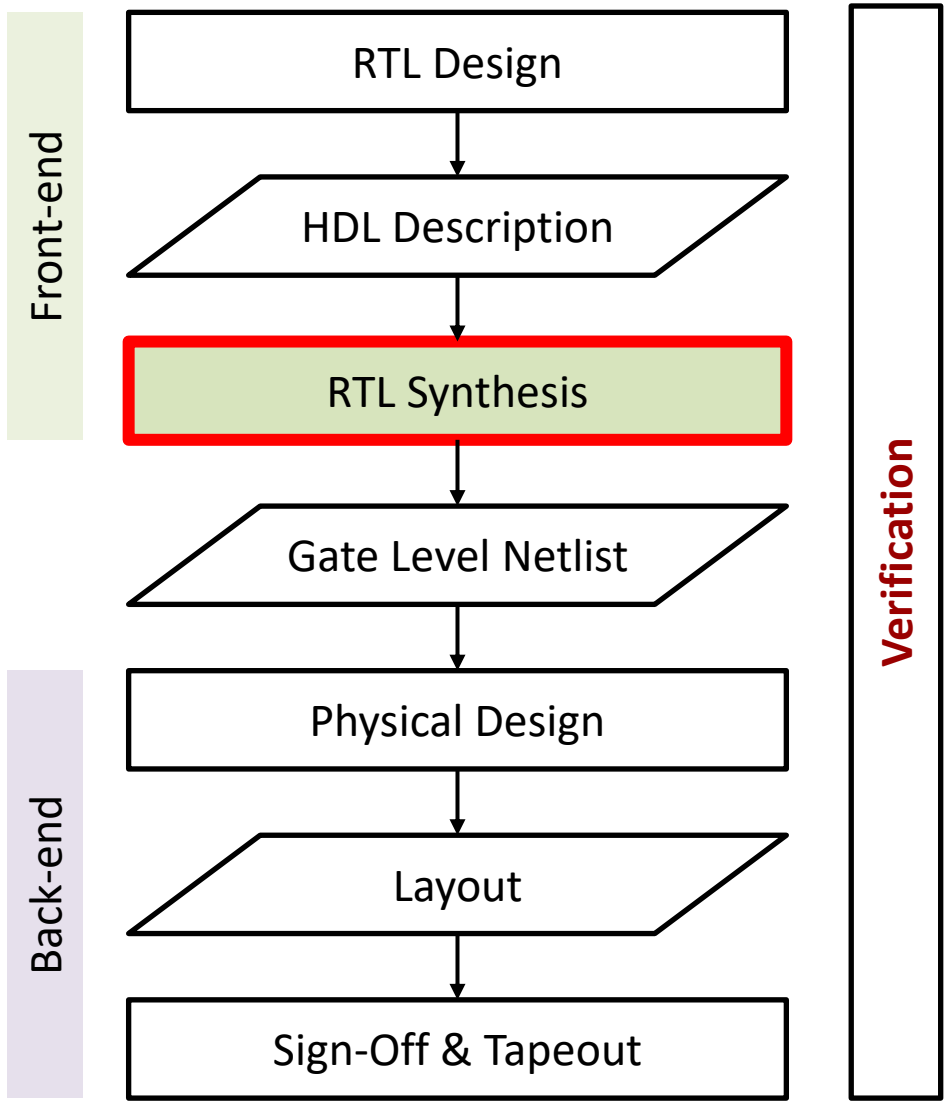
EE-429
Fundamentals of VLSI Design

The Semi-Custom Frontend: Synthesis

Andreas Burg

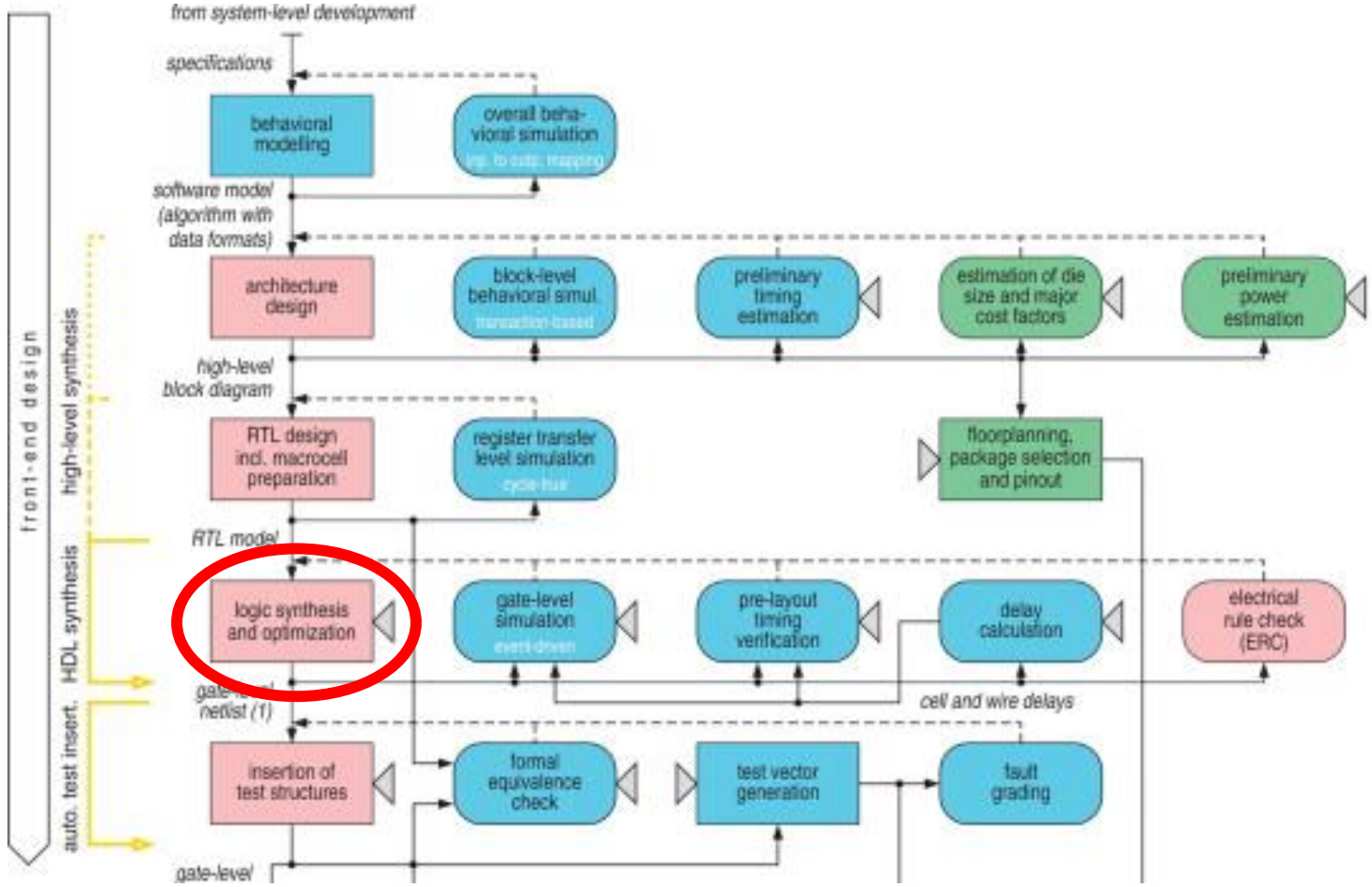
Semi-Custom (Digital) ASIC Design Flow

- **Semi-custom design flow:**
  - Starts from a Register Transfer Level description in a hardware description language (HDL)
  - **Front-end flow**: handles the transition from RTL to the gate level
  - Back-end flow: handles the transition from a netlist to physical design data
- **Each step is** always **accompanied** by **verification**
  - Check functionality, timing, and physical constraints



Front-End Design Flow Details

- **Provide a functional netlist for the backend flow that has a chance (with further optimization) to meet requirements after physical design**



Logic Synthesis

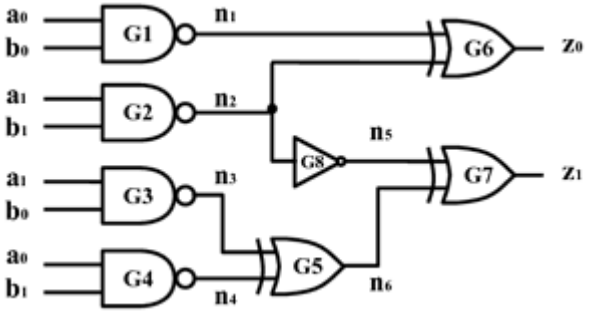
- **Logic synthesis converts a generic RTL design into a** technology-specific (gate-level) **netlist** of standard-cells

RTL code

Timing constraints

Synthesis scripts

netlist

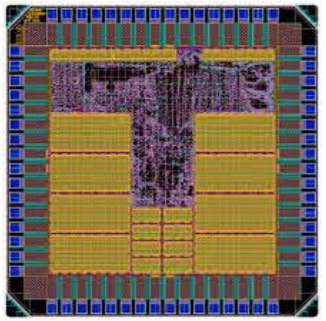


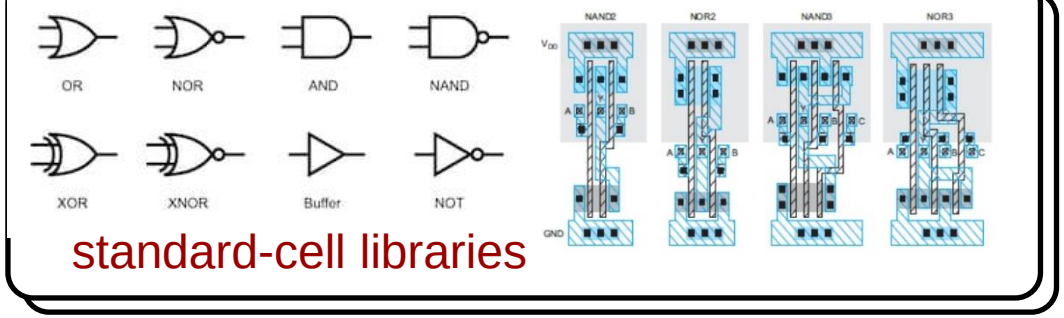
Floorplan

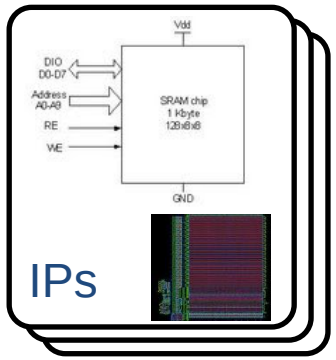
Clock tree constraints

Power planning

Backend scripts



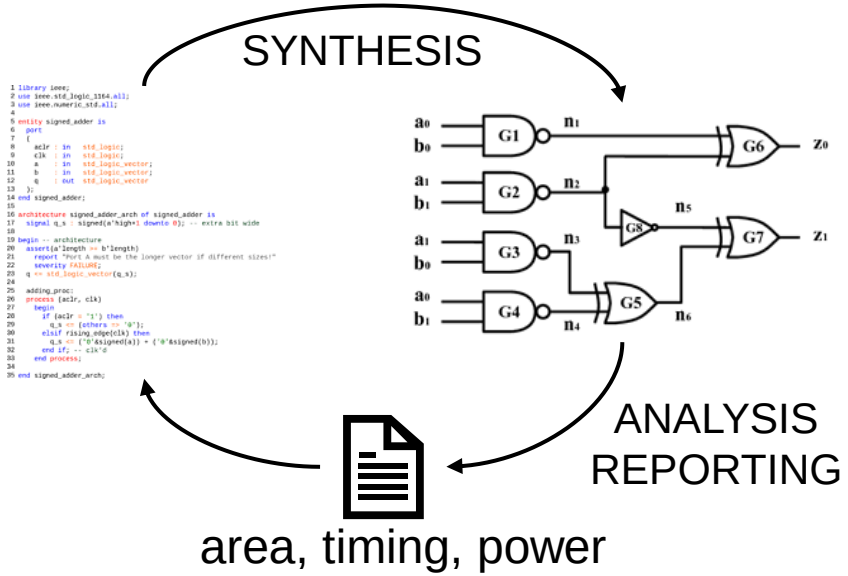




Technology Files

Synthesis in the Design Flow

- **When designing a chip, you iterate between**
  - updating/**improving a design**
  - **synthesizing** the design
  - **analysing** the design parameters (e.g., area/timing)



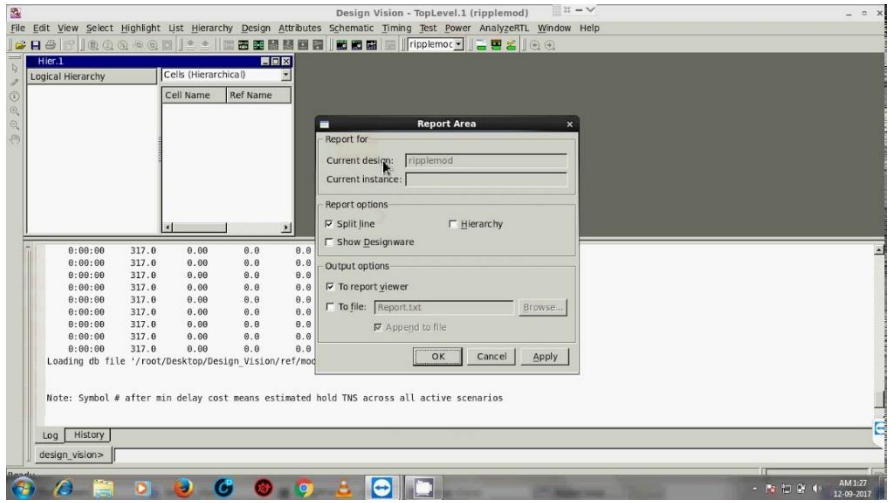
- The **analysis** of design characteristics **is a part of the synthesis** process
  - Static timing analysis
  - Area analysis
  - Power analysis

- **Synthesis and analysis** of a complex design **can take days and involve hundreds of steps** and commands
  - Trust me: you will do every step more than once even when you think the design is finished

Synthesis in the Design Flow

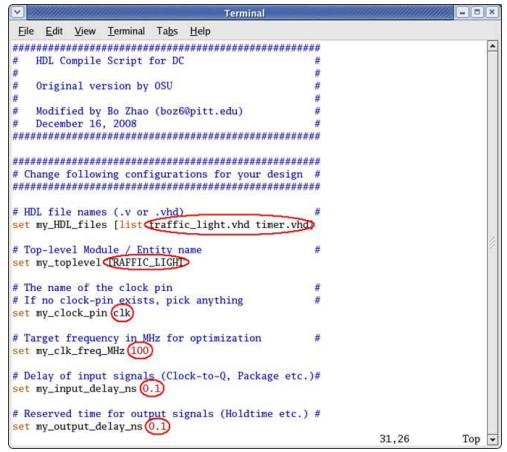
- **Synthesis process needs to be automated** to be efficient and reproducible
  - All modern EDA tools support extensive scripting, usually based on TCL

- **EDA is always a transition from interactive work to automated scripts**
  - First iteration: develop your commands and constraints interactively and keep in scripts
    - Sometimes done using the GUI, but **usually** much faster and more flexible **on the command line**
  - Later: run and modify your scripts



Interactively develop and build your scripts

Copy/paste from scripts for step-by-step debug, then run in batch mode

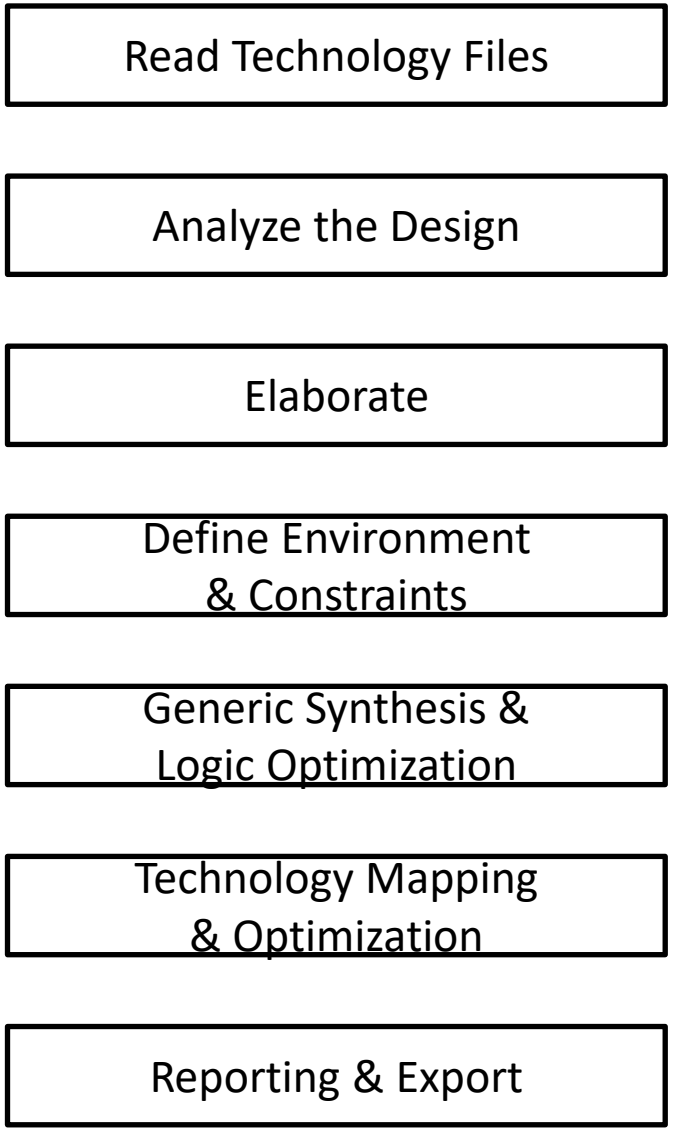


RTL Synthesis Objectives

- **RTL synthesis is an optimization process with various objectives:**
  - **Functionality**: turn an abstract behavioural description of logic into a
  - **Design rules**: assure compliance with basic electrical design rules that ensure accuracy and compliance with library and process specifications
  - **Timing** (set by the designer): meet requirements on speed/timing (e.g., clock period)
  - **Area** (set by the designer): minimize area (or even meet a maximum area requirement)
  - **Power** (set by the designer): minimize active and leakage power consumption

RTL Synthesis Steps (Overview)

- **Synthesis process is broken into steps**
  - Tool setup defines technology information and libraries
  - Read and expand the HDL design into a form that can be processed further and is more structured than the HDL code
  - Definition of the design requirements and constraints by the designer
  - Synthesis and optimization process mostly done “inside” the tool with some options to break it up into parts for complex designs
  - Analysis and reporting to inspect the results and enable insights into issues to guide design improvements

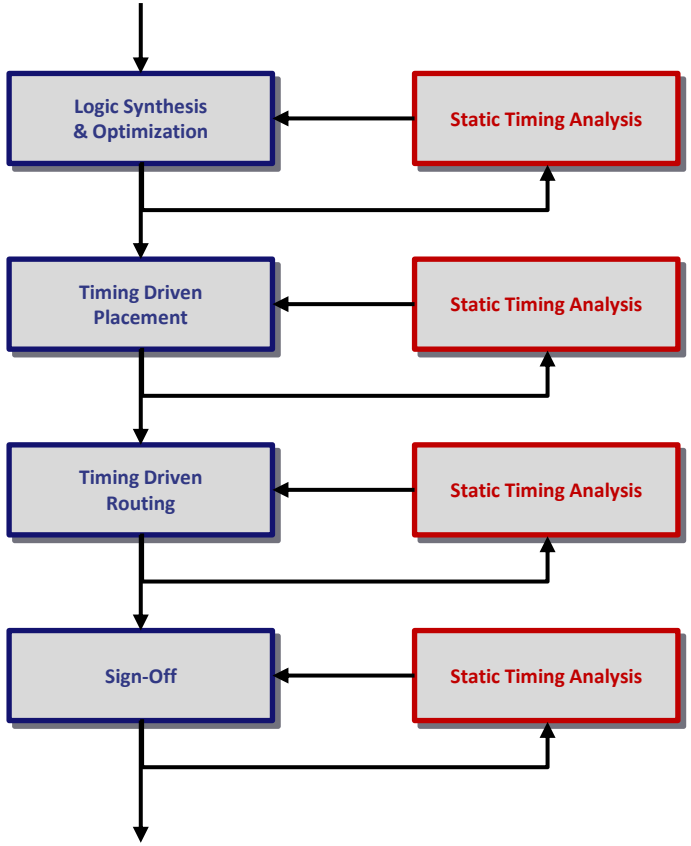


Iterations: Synthesis and Static Timing Analysis

- **Static timing analysis (STA) determines the delay of a circuit, checks timing requirements, and identifies bottlenecks for optimization**

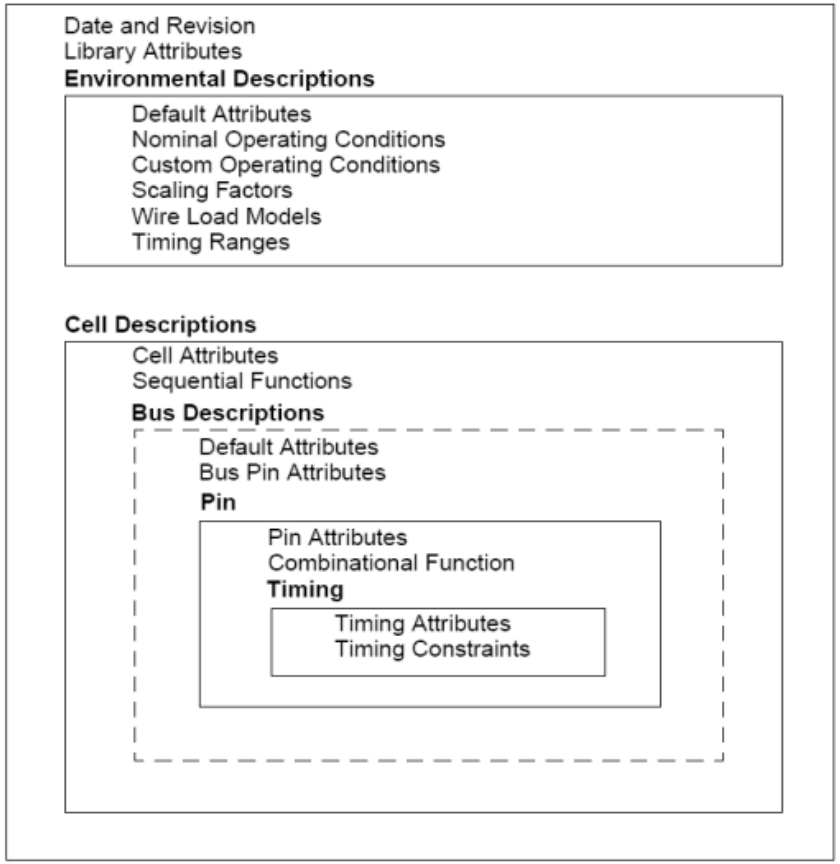
- **STA is a key component of digital circuit design**
  - Mostly managed by automatic EDA tools
  - Used in several stages of the design
    - during synthesis to drive logic optimization
    - during placement to optimize cell locations
    - during routing to consider parasitics
    - during sign-off to confirm timing requirements

- **Need an efficient way to rapidly analyse timing of large designs**



Timing View with Liberty (.lib) Files

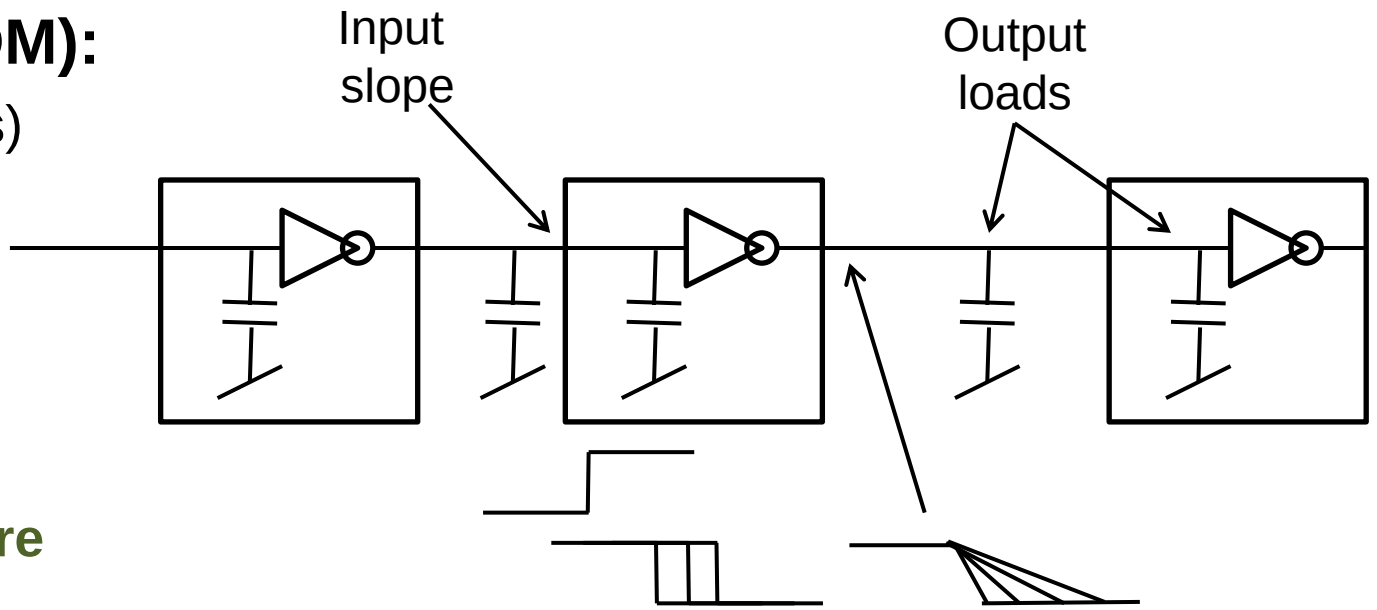
- **Analog simulation is too complex, but models such as the logical effort model are not sufficiently accurate**
  - **.lib files provide abstract timing information** about a library or a single cell
- **Lib files are comprised of various parts:**
  - **General information** about the library:
    - Operating conditions, wireload models, general information on the delay models
  - **Cell information** that is specific to each cell:
    - Cell function, Cell area and footprint
  - **Pin information** for each pin of a cell:
    - Propagation delays, transition times (slopes), timing requirements (setup/hold), capacitance, active power, leakage power, direction, type



STA: Delay Calculation

- **Computing cell delays requires an abstract model of the** cell **delay** that is adjusted to the “context” of the cell

- **Non-linear delay model (NLDM):**
  - Cell-delay (and constraints for FFs) depends on
    - **Input slope :** output slope of the preceding cell
    - **Output load**
  - **Cell-delays and output-slopes are pre-computed** and stored



Delay depends on:
- Output load
- Input slope

Slope depends on
- Output load
- Input slope

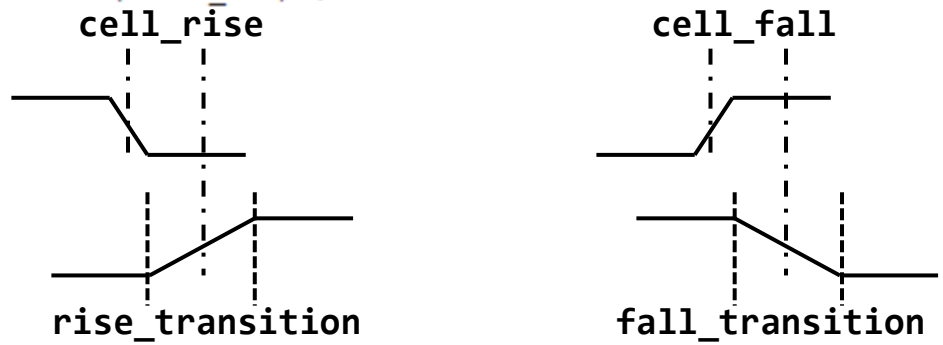
NLDM Delay Information in .lib Files

- **.lib files provide delay information in plain-text (ASCII) tables**
  - Tables are parameterized on slope and transition of related input pin
  - For combinational cells we use
    - Propagation delays for rising and falling edges
    - Slopes as rise and fall times
  - For sequential cells, we use
    - Setup and hold constraints for rising and falling inputs related to

```
timing() {
  related_pin : "I1";
  timing_type : combinational;
  timing_sense : positive_unate;
  cell_rise(DELAY_6x6) {
    index_1("0.026971,0.047698,0.103244,0.334223,0.800166,1.965874");
    index_2("0.002000,0.005759,0.016584,0.047756,0.137519,0.396000");
    values("0.089826,0.101375,0.129890,0.203833,0.411923,1.008867",\
           "0.093948,0.105494,0.133989,0.207890,0.416028,1.012842",\
           "0.103992,0.115593,0.144121,0.217939,0.426091,1.022870",\
           "0.125361,0.137351,0.166452,0.240225,0.448016,1.044857",\
           "0.138130,0.150888,0.181447,0.255534,0.462591,1.059032",\
           "0.122339,0.136560,0.170784,0.249163,0.456938,1.052728");
  }
  rise_transition(DELAY_6x6) {
```

input transition

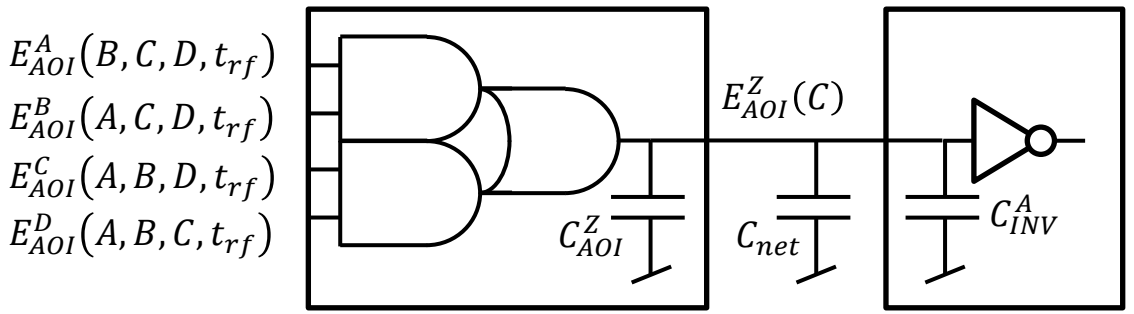
output load



Modelling Power Consumption in .lib File

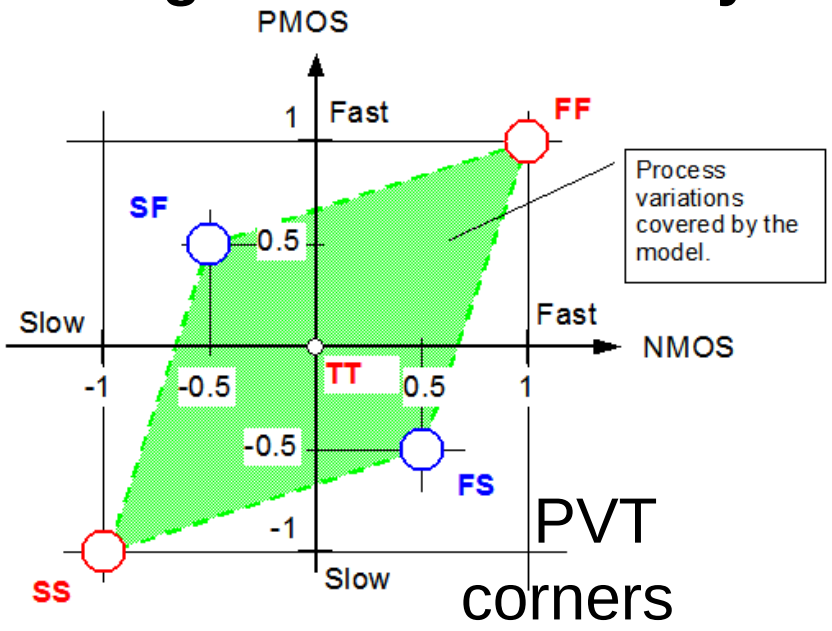
- **Power consumption is also modelled with LUTs**
- **We distinguish between Static Power and Dynamic Power**

- **Dynamic Power is further partitioned into**
  - **Internal power:** all the power consumed inside the cell boundary is independent of the context and can be tabulated in the .lib file
  - **Switching power:** depends on the context and is computed during power analysis
- **Caveat:** power consumed by switching "output capacitance" can either be part of the internal (output capacitance set to zero) or switching power (output capacitance set to its actual value)



Covering Different PVT Corners

- **Electrical (timing and power) characteristics of integrated circuits vary with**
  - Process corner
  - Temperature
  - Supply voltage

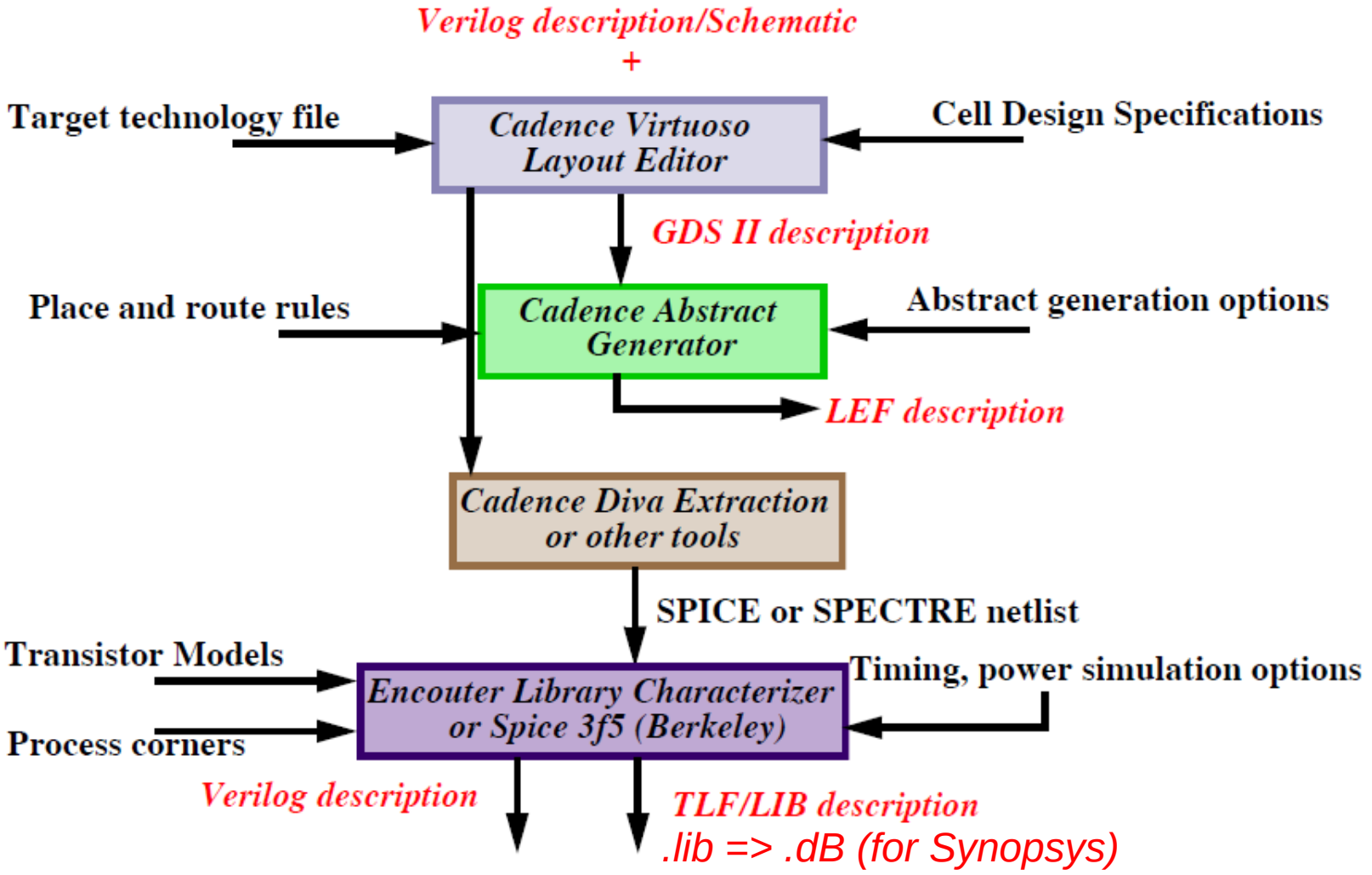


- **Circuits need to work under all conditions**
  - **Timing properties do not change equally across corners!!**
  - Difficult/impossible to predict behavior/ timing from corner model to another corner

- **Characterization needed for each cell for each relevant corner**
  - Old technologies (down to 65nm: usually 2-4 corners are sufficient (SS/FF)
  - Modern technologies <65nm: hundreds of corners

Automatic Characterization of Standard Cells

- **Standard cells are characterized with extensive simulations**
- **Tools automate this flow with generated testbenches and scripts**



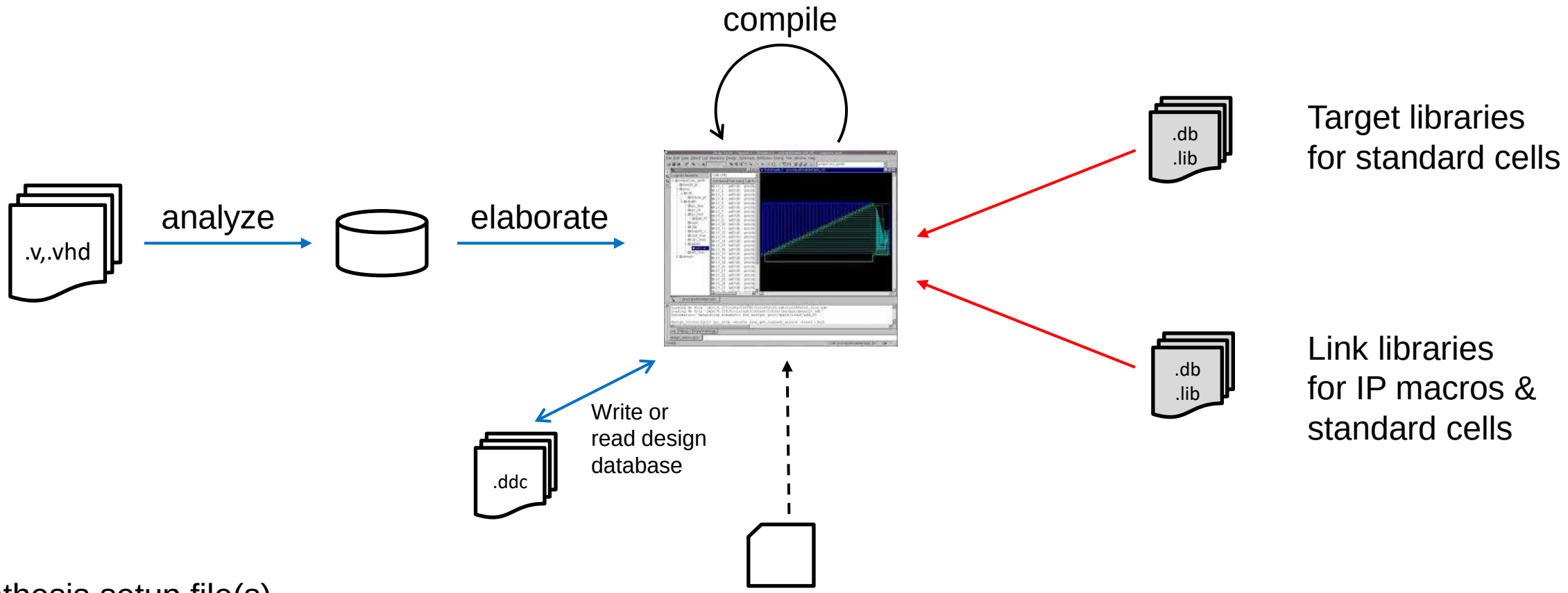
The Synthesis Environment

- **RTL synthesis runs in a dedicated sub-directory and your project**
- **In this directory, you find typically**
  - Setup/configuration files for the synthesis tool (e.g., .synopsys_dc.setup)
  - Folders for synthesis reports & outputs: RPT, TIM
  - Folders for synthesis scripts (command sequences): BIN
  - Folders for constraint files: SDC
  - Links to HDL files and library views of standard cells and IP macros.
  - Temporary design libraries

```
DesignCompiler/
├── BIN                          - - - folder for tcl scripts
├── DB                           - - - folder for design databases
├── DLIB                         - - - folder for design library temp file storage
├── edadk.conf -> ../edadk.conf  - - - configuration file used in EPFL for EDA tools
├── HDL -> ../HDL/               - - - link to VHDL/Verilog source files
├── IPS -> ../IPS/               - - - link to IPS folder
├── RPT                          - - - folder for hosting reports
├── SDC                          - - - folder for SDC files
└── TIM                          - - - folder for SDF timing files
```

The Simulation Environment & Libraries

- **RTL synthesis work with temporary libraries, an internal (in memory) database and references to abstract models of standard cells and IP macros**



Tool Setup: Reading Libraries and IPs

- **Synthesis relies on libraries for the synthesis process and designs often include other designs as IP macros (e.g., memories)**
  - Similar to the simulator (and following the HDL convention) we use libraries as a resource

- **Tool setup is handled in a start-up script:**
  - `.synopsys_dc.setup` : located in the run-directory of design compiler
  - Setup script defines (among other things):
    - **Target libraries** : libraries of standard-cells cells used for technology mapping during synthesis
    - **Link libraries** : libraries containing descriptions of leaf cells, macros and IPs instantiated in the design (e.g., memories or IO cells) – generals includes also the target libraries
    - **Synthetic libraries** : libraries of complex functions generated during synthesis (e.g., arithmetic)
    - **Design libraries** : libraries containing the design itself (i.e., the synthesis result)

Reading the HDL Design

- **HDL designs are first compiled into an internal representation**
  - Reads an HDL source file and performs HDL syntax checking and Synopsys rule checking
  - Checks file for errors without building generic logic for the design
  - Creates HDL library objects in an intermediate format, stored in the `design_library`

```
analyze -library work -format VHDL|verilog <VHDL or Verilog File Name>
```

- **Elaboration maps a design from the design_library to a technology-independent, unoptimized/un-timed logical structure (GTECH elements)**
  - Substitutes design parameters (GENERICS)
  - Bind all leaf cells to provided libraries
  - Evaluates and unrolls loops and generate statements
  - Replaces arithmetic operators with DesignWare components from the `synthetic_library`
  - Converts RTL into Boolean structure and infers registers

```
elaborate <Entity Name> -library work
```

Checking the Elaboration Report

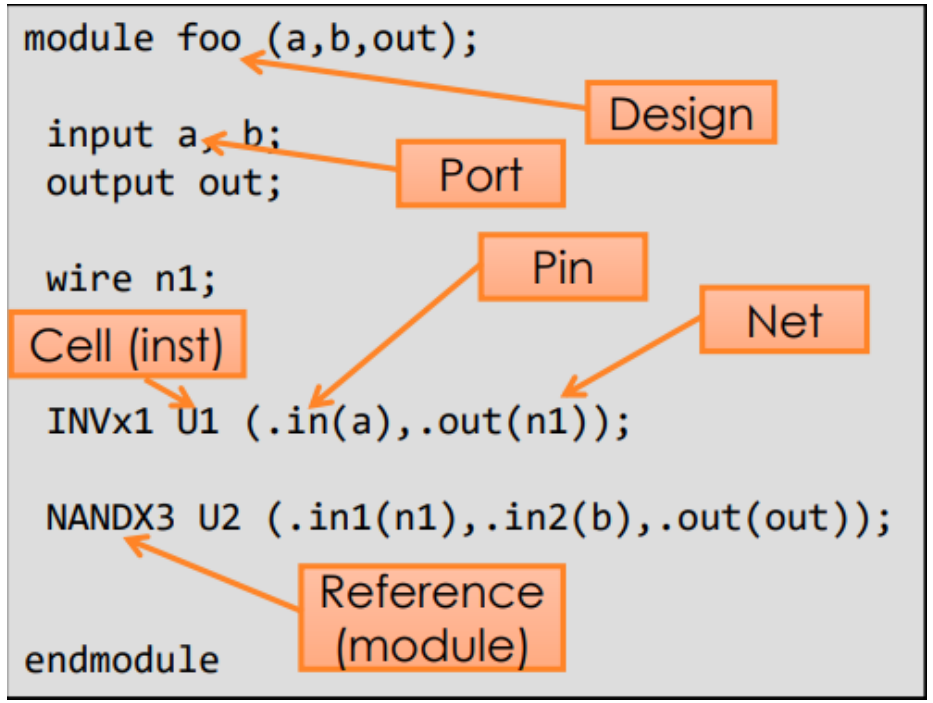
- **Elaboration generates a data base in the memory** (can be saved as .ddc file)
- **Elaboration reports many interesting points:**
  - Provides details on design hierarchy
  - Reports missing instances for which no HDL was available and which could not be found in
    - the database of previously analyzed HDL files
    - in the memory database of the tool (not loaded)
    - in any of the link libraries
  - Provides details on the inferred registers (and possibly inferred latches)

Example from EDA Labs

```
Inferred memory devices in process
        in routine mult_32b line 45 in file
                './HDL/RTL/mult_32b.vhd'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     reg_op2_reg     | Flip-flop |  32   |  Y  | N  | Y  | N  | N  | N  | N  |
|     reg_op1_reg     | Flip-flop |  32   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
```

Design Objects

- **Designs are comprised of several objects that carry particular names**
  - **Design**: A circuit description that performs one or more logical functions (i.e Verilog module)
  - **Cell**: An instantiation of a design within another design (i.e. VHDL/Verilog instance)
  - **Reference**: The original design that a cell "points to" (i.e. VHDL/Verilog sub-module)
  - **Port**: The input, output or inout port of a Design
  - **Pin**: The input, output or inout pin of a Cell in the Design
  - **Net**: The wire that connects Ports to Pins and/or Pins to each other.
  - **Clock**: Port of a Design or Pin of a Cell explicitly defined as a clock source

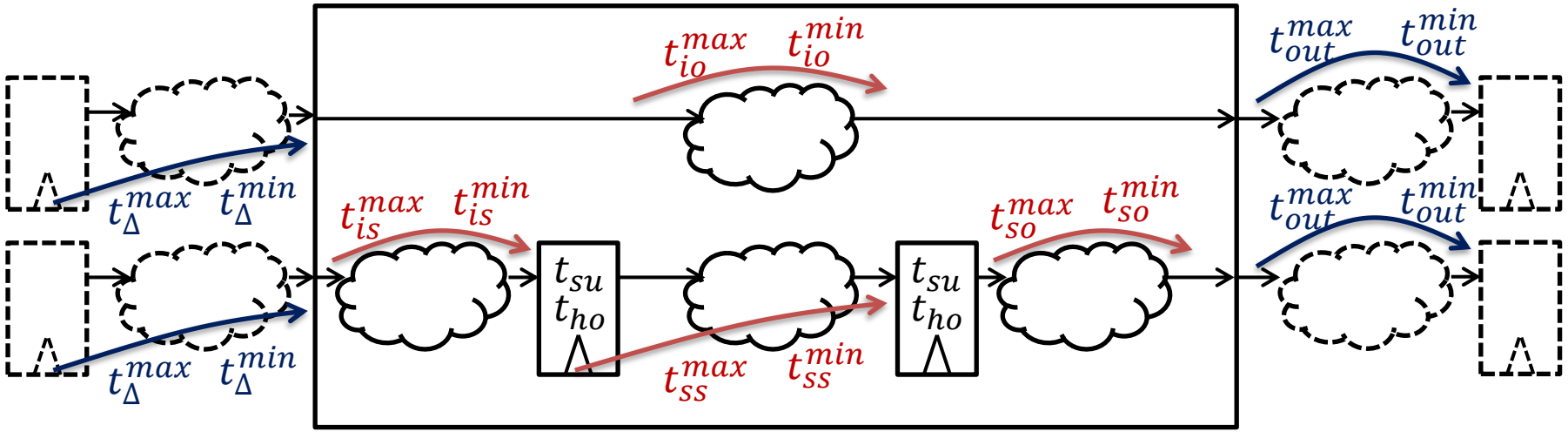


Static Timing Analysis (STA)

- **STA drives the synthesis process and checks after synthesis if constraints are met**

- **Reminder: STA checks timing constraints for four path groups:**
  - Register-to-register paths
  - Input-to-register paths
  - Register-to-output paths
  - Input-to-output paths

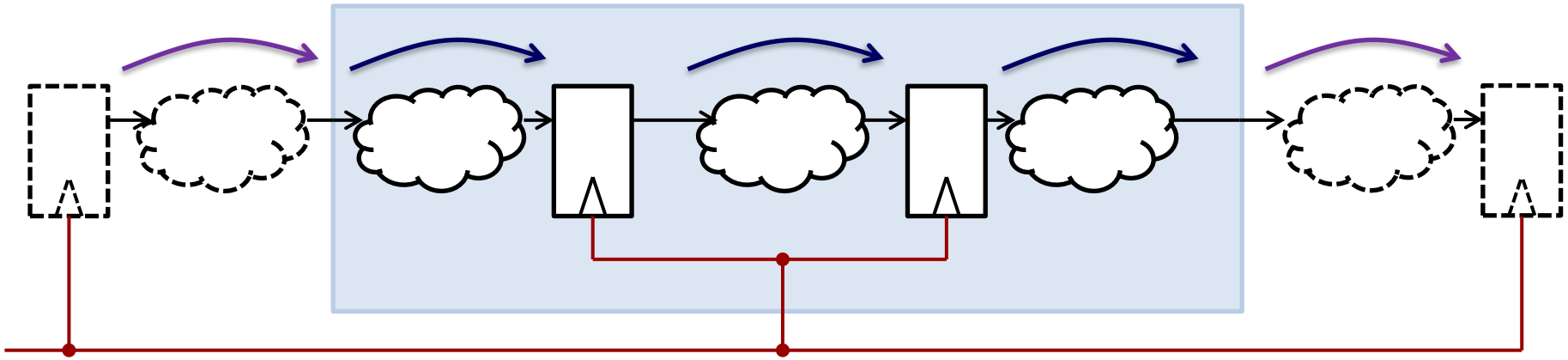
$$t_{ss}^{max} < T_{CLK} - t_{su} \qquad t_{ss}^{min} > t_{ho}$$
$$t_{\Delta}^{max} + t_{is}^{max} < T_{CLK} - t_{su} \qquad t_{\Delta}^{min} + t_{is}^{min} > t_{ho}$$
$$t_{ss}^{max} + t_{out}^{max} < T_{CLK} \qquad t_{ss}^{min} + t_{out}^{min} > 0$$
$$t_{\Delta}^{max} + t_{io}^{max} + t_{out}^{max} < T_{CLK} \qquad t_{\Delta}^{min} + t_{io}^{min} + t_{out}^{min} > 0$$



Specifying Timing Constraints

- **Synthesis is heavily driven by timing**
  - Need to define expectations to enable timing analysis

- **We usually define our expectations (optimization objectives) in three ways**
  - **Clock definitions** : define the clock signals and periods
  - **Modelling the world external to the chip** : define delays of connected designs
  - **Explicit timing requirements :** define explicit delay expectations
  - **Timing exceptions** : define unusual timing conditions (USE ONLY WITH GREAT CARE)



Specifying Clocks

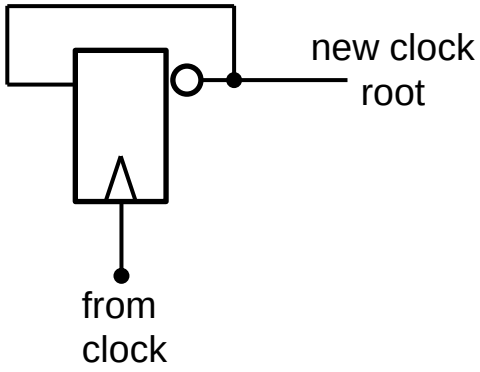
- **Clocks are the basis for different constraints**
- **Clocks are defined by**
  - Clock period : basic timing requirement
  - Clock name : used later to relate other timings to a clock)
  - Pin to which it is attached : to identify affected sequential elements

```
create_clock -period <period> -name <clock name> [get_ports <clock root port>]
```

- **Some designs can have multiple clocks** (use with care)
  - Clocks can be derived from a mother clock with a ratio

```
create_generated_clock -name <clock name> -source [get_ports <from clock>]
-divide_by <factor> [get_pins <clock root pin|port>]
```

  - Clocks can be fully independent clocks



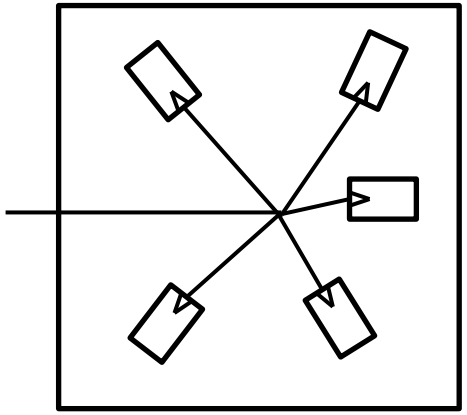
Anticipating Non-Ideal Clocks

- **Clock signals must be distributed to the connected sequential elements**
  - During synthesis, high quality clock distribution is not possible (no knowledge on layout)
  - Clock-tree insertion (clock distribution) is deferred to the layout stage

```
set_dont_touch <port|net>
```

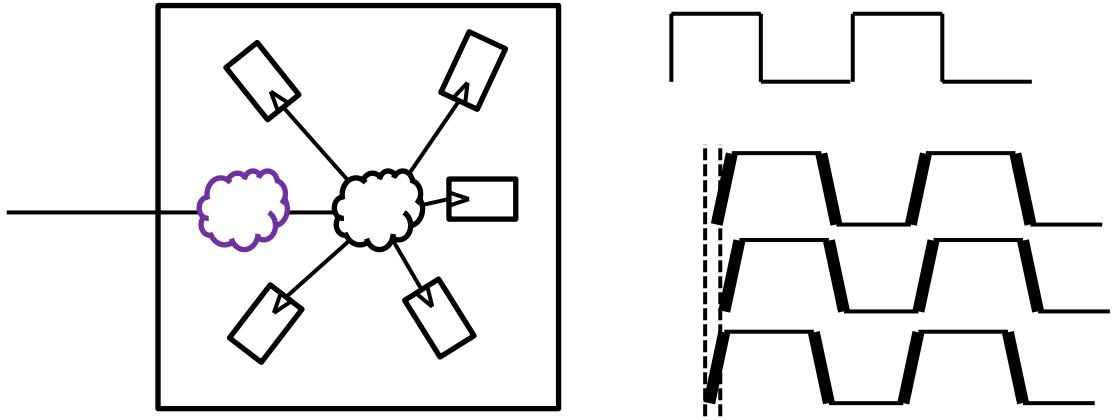
- **Two common approaches to model clocks during synthesis**

Assume ideal clocks



```
set_ideal_network <port|net>
```

Anticipate clock non-idealities

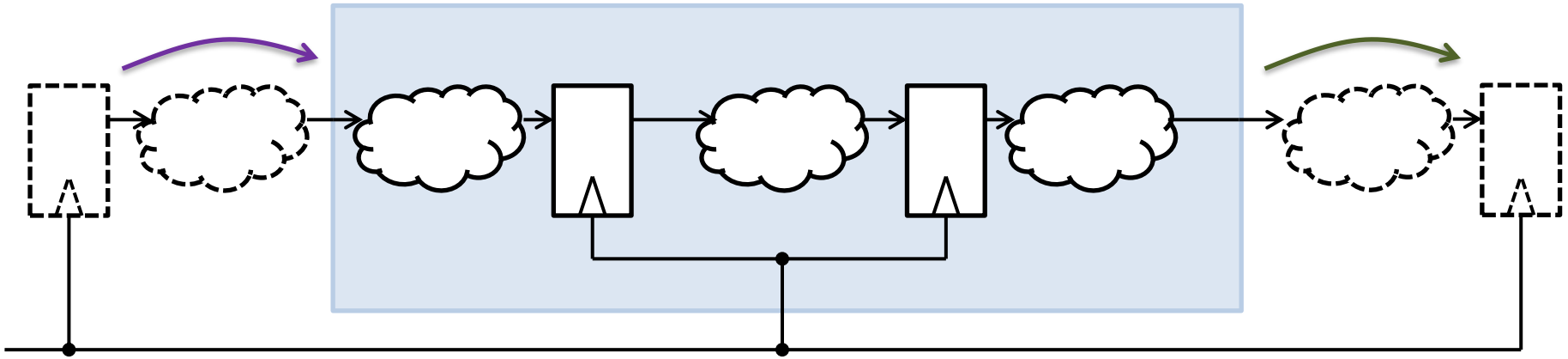


```
set_clock_transition <value> <clock>
set_clock_latency <value> <clock>
set_clock_uncertainty <value> <clock>
set_clock_jitter <value> <clock>
```

Specifying Input/Output Timing Conditions

- **Constraints for input- and output-logic are defined by surrounding blocks**
  - Input-to-register constraint: depend on output delay of previous block
  - Register-to-output constraint: depend on input delay of next block

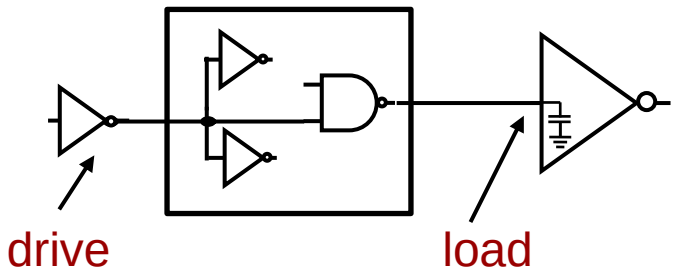
```
set_input_delay <value> -clock <clock> [remove_from_collection [all_inputs] [get_ports <clock port>]]
set_output_delay <value> -clock <clock> [all_outputs]
```



  - Delays can be defined for each input/output individually or for multiples (see example)
  - Input and output delays must be defined relative to a clock
  - Pay attention not define input delays for clock ports

Specifying Electrical Boundary Conditions

- **Any circuit or sub-circuit is always connected to other circuits**
  - Electrical characteristics of the interface influence the timing and of the circuit under considerations



- **Need to define boundary conditions**
  - Load on the outputs that must be driven to provide sufficient drive strength for the outputs

```
set_load <load_capacitance> [all_outputs]
set_load [load_of lib/cell/pin] [all_outputs]
```

  - Drive strength of the inputs to avoid overly large fanout of the inputs

```
set_driving_cell -lib_cell <cell name> -pin <pin_name> [remove_from_collection [all_inputs] [get_ports <clock>]]

set_drive <driver resistance> <port>                    # NOT RECOMMENDED (INACCURATE with NLDM)
set_input_transition <transition time> <port>
```

Synthesis and Optimization

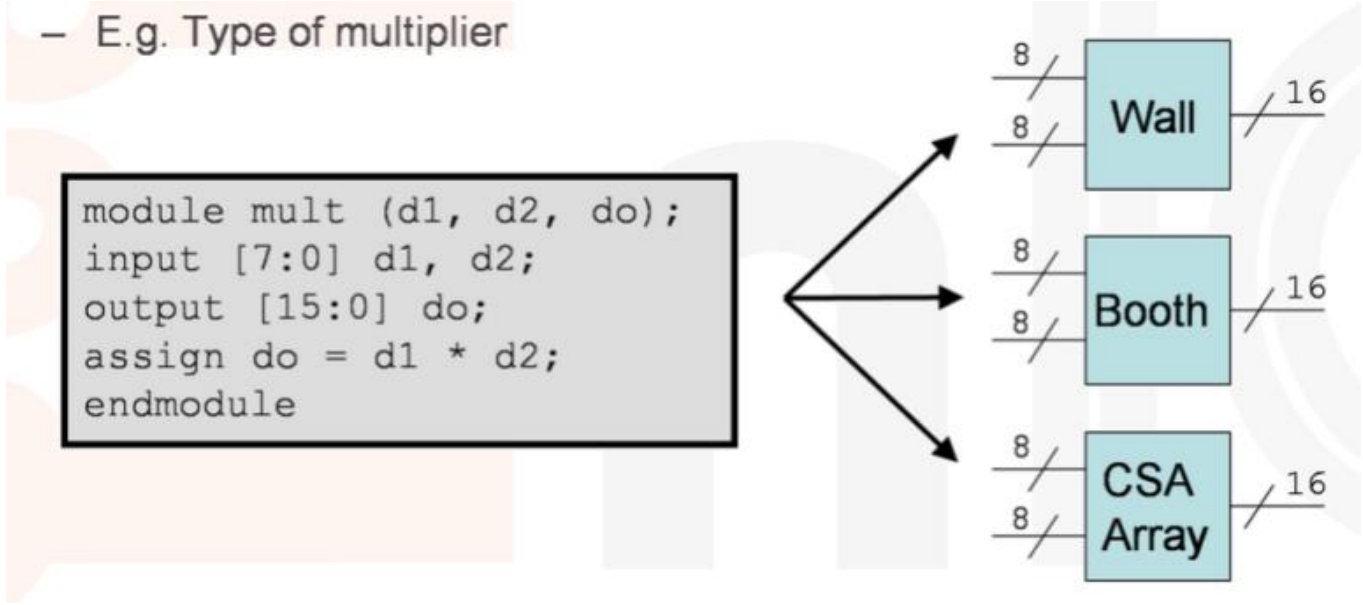
- **Once the design is elaborated and constraints are defined, we can start the actual synthesis & optimization**

- **Design compiler integrates the entire synthesis into one command with many options to control the optimization**

  - `compile` vs. `compile_ultra` : different optimizations turned on, depending on your license
  - `- exact_map` : skip many optimizations to obtain a mostly exact RTL mapping
  - `- incremental` : re-run the logic optimization on an already compiled design to improve further
  - `- map_effort [low|medium|high]` : tradeoff between run-time and quality
  - `- retime` : allow moving registers to balance pipelines (works only when registers are already in a good place (only small moves allowed)
  - `- boundary_optimization` : allow optimization across hierarchy boundaries

Synthesis of Datapath Operations

- **Complex datapath operations are implemented in a special way**
  - Initially, they are not part of the logic synthesis (no immediate translation into logic)
  - Many people have spent their time on optimal arithmetic circuits
- **Special datapath libraries/compilers are used to generate arithmetic ops.**



- **Synopsys and cadence refer to these as DesignWare and ChipWare IP**

Reporting

- **We are interested in various design metrics which can be reported:**

report_area

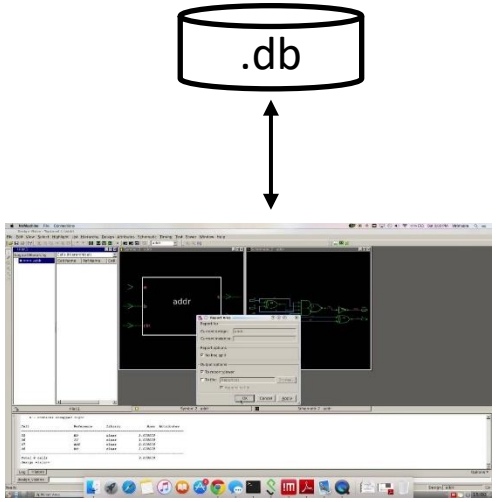
```
****************************************
Report : area
Design : xtal_chip_with_io
Version: I-2013.12-SP5-2
Date   : Wed Aug  5 10:57:32 2015
****************************************

Library(s) Used:

    c5n_utah_std_v5_t27 (File: /home/synopsys/ON.PDK/sample/logic
    io (File: /home/synopsys/ON.PDK/sample/logic/io.db)

Number of ports:                          8
Number of nets:                         187
Number of cells:                        185
Number of combinational cells:          137
Number of sequential cells:              40
Number of macros/black boxes:             8
Number of buf/inv:                       21
Number of references:                    11

Combinational area:          42120.000000
Buf/Inv area:                 4536.000000
Noncombinational area:       54720.000000
Macro/Black Box area:            0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:             96840.000000
Total area:                 undefined
```

report_timing

```
Startpoint: I_xtal_chip/count1_reg[0]
            (rising edge-triggered flip-flop clocked by main_clock)
Endpoint: I_xtal_chip/count1_reg[19]
          (rising edge-triggered flip-flop clocked by main_clock)
Path Group: main_clock
Path Type: max

Point                                                   Incr       Path
-----------------------------------------------------------------------------
clock main_clock (rise edge)                            0.00       0.00
clock network delay (ideal)                             0.00       0.00
I_xtal_chip/count1_reg[0]/CLK (DCX1)                    0.00       0.00 r
I_xtal_chip/count1_reg[0]/Q (DCX1)                      0.80       0.80 f
U245/Y (NAND2X1)                                        0.24       1.04 r
U246/Y (INVX2)                                          0.18       1.23 f
U237/Y (NAND2X1)                                        0.30       1.53 r
U235/Y (NOR2X1)                                         0.42       1.95 f
...

U241/Y (OAI21X1)                                        0.25       8.06 r
U242/Y (AOI21X1)                                        0.28       8.34 f
I_xtal_chip/count1_reg[19]/D (DCX1)                     0.00       8.34 f
data arrival time                                                  8.34

clock main_clock (rise edge)                           20.00      20.00
clock network delay (ideal)                             0.00      20.00
I_xtal_chip/count1_reg[19]/CLK (DCX1)                   0.00      20.00 r
library setup time                                     -0.39      19.61
data required time                                                19.61
-----------------------------------------------------------------------------
data required time                                                19.61
data arrival time                                                 -8.34
-----------------------------------------------------------------------------
slack (MET)                                                       11.27
```

Exporting Design Data

- **Design database and further design information can be saved in different formats to restore later or to proceed to other steps in the design flow**



Timing constraints for layout tool

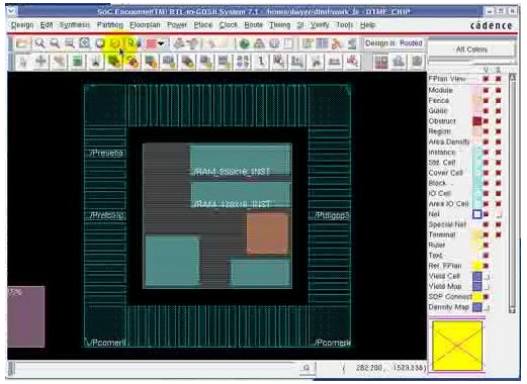
.sdc

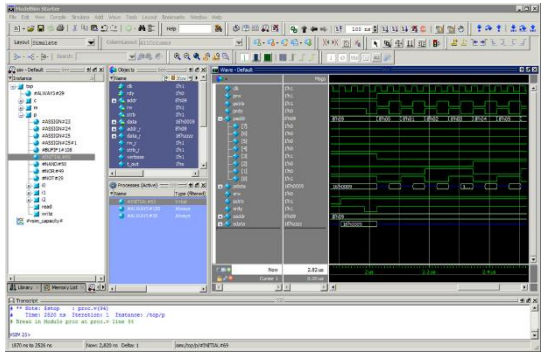
NETLIST

.v

Delay Information from timing analysis

.sdf





Exporting the Design Data

- **Writing out a complete design database to be used in design compiler**

- **Exporting a Verilog netlist**
  - Structural Verilog is the de-facto standard for digital netlists
  - Change names in the design to meet Verilog conventions

```
change_names -rules verilog -hierarchy -verbose
write -hierarchy -format verilog –output <file name>
```

Exporting the Design Data

- **Writing out a complete design database to be used in design compiler**

- **Exporting timing information for simulations**
  - The SDF file contains timing information for all timing arcs
  - This information is used by the Verilog models of standard cells and IPs to behave accordingly in the gate-level simulation

```
write_sdf -version 2.1 <file name>
```

```
(DELAYFILE
(SDFVERSION "OVI 2.1")
(DESIGN "alu32top")
(DATE "Wed May  9 11:47:29 2018")
(VENDOR "fsd0a_a_generic_core_ss0p9v125c")
(PROGRAM "Synopsys Design Compiler cmos")
(VERSION "J-2014.09-SP5")
(DIVIDER /)
(VOLTAGE 1.10:0.90:0.90)
(PROCESS "BCCOM:WCCOM:WCCOM")
(TEMPERATURE -40.00:125.00:125.00)
(TIMESCALE 1ns)
...
(CELL
  (CELLTYPE "INVX2")
  (INSTANCE U3)
  (DELAY
    (ABSOLUTE
    (IOPATH I O (0.229:0.483:0.483)
(0.118:0.268:0.268))
    )
  )
)
...
```

Post Synthesis Gate Level Simulations

- **Simulate the design (netlist) after synthesis to**
  - **Verify correct functionality**: in most cases unnecessary as synthesis rarely makes mistakes when RTL code is written “properly” according to the rules of synchronous design
  - **Obtain** initial **indication of toggle activity for** initial **power analysis**

- **Post-synthesis simulation results should be taken with lots of care:**
  - Timing is far from accurate as layout parasitics are not yet included
  - No clock or reset tree included in the design (ideal clock)
  - Backend will still heavily modify the design to account for parasitics (buffer insertion, re-synthesis of critical path, addition of a clock tree, …)

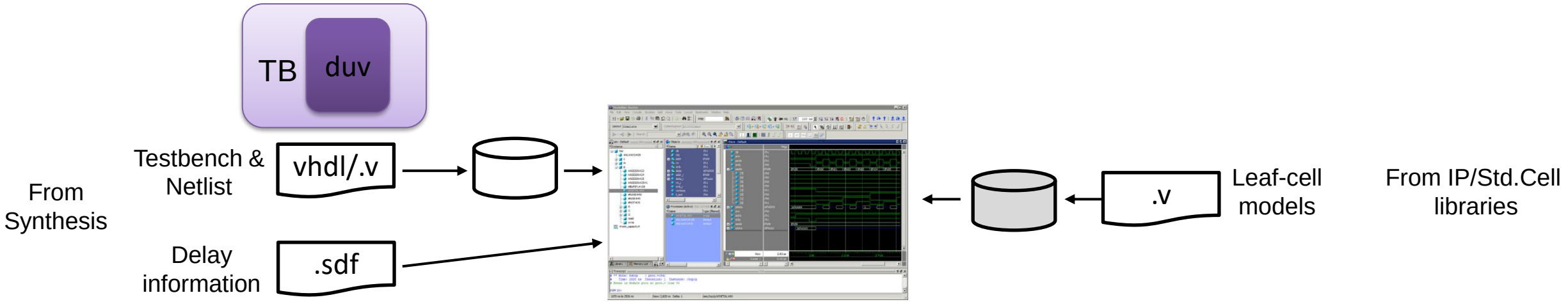
Post Synthesis Gate Level Simulations

- **For a Gate level simulation, the following information is requires**
  - The netlist of the synthesized design: provided by the synthesis tool
  - The delay information for each gate (per instance) in the design: provided by the synthesis tool

  - **Models for the instantiated leaf-cells** (e.g., standard-cell or memories): provided as part of the IP (standard-cell libraries and IP macros)
    - Beahvioral model for these cells in VHDL or Verilog
    - Models must also model the timing/delay and check potential constraints (e.g., setup/hold times)
    - Timing parameters (delays/constraints) must connect to the timing provided by the SDF => BACKANNOTATION

Post Synthesis Gate Level Simulations

- **When calling the simulator, we must provide all this information and "link" the timing to the relevant part of the design**



Compiled library of Verilog models for the standard cells

Timing information for every instance of the design entity "duv"

```
> vsim
        -L .../MGC_QSIM/DLIB/umc_65nm_ll_uk65lscllmvbbr_sdf21_vlog
        -sdftyp duv=.../SNPS_DC/TIM/alu32top_rtl_clk3ns_mapped_vlog.sdf
        ALU32-MAPPED.alu32top_tb_conf_mapped
        -t ps -voptargs=+acc
```