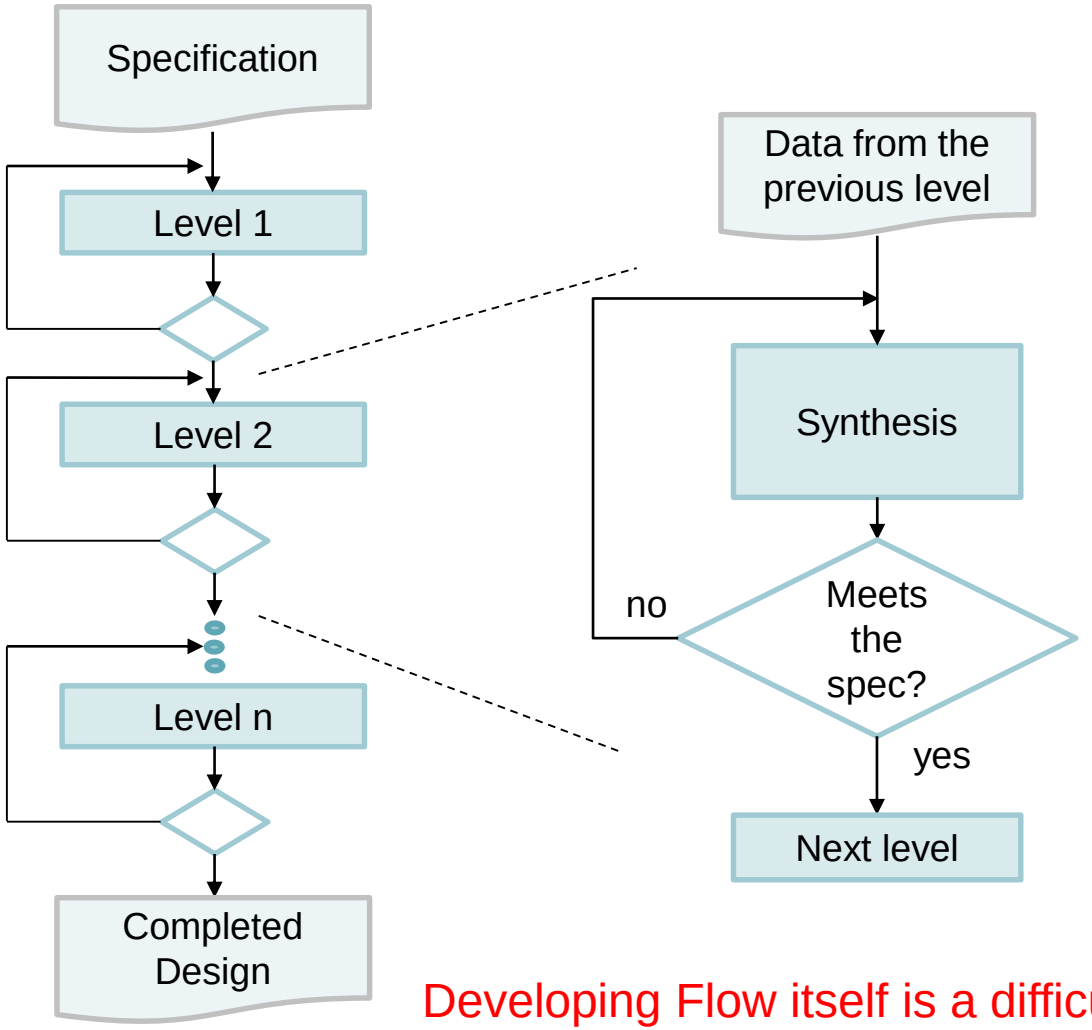
Verification

- Verification is used to check if the design object produced by the design step is the same as the needed one



source: Synopsys

The Concept of Design Flow



A SoC Design Team in a Large Company

- Architects
- RTL Design Team (deliver RTL)
- CAD Team (for flow development)
- Physical Design Team (for running the flow and close the design with given spec)
- Verification Team (verifying the functionality of RTLs)
- IP Team (for any custom design)
  - Standard Cell team
  - Analog team
  - …
- IT Support team

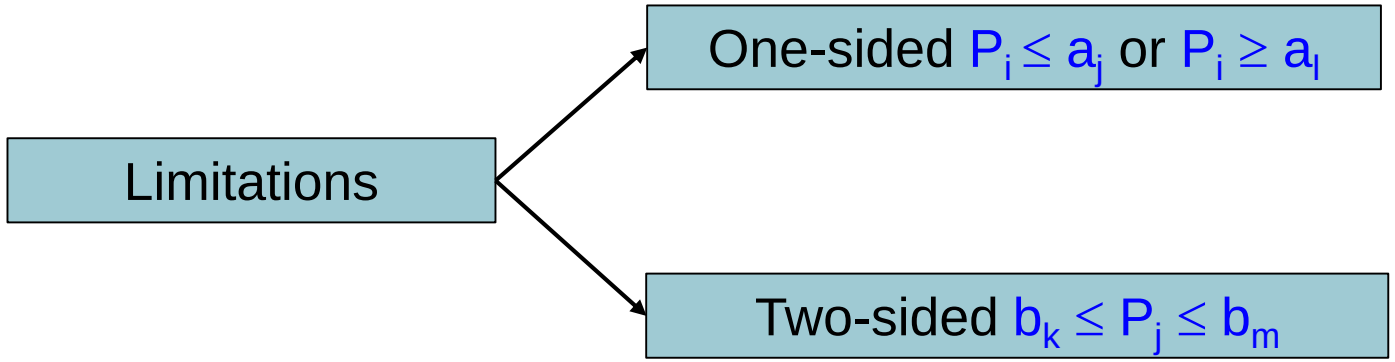
Developing Flow itself is a difficult task.

source: Synopsys

LOGIC SYNTHESIZE

From RTL to Gate-level Netlist

IC Design starts from Specification



P_i is i^{th} parameter of IC

a_j is boundary value of i^{th} parameter of IC

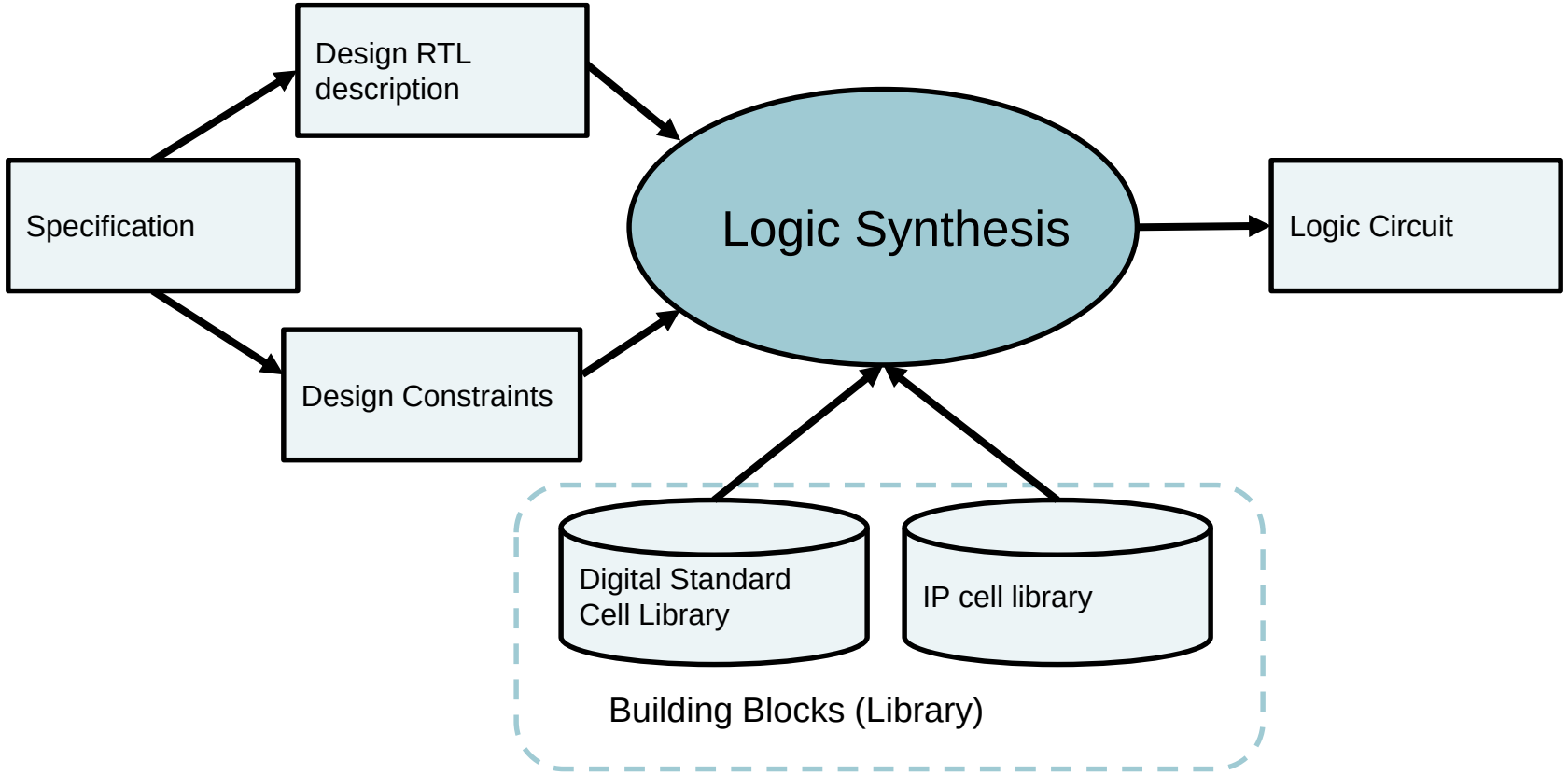
Specification Example

N0	Parameter description	Min	Typ	Max	Units
1.	Process		3.3V IO devices in TSMC 0.11		
2.	Resolution	9		10	Bits
3.	Conversion Rate	18		30	MHz
4.	Input Clock Frequency	18		30	MHz
5.	Integral Nonlinearity			1	LSB
6.	Differential Nonlinearity			0.5	LSB
7.	Gain Error		5		%FSR
8.	Offset error		5		%FSR
9.	Signal to Noise Ratio	56		62	DBc
10.	Harmonic Distortion		-60		DBc

Specification Example (2)

N0	Parameter description	Min	Typ	Max	Units
11.	Temperature Drift			12	ppm/C
12.	Reference Voltage		1.25		V
13.	Analog Input Voltage		1.6		V
14.	Power Supply Voltage1	1.08	1.2	1.32	V
15.	Power Supply Voltage2	3	3.3	3.6	V
16.	Power Dissipation		125	180	mW
17.	Operating Temperature	0		125	° C
18.	Spurious Free Dynamic Range			-10	dB
19.	Effective Resolution Band Width		6		MHz
20.	Clock jitter			28	Ps

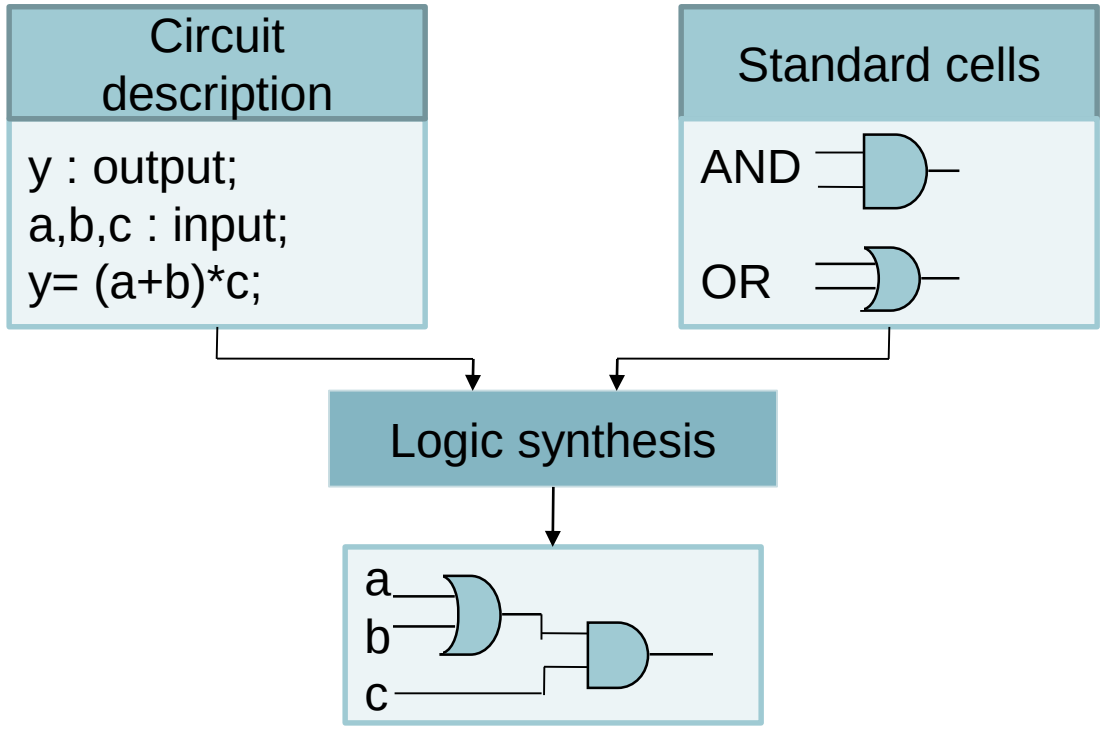
Logic Synthesis



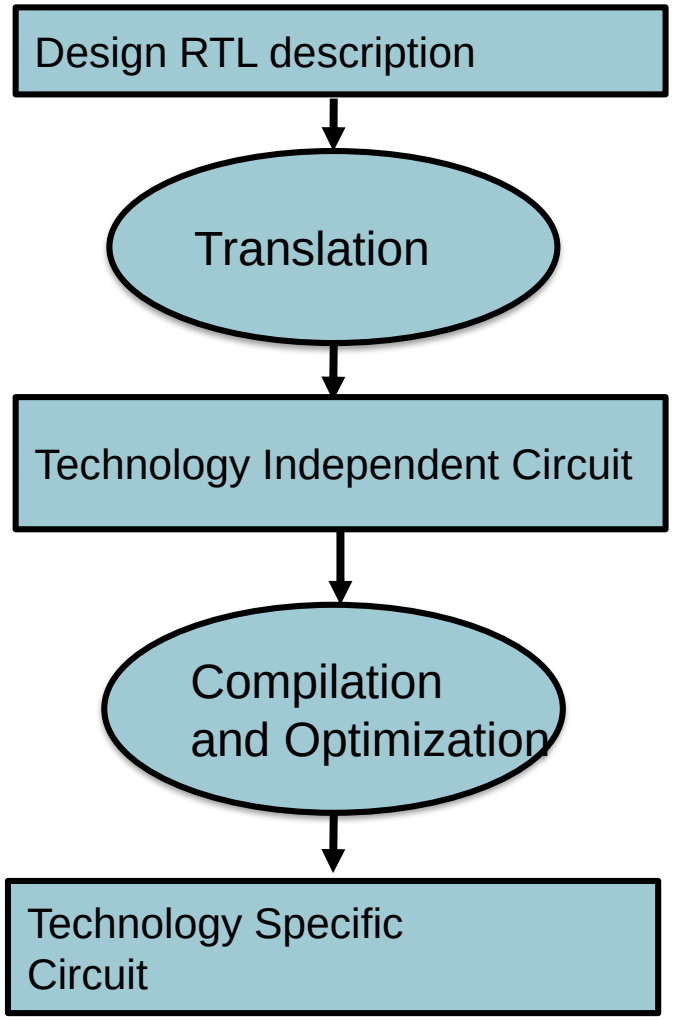
source: Synopsys

Logic Synthesis

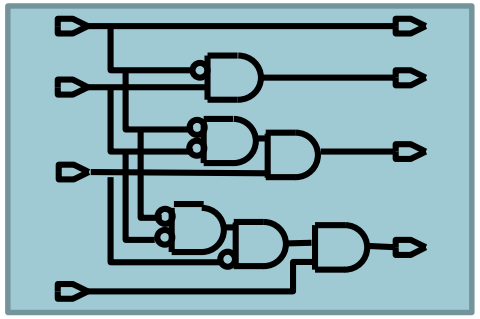
Logic synthesis is the process which produces logic circuit from circuit description



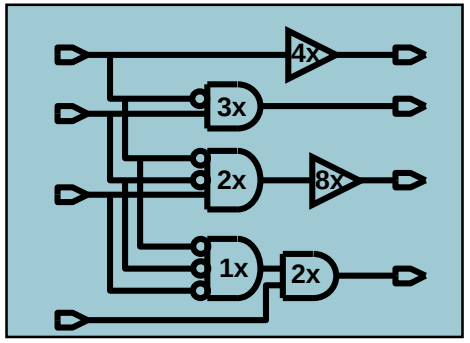
Logic Synthesis Steps




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



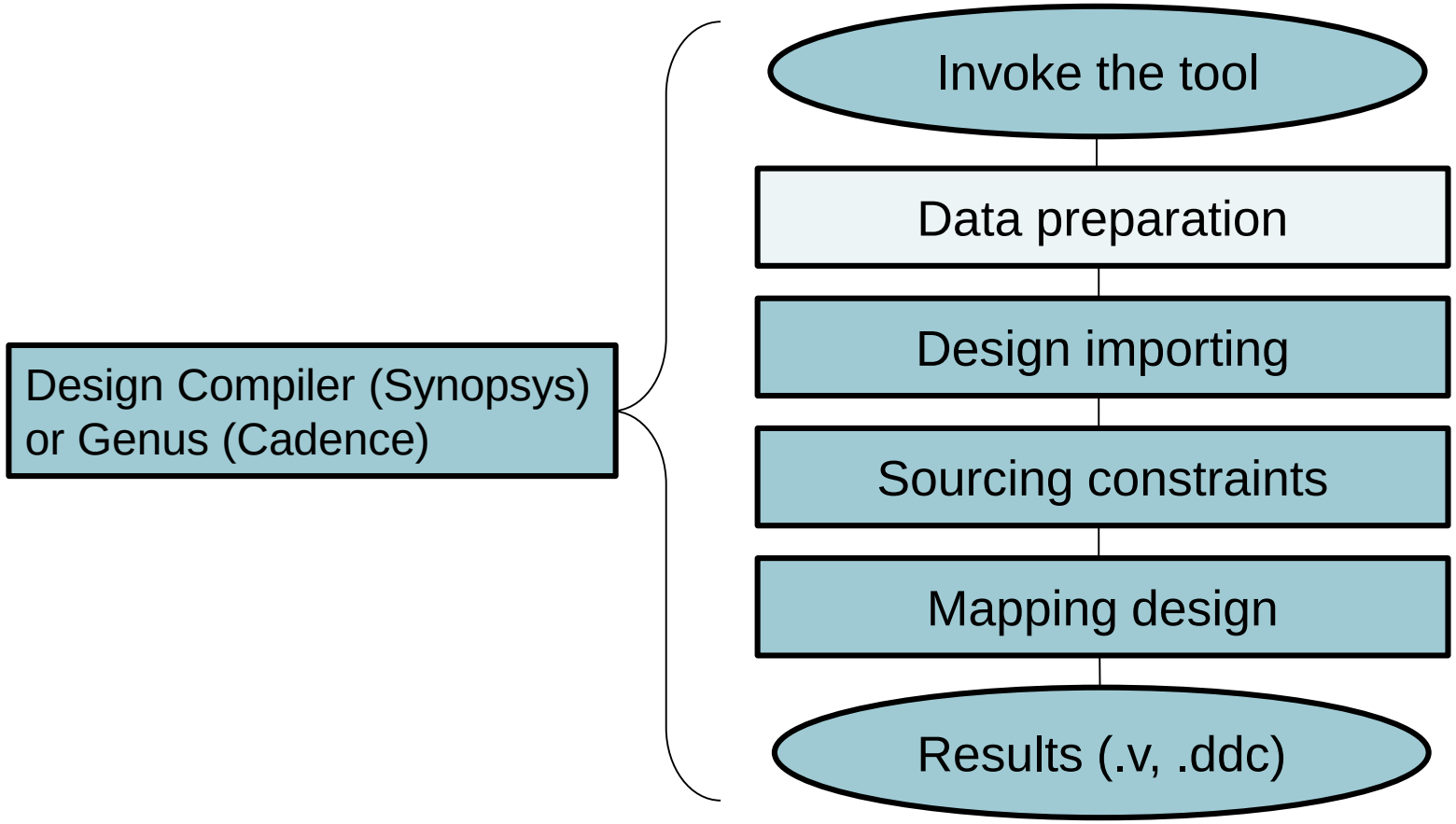
Technology Specific Circuit is get from independent one by replacing all components by real blocks (standard cells). This replacement process is also called mapping



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

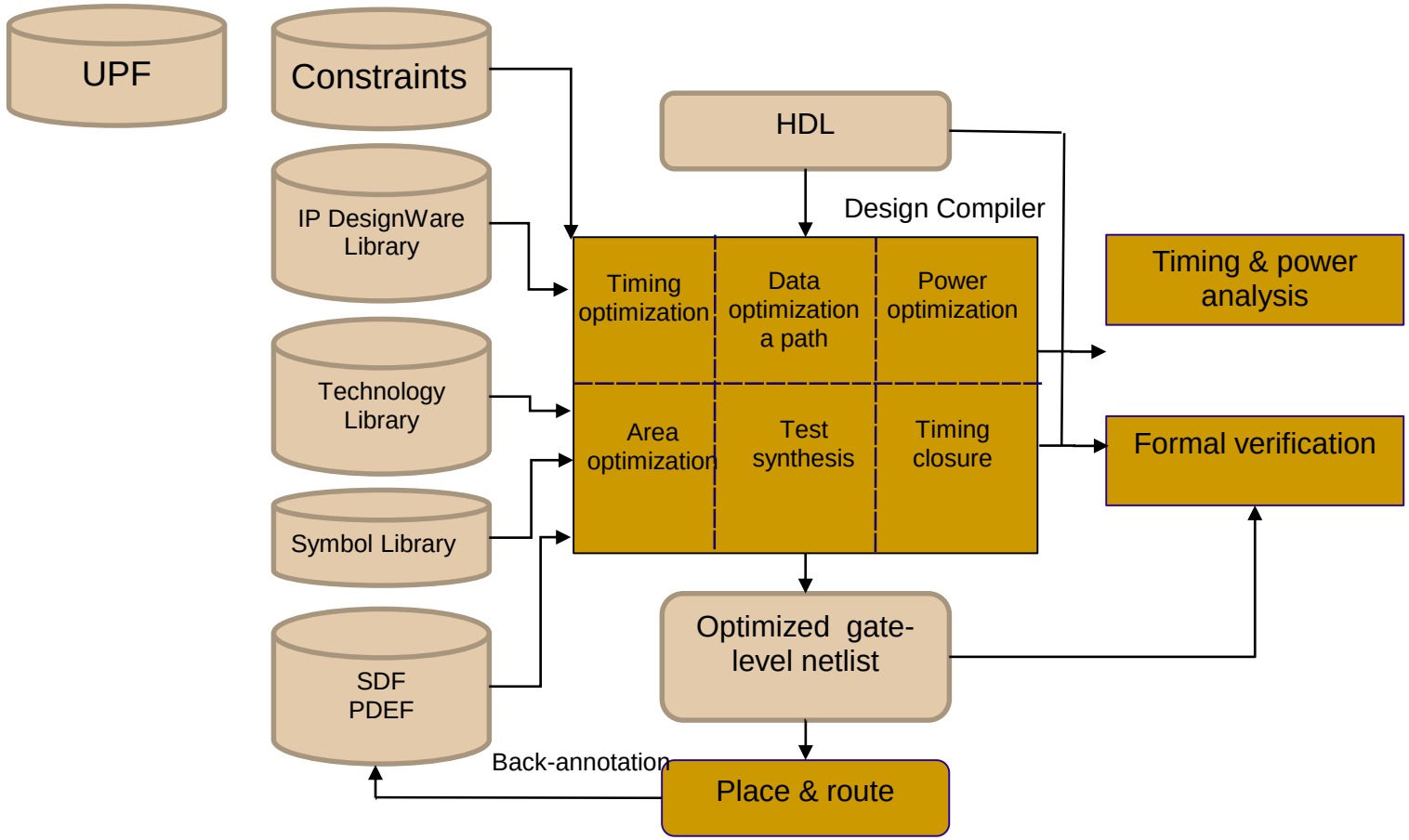
source: Synopsys

Logic Synthesis Steps



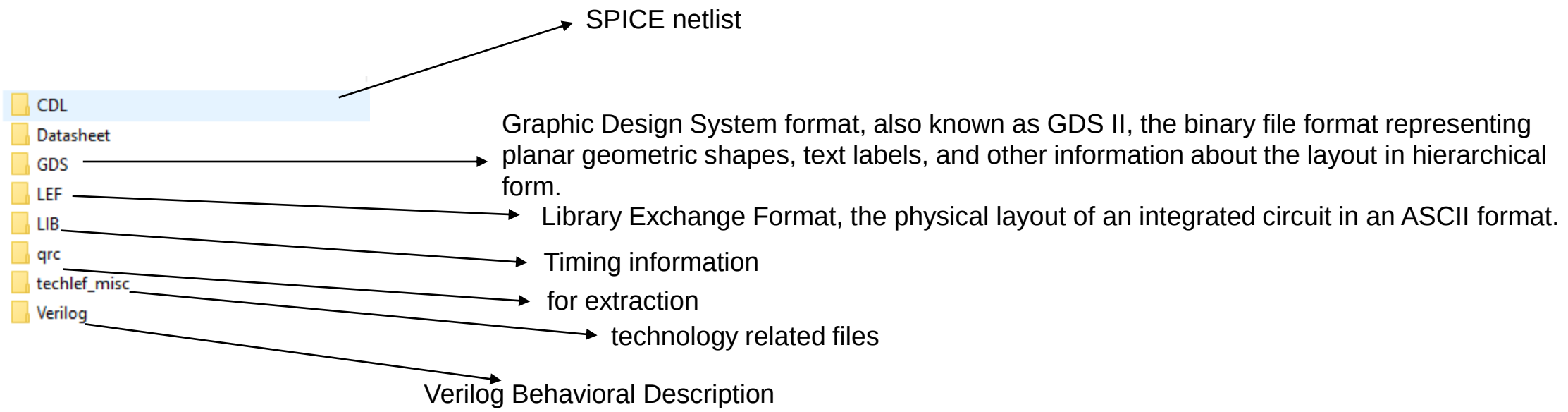
source: Synopsys

DC and Design Flow



This example is based on Synopsys Design Compiler

What is in a design library deliverable



- There are much more different formats? Why not just one? – Different information need different carriers, it is not possible to carry all information in one!
- Stannard cell library and process design kit (PDK) are different!

source: Cadence

Example of a .lib

```
library (asap7sc7p5t_AO_LVT_FF_ccs_201020) {
  /* Models written by Liberate 18.1.0.293 from Cadence Design Systems, Inc. on Wed Dec  2 15:22:14 MST 2020 */
  comment : "";
  date : "$Date: Wed Dec  2 11:09:27 2020 $";
  revision : "1.0";
  delay_model : table_lookup;
  capacitive_load_unit (1,ff);
  current_unit : "1mA";
  leakage_power_unit : "1pW";
  pulling_resistance_unit : "1kohm";
  time_unit : "1ps";
  voltage_unit : "1V";
  voltage_map (VDD, 0.77);
  voltage_map (VSS, 0);
 default_operating_conditions : PVT_0P77V_0C;
 output_current_template (ccs_template) {
   variable_1 : input_net_transition;
   variable_2 : total_output_net_capacitance;
   variable_3 : time;
 }
 lu_table_template (delay_template_7x7_x1) {
   variable_1 : input_net_transition;
   variable_2 : total_output_net_capacitance;
   index_1 ("5, 10, 20, 40, 80, 160, 320");
   index_2 ("0.72, 1.44, 2.88, 5.76, 11.52, 23.04, 46.08");
 }
```

source: Cadence

Concept of Design Corners

- Process Variations

In film thickness, lateral dimensions, dopings
Measured:
- From wafer to wafer
- From die to die – *inter-die*
- Across die – *intra-die* or *process tilt*

- Interconnect Variations

Line width and line spacing
- Photolithography
- Etching proximity effects

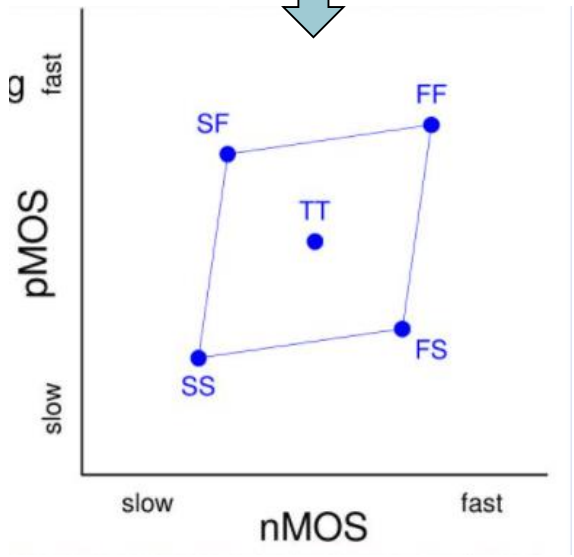
Metal and dielectric thickness
- Chemical Mechanical Polishing

Contact resistance
- Contact dimensions
- Etch and clean steps

Channel length L
- Photolithography proximity effects
- Optics deviations
- Plasma etch dependencies

Oxide thickness t_{ox}
- Well-controlled -- only significant between wafers

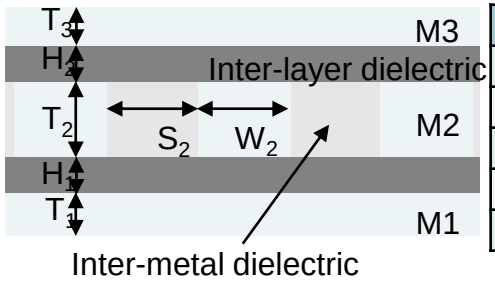
Threshold voltage V_t
- Varying dopings
- Annealing effects
- Mobile Q in gate oxide
- Discrete dopant variations (few dopant atoms in transistors)



source: UC Berkley

Design Corners (PVT)

- Process: FF (fast fast)/SF (slow fast)/SS (slow slow)/FS (fast slow)/TT (typical typical)
- Interconnect:



	ΔW	ΔT	ΔH
Typical	typical	typical	Typical
C-best	min	min	max
C-worst	max	max	min
RC-best	max	max	max
RC-worst	min	min	min

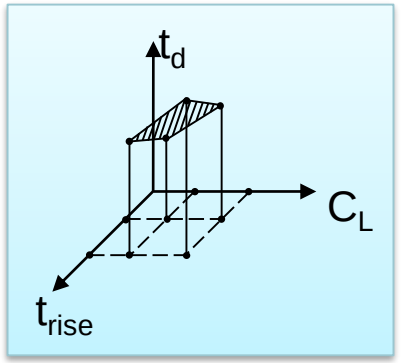
- Voltage: different voltage levels
- Temperature: From -40C to 125C, discrete values
- Threshold voltage: HVT, RVT, LVT
- Example Corner: SS_0p85V_N40C_RVT

source: Prof. Andrew K (UCSD)

Synopsys Liberty Format (.lib)

```
library (Digital_Std_Lib) {
      technology (cmos);
      delay_model : table_lookup;
      cell(AND2) {
            area : 2;
            pin(A) {
                        direction : input;
            }
            pin(B) {
                        direction : input;
            }
            pin(Z) {
                        direction : output;
                        function : "A*B";
                        timing() {
                           related_pin : "A" ;
                           timing_type : "combinational" ;
                           cell_rise(…) {
                                   index_1("0.016, 0.032, 0.064");
                                   index_2("2, 4");
                                   values("1.0020, 1.1280, 3.547 ", \
                                          "1.0080, 1.1310, 3.847 "   );
                           }
                        }
            } /* end of pin */
      } /* end of cell */
} /* end of library*/
```

		t_{rise}		
		0.016	0.032	0.064
C_L	2	1.0020	1.1280	3.547
	4	1.0080	1.1310	1.1310

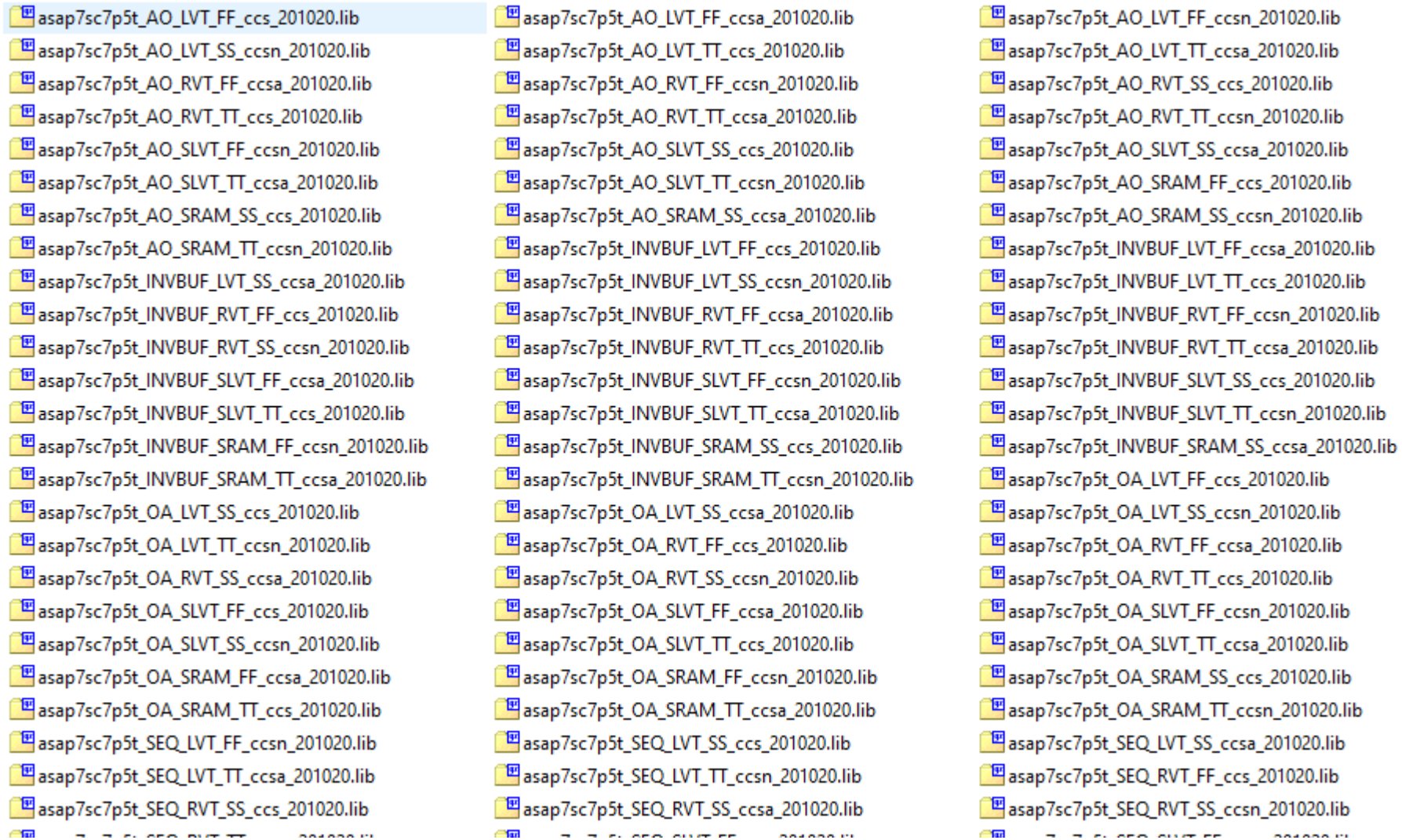


Operating Conditions

- name
  - The name identifies the set of operating conditions
- process
  - The scaling factor accounts for variations in the outcome of the actual semiconductor manufacturing steps. This factor is typically 1.0 for normal operating conditions
- Temperature
  - The ambient temperature in which the design is to operate
- Voltage
  - The operating voltage of the design
- tree_type
  - The definition for the environment interconnect model

```
library(){

nom_voltage : 1.160000;
nom_temperature : 25.000000;
nom_process : 1.01;

operating_conditions (ff0p85v25c) {
        process : 1.01;
        voltage : 1.160000;
        temperature : 25.000000;
        tree_type : "worst_case_tree";
        }
default_operating_conditions : “ff0p85v25c";


}
```

Example: Std Cell Lib Deliverable



source: ASU

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	1.08	1.2	1.32	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 ≤ 100 mW
Area = 2 um^2
18MHz < Frequency < 30 MHz

Design constraints are used as input for synthesis.

Synopsys Design Contraints File (.sdc)

- Similar to XDC file in Xilinx tool chain
- Also compatible in Cadence tools and other EDA tools
- Contains design constraints and timing assignments in the industry-standard Synopsys Design Constraints format.
- Defines clock, timing exceptions, system interface, design rule constraints, etc.
- More details about sdc: https://www.teamvlsi.com/2020/05/sdc-synopsys-design-constraint-file-in.html

1. SDC Version
2. Units

System Interface

3. Set driving cells
4. Set load

Design rule constraints

5. Set maximum fanout
6. Set maximum Transition

Timing constraints

7. Create Clock
8. Create Generated Clock
9. Group Path
10. Clock Uncertainty
11. Clock Latency
12. Input Delay
13. Output Delay

Timing Exception

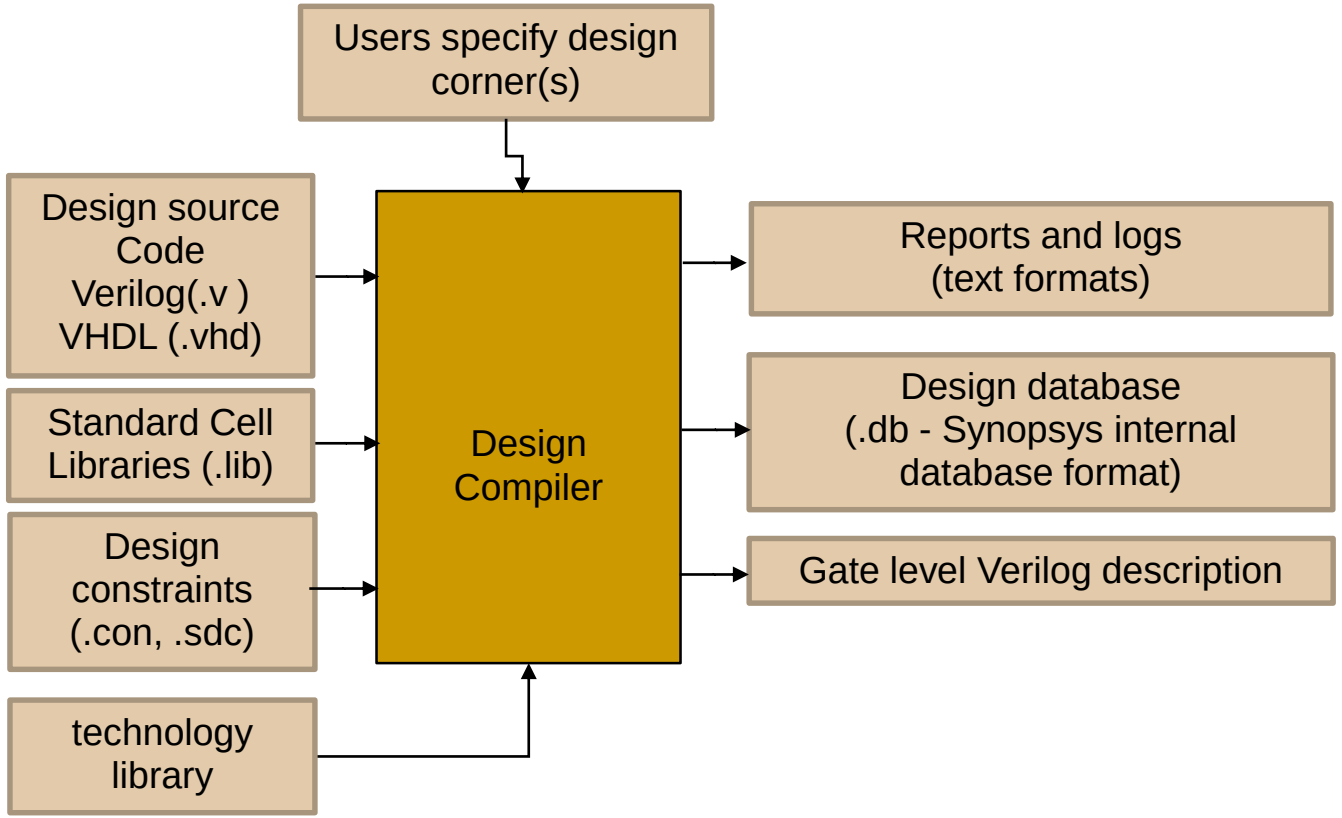
14. Multicycle Path
15. False Path

A sdc example

```
# Constrain clock port clk with a 10-ns requirement
create_clock -period 10 [get_ports clk]

# Automatically apply a generate clock on the output of phase-locked loops (PLLs)
# This command can be safely left in the SDC even if no PLLs exist in the design

derive_pll_clocks

# Constrain the input I/O path
set_input_delay -clock clk -max 3 [all_inputs]
set_input_delay -clock clk -min 2 [all_inputs]

# Constrain the output I/O path
set_output_delay -clock clk -max 3 [all_inputs]
```

source: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-examples/quartus/exm-tq-bsic-sdc-template.html

Input and Output Files



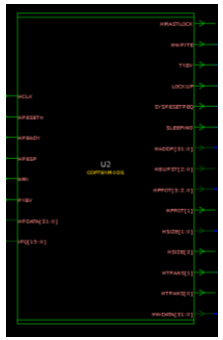
This example is based on Synopsys Design Compiler

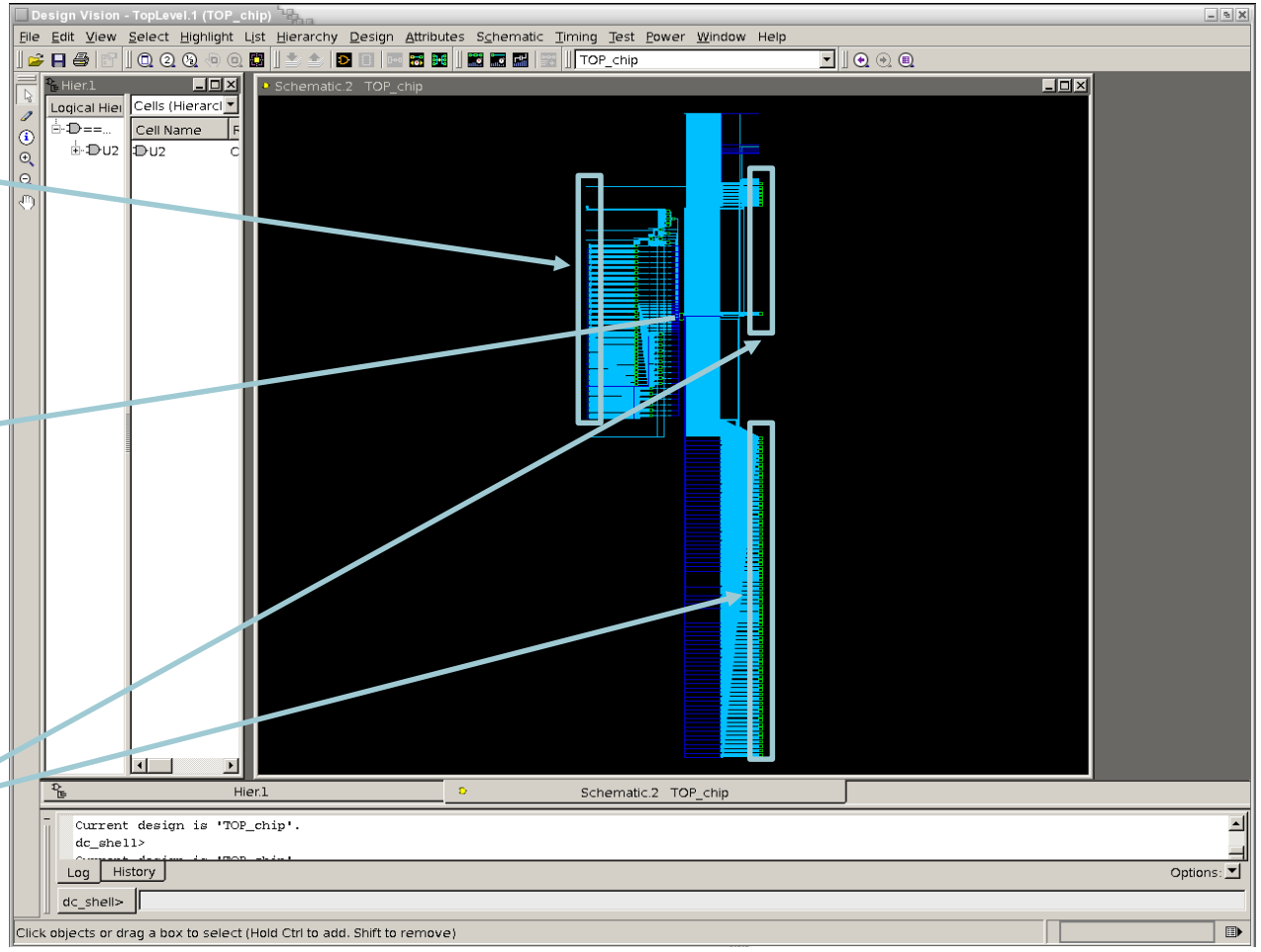
Design View After Importing

Input I/O

CORE

Output I/O



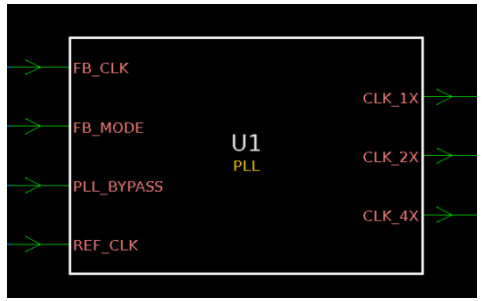


This example is based on Synopsys Design Compiler

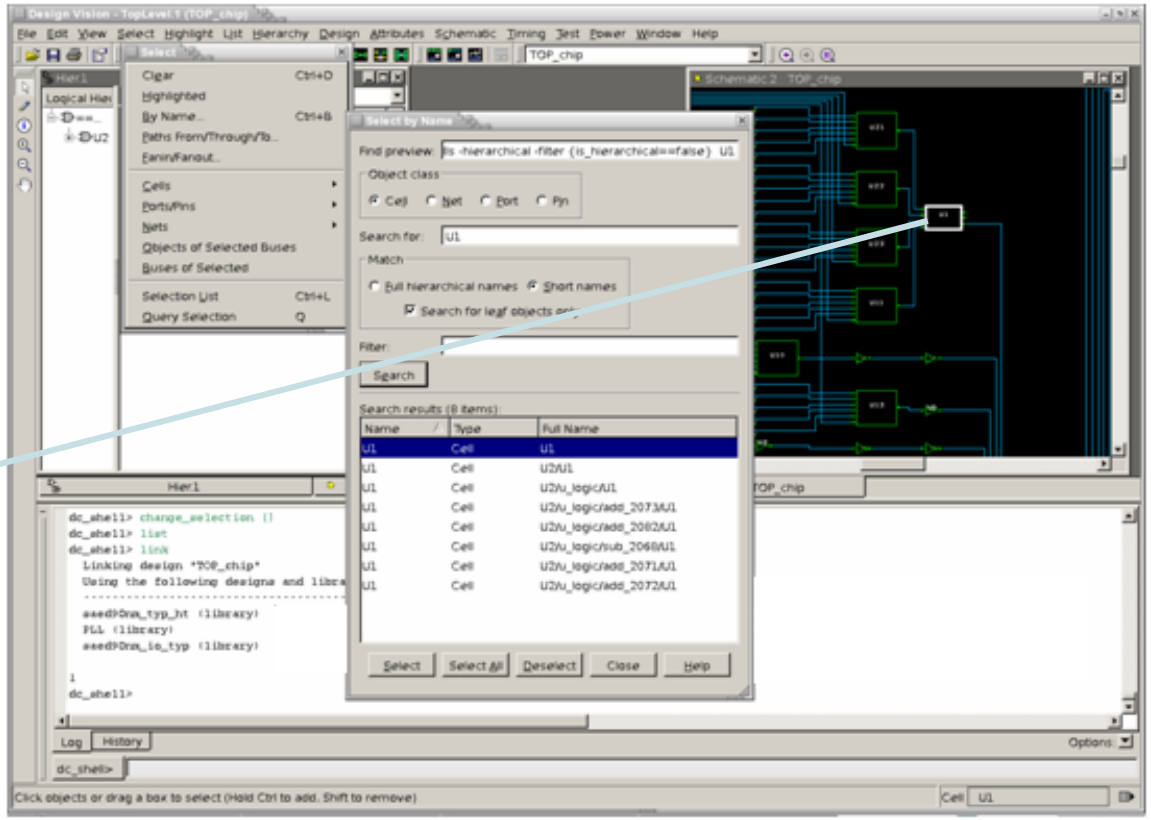
Navigating in Design

Select → By Name

Selects cells, nets, ports, or pin by name, based on matching a specified pattern.

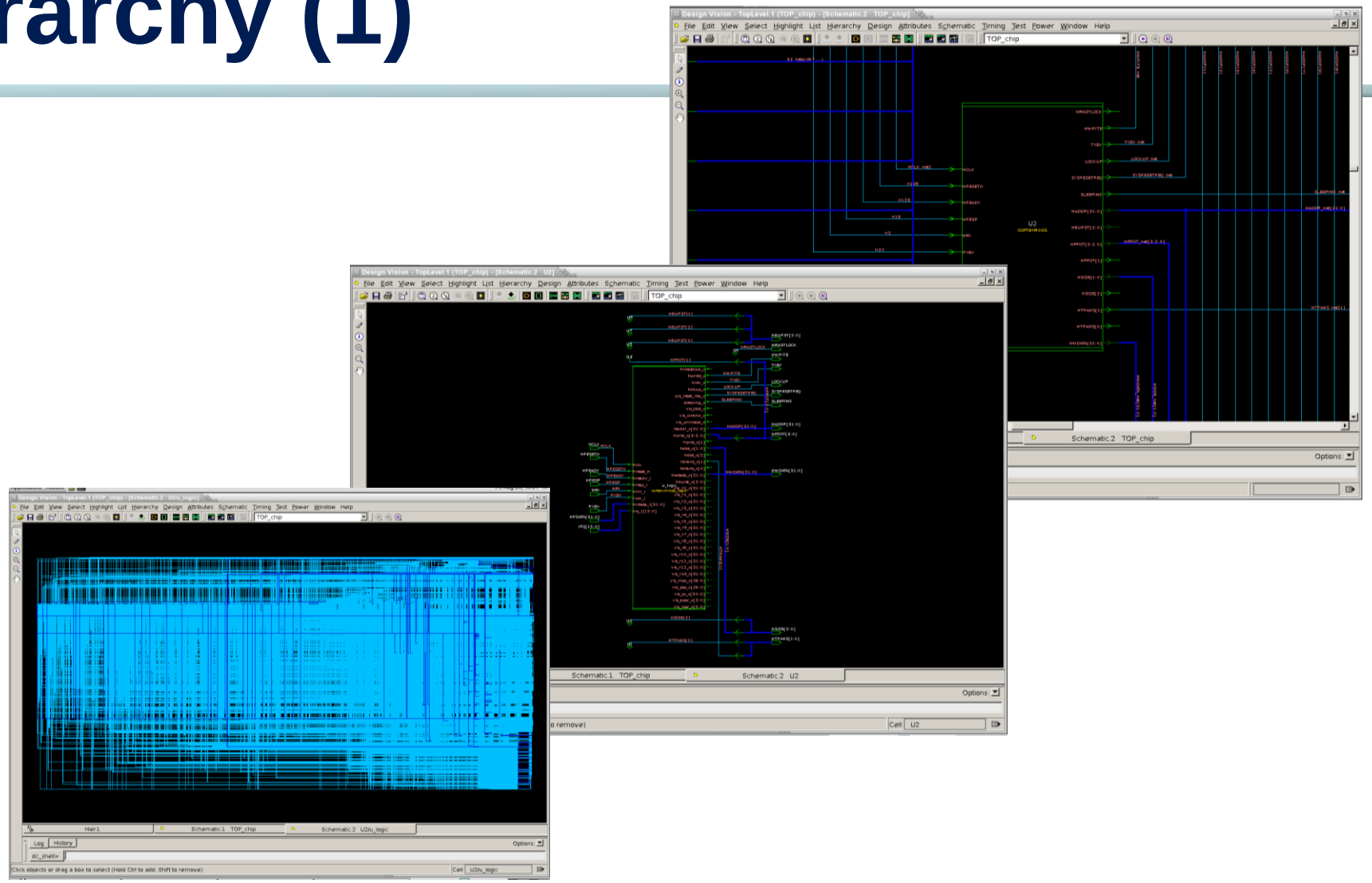


Selects cells



This example is based on Synopsys Design Compiler

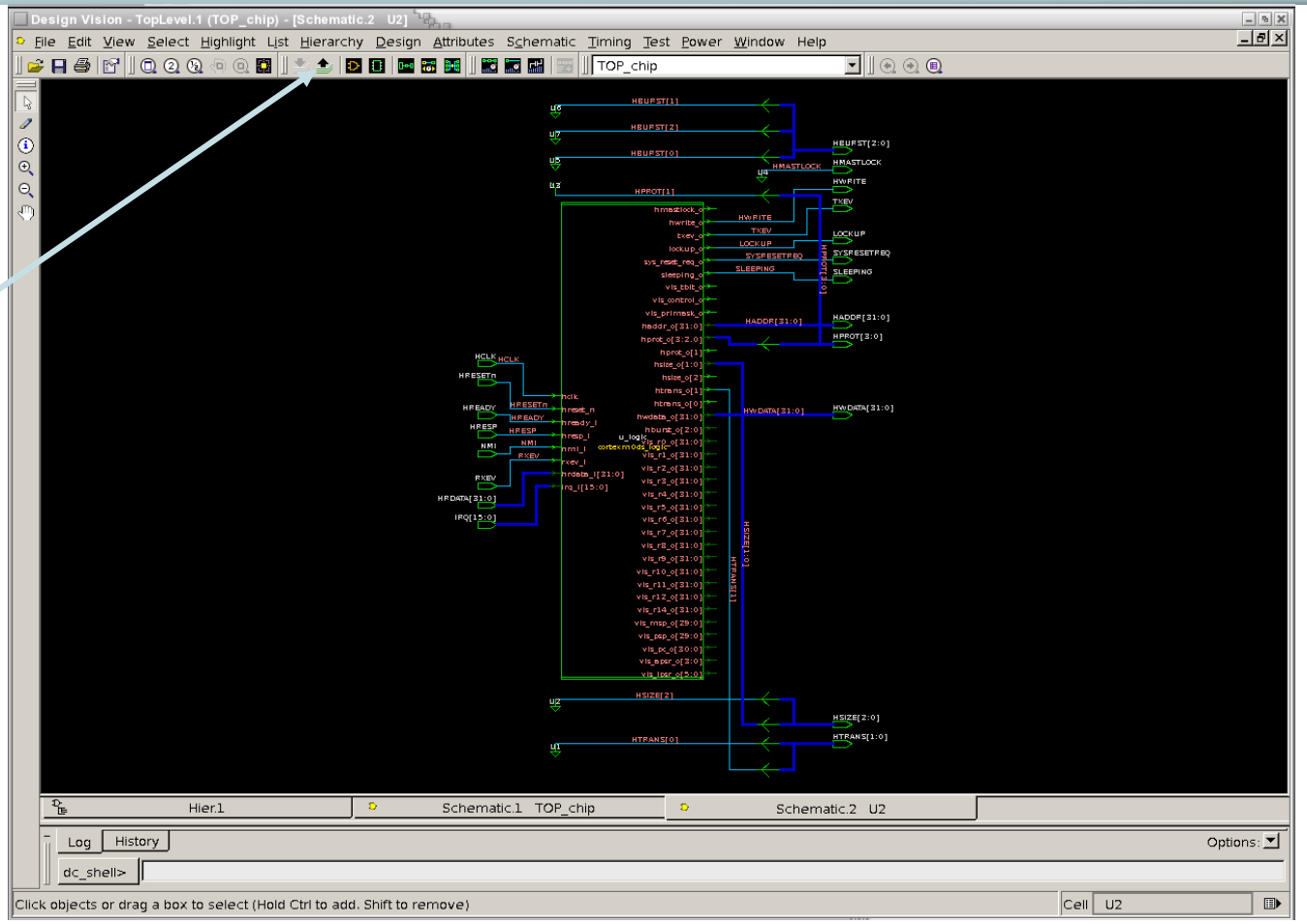
Hierarchy (1)



This example is based on Synopsys Design Compiler

Hierarchy (2)

Move up one level



This example is based on Synopsys Design Compiler

TCL/Tk Language

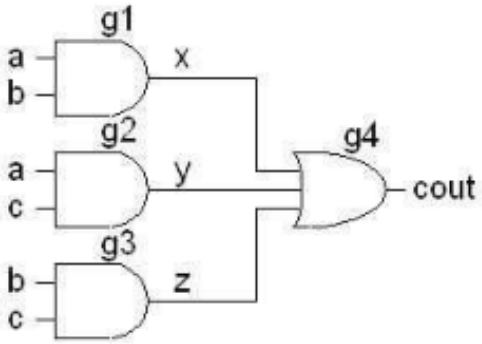
- Tool Command Language
- a scripting language, ugly but simple
- used for Web and desktop applications, networking, administration, testing, rapid prototyping, scripted applications and graphical user interfaces (GUI)
- available across Unix, Windows and Mac OS platforms
- Used in most of the mainstream EDA tools/FPGA tools
- Examples:

```
set  myName XXX
puts "My Name is $myName”
set class CPSC-481; puts -nonewline $class
```

Gate level netlist

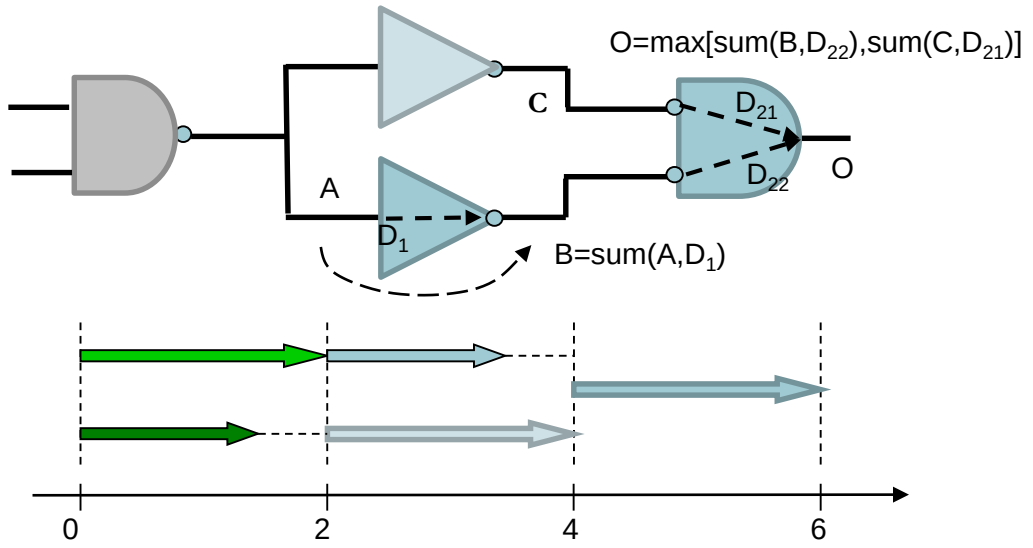
```
module carry(input  a, b, c,
             output cout)

  wire   x, y, z;

  and  g1(x, a, b);
  and  g2(y, a, c);
  and  g3(z, b, c);
  or   g4(cout, x, y, z);
endmodule
```

standard cell name



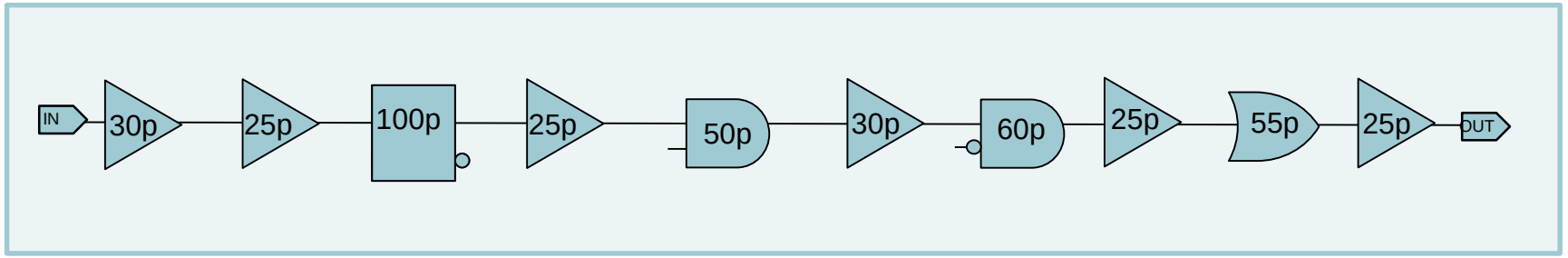
Static Timing Analysis (STA)

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



Example

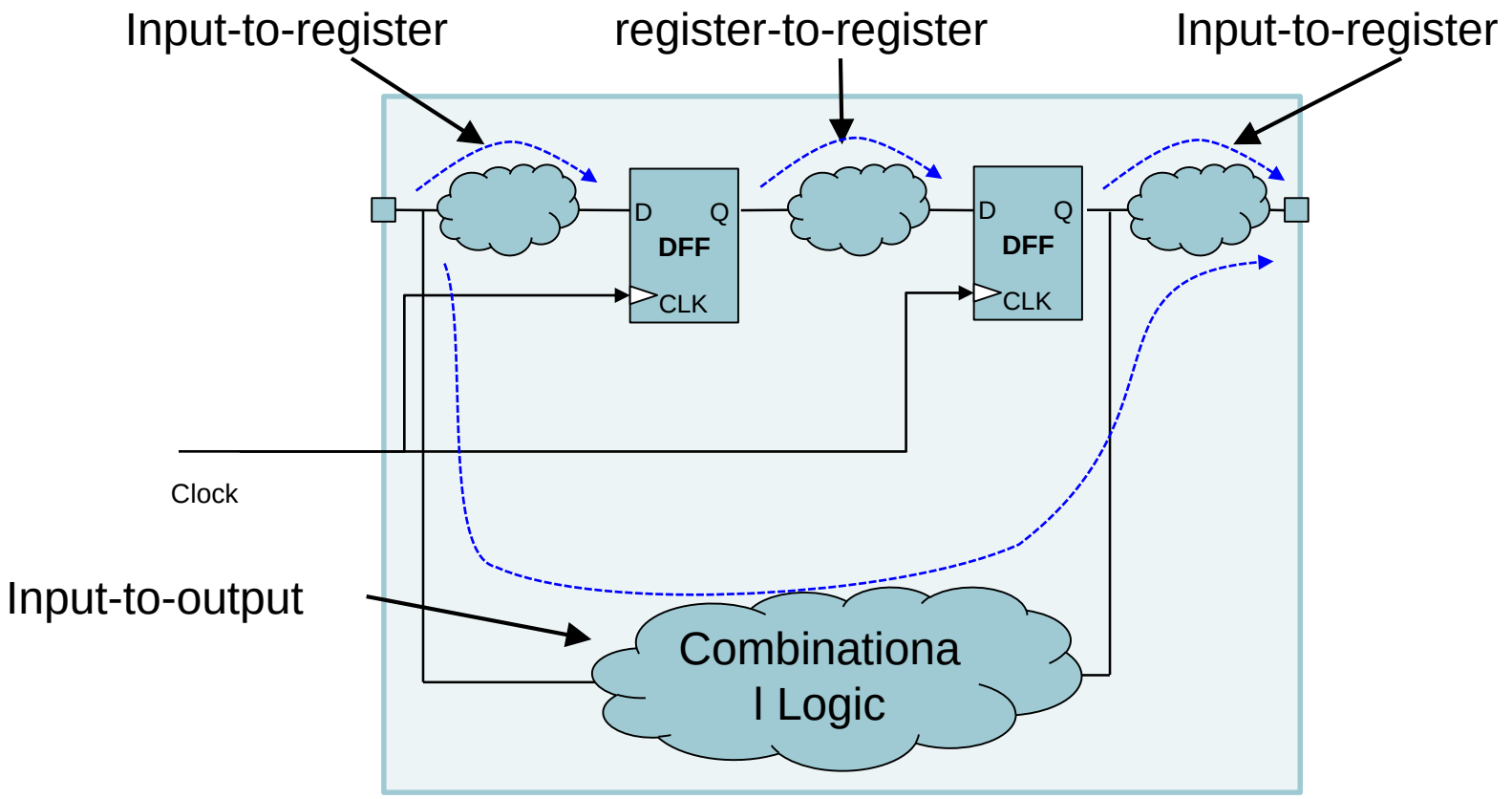
- Static Timing Analysis (STA) is a method of computing the expected timing of a digital circuit without requiring simulation



Delay=30p+25p+100p+25p+50p+30p+60p+25p+55p+25p=425p

Timing 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

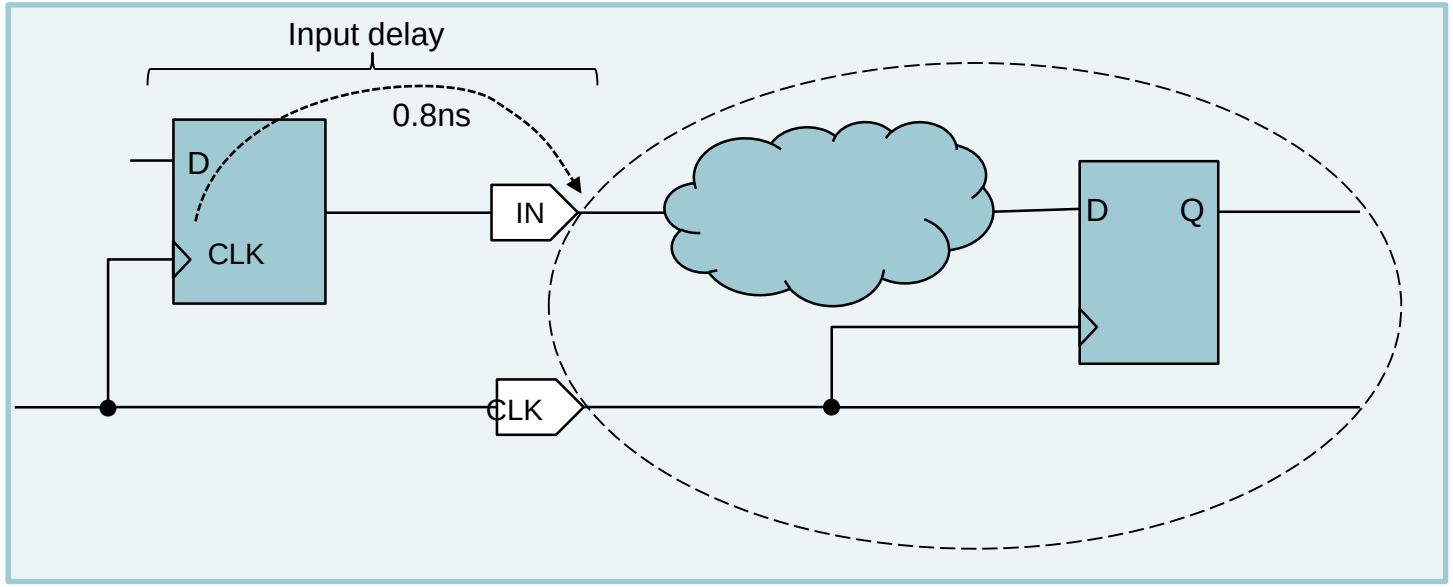
- The startpoint is a place where data is launched by a clock edge:
  - a sequential element's clock pin
  - an input port of the design
- Data is propagated through the path and then captured at the endpoint by another clock edge:
  - a sequential element's data input pin
  - an output port of the design

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

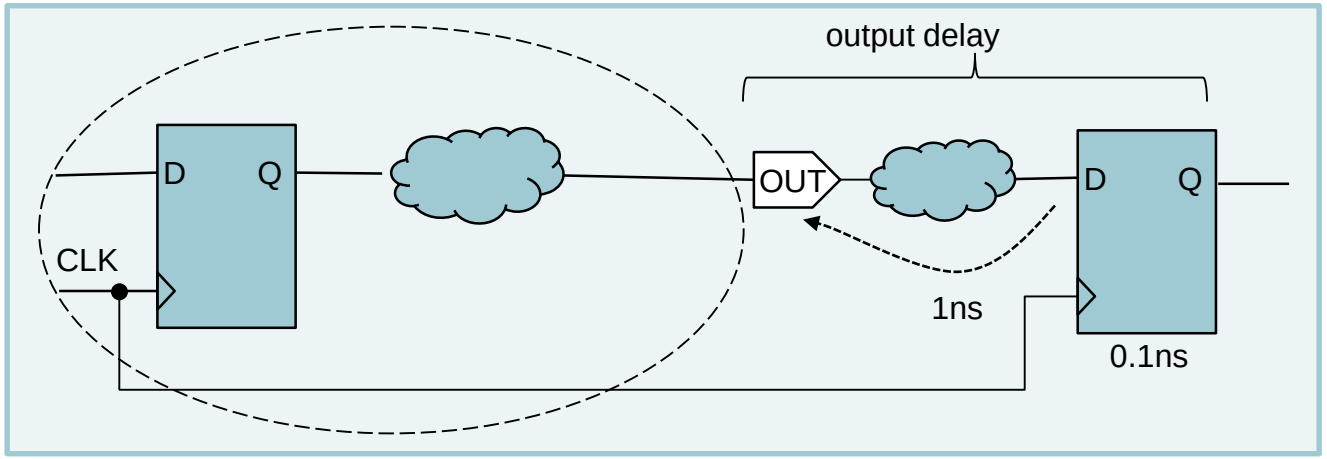
- The input delay is the CLK-to-IN external delay



- 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

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

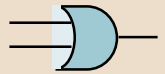
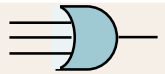
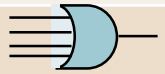
Timing report after synthesis

```
 Startpoint: hash_core/permutation/current_round_reg[3]
             (rising edge-triggered flip-flop clocked by clock)
  Endpoint: hash_core/permutation/x1_Reg_reg[31]
            (rising edge-triggered flip-flop clocked by clock)
  Path Group: clock
  Path Type: max

  Point                                                    Incr       Path
  -----------------------------------------------------------------------------
  clock clock (rise edge)                                  0.00       0.00
  clock network delay (propagated)                         0.32       0.32
  hash_core/permutation/current_round_reg[3]/CLK (DFFX1_HVT)
                                                           0.00       0.32 r
  hash_core/permutation/current_round_reg[3]/Q (DFFX1_HVT)
                                                           0.17       0.50 f
  hash_core/permutation/single_round/io_round_in[3] (permutation)
                                                           0.00       0.50 f
  hash_core/permutation/single_round/addition/io_round_in[3] (addition_layer)
                                                           0.00       0.50 f
  hash_core/permutation/single_round/addition/U16/Y (INVX4_HVT)
                                                           0.02 &     0.52 r
 hash_core/permutation/single_round/addition/U13/Y (NAND2X2_HVT)
                                                           0.09 &     0.61 f
  hash_core/permutation/single_round/addition/U11/Y (XOR2X1_HVT)
                                                           0.14 &     0.75 r
...

  hash_core/U61/Y (AND2X1_HVT)                             0.04 &     2.47 r
  hash_core/permutation/io_s_in[223] (permutation_new)
                                                           0.00       2.47 r
  hash_core/permutation/U84/Y (AO22X1_HVT)                 0.07 &     2.54 r
  hash_core/permutation/x1_Reg_reg[31]/D (DFFX1_HVT)       0.00 &     2.54 r
  data arrival time                                                   2.54

  clock clock (rise edge)                                  1.60       1.60
  clock network delay (propagated)                         0.30       1.90
  clock uncertainty                                       -0.03       1.87
  hash_core/permutation/x1_Reg_reg[31]/CLK (DFFX1_HVT)
                                                           0.00       1.87 r
  library setup time                                      -0.04       1.83
  data required time                                                  1.83
  -----------------------------------------------------------------------------
  data required time                                                  1.83
  data arrival time                                                  -2.54
  -----------------------------------------------------------------------------
  slack (VIOLATED)                                                   -0.71
```

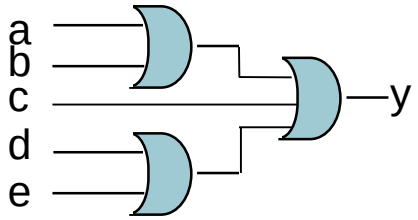
Main Optimization Trade-Offs

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

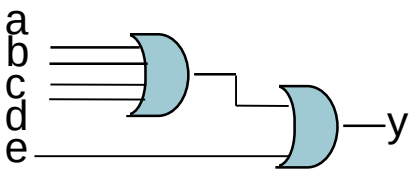
- Timing optimization
  - Goal: small delays
- Power optimization
  - Goal: low power consumption
- Area optimization
  - Goal: small area

Cell	Power
	2
	2.5
	3

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



Total power:~6



Total power:~5

What can go wrong with synthesis?

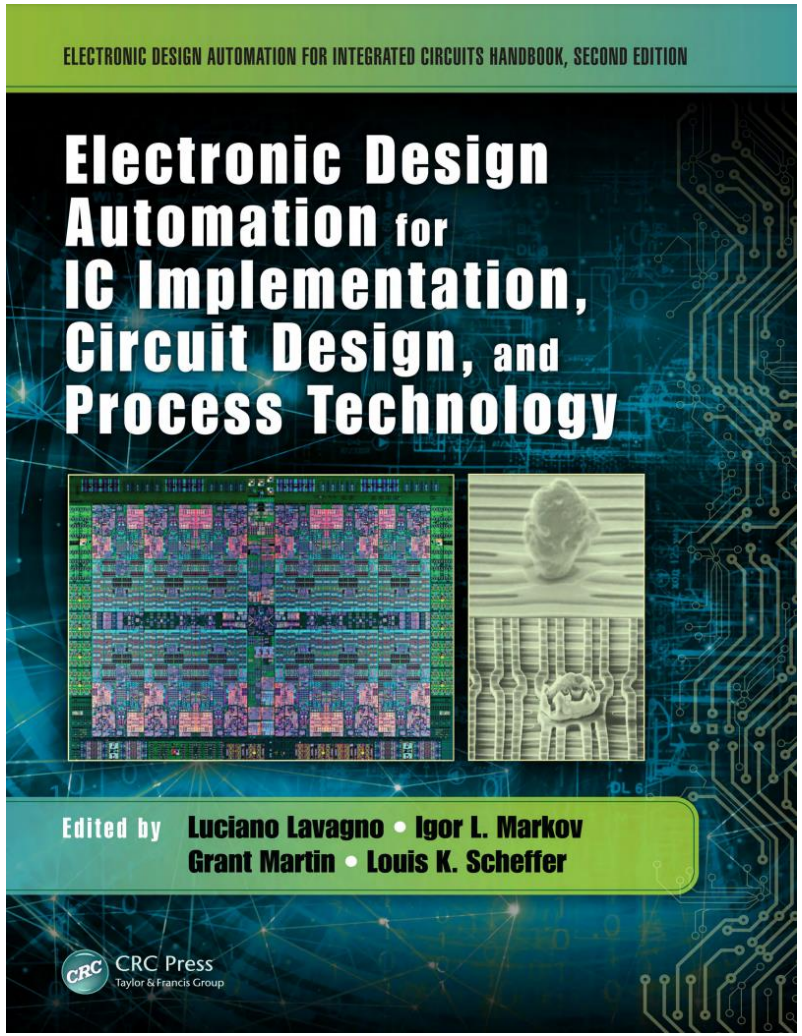
- This generated gate level netlist is the starting point for all of the following steps, so it is very important
- Need to check immediately after synthesis
- Picked the wrong corner
- Less optimizations
- Timing is too bad
- Mismatch with original RTL code (failed formal verification)
- Too many levels of logic – Need to feed back to designers
- Synthesis is able to identify many potential design problems

Where are we Heading?

- ASIC Design Flow II

Action Items

- HW#3 is due on Nov. 11th!
- Reading Materials
  - Slides
  - Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology (on canvas)



Acknowledgement

Slides in this topic are inspired in part by material developed and copyright by:

- Synopsys Courseware