Digital System Design

Synopsys Synthesis Overview

Speaker: 王景平

Adviser: Prof. An-Yeu (Andy) Wu

Date: 2025.04.10

generate for vs. for

```
integer i;
for (i = 0; i < 8; i = i+1) begin
  out[(i+1)*8 : i*8] = in[(i+1)*8 : i*8];
end
```

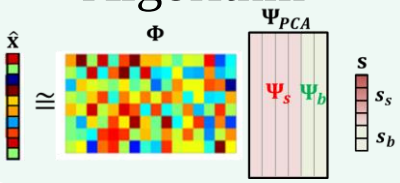
→ `out[i*8 +: 8]`

```
genvar i;
generate
  for (i = 0; i < 8; i = i+1) begin
    out[(i+1)*8 : i*8] = in[(i+1)*8 : i*8];
  end
endgenerate
```

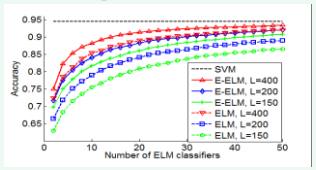
Cell-Based IC Design Flow

Algorithm

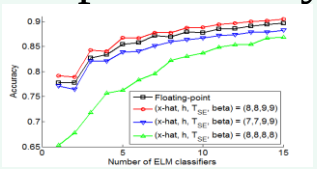
Algorithm



Floating-point Analysis



Fixed-point Analysis

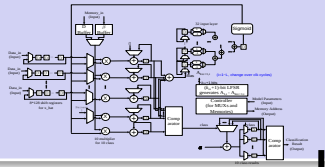


RTL

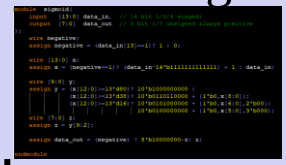
Spec

Parameter	Range	Descript ion
Dimension of input data (D_x)	128, 256, 512, 1024	MNIST Databas e
Number of hidden neurons (h_n)	1-256	

Architecture

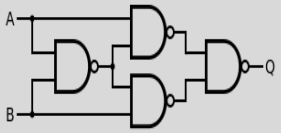


Verilog

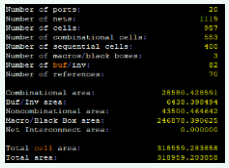


RTL Simulation

Synthesis



Gate-level Netlist

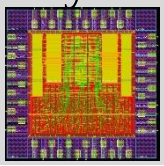


Gate-level Simulation

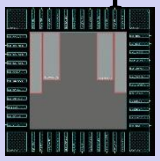
Power Optimization

Not cover

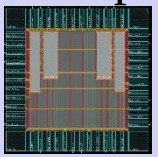
Layout



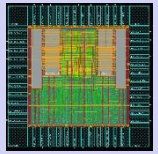
Floorplan



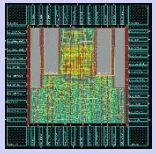
Powerplan



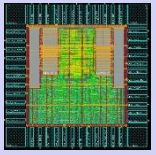
Placement



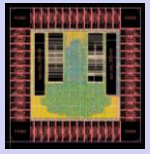
CTS



Route

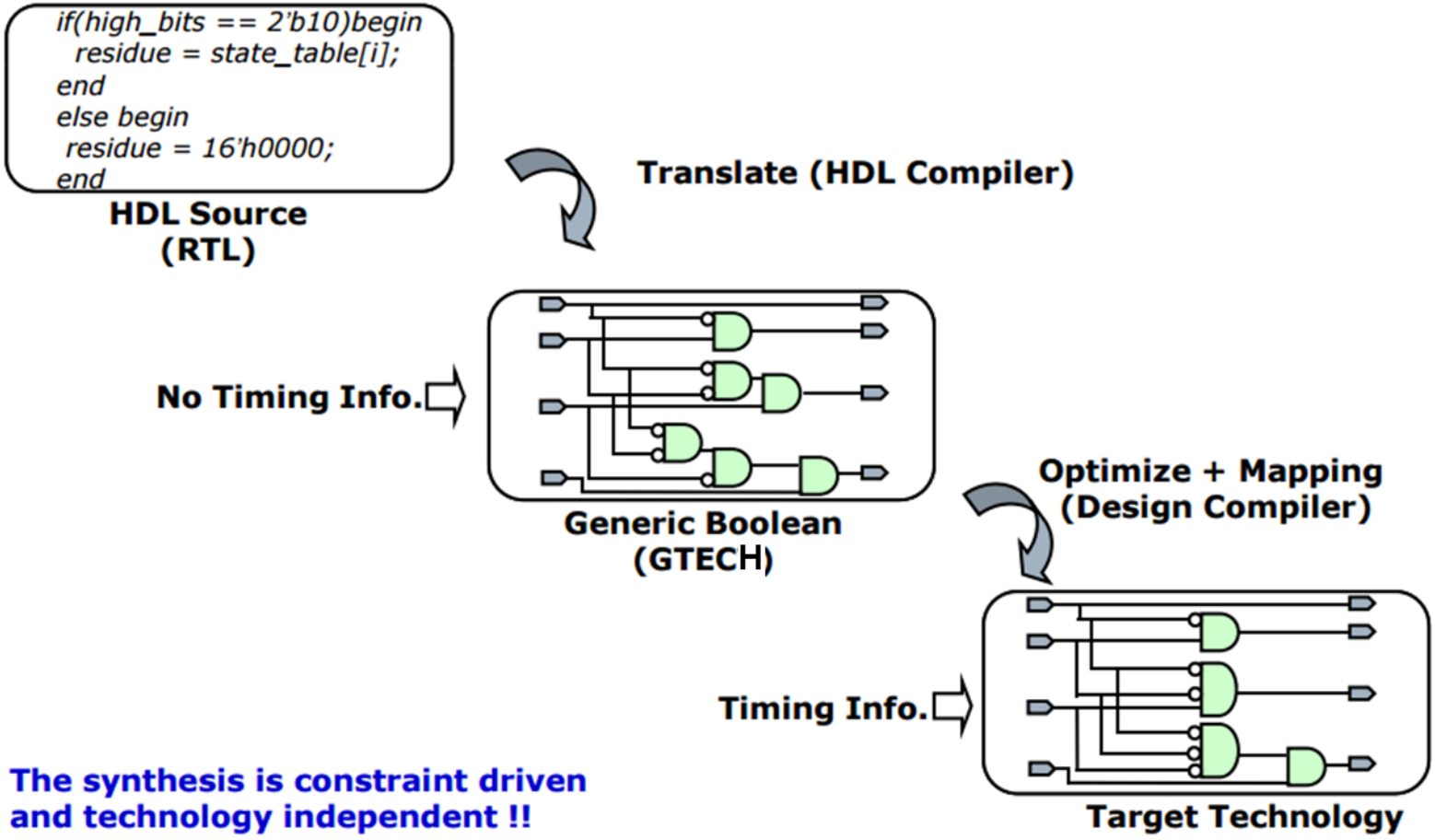


DRC/LVS



What is Synthesis

- Synthesis = translation + optimization + mapping



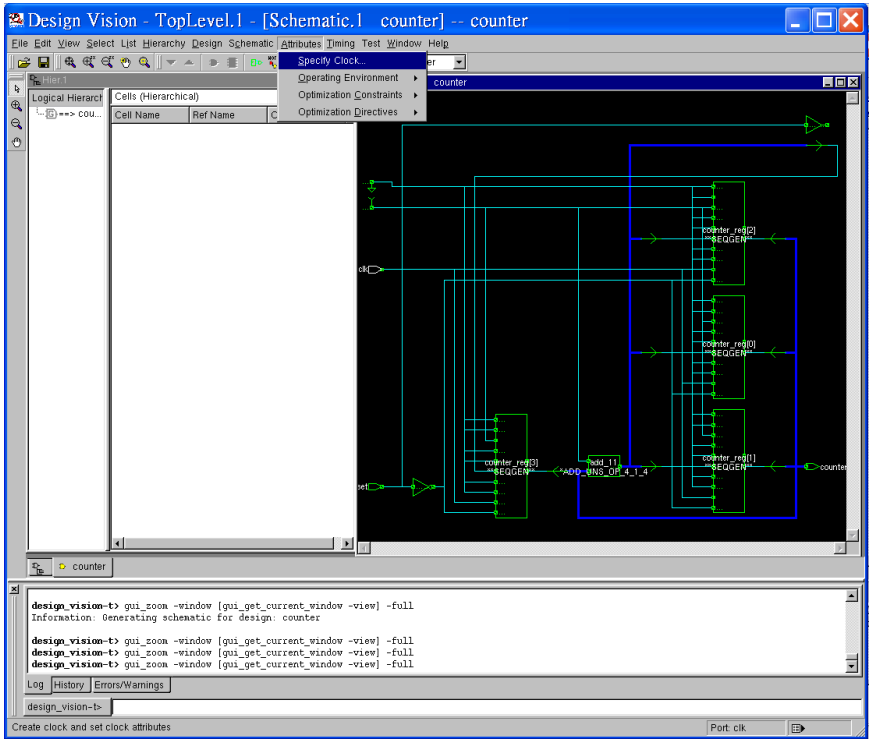
HDL Compiler Translation

```
always @(reset or set) begin
  if (reset)
    y=1'b0;
  else if (set)
    y=1'b1;
end

always @(gate or reset) begin
  if (reset)
    t=1'b0;
  else if (gate)
    t=d;
end
```

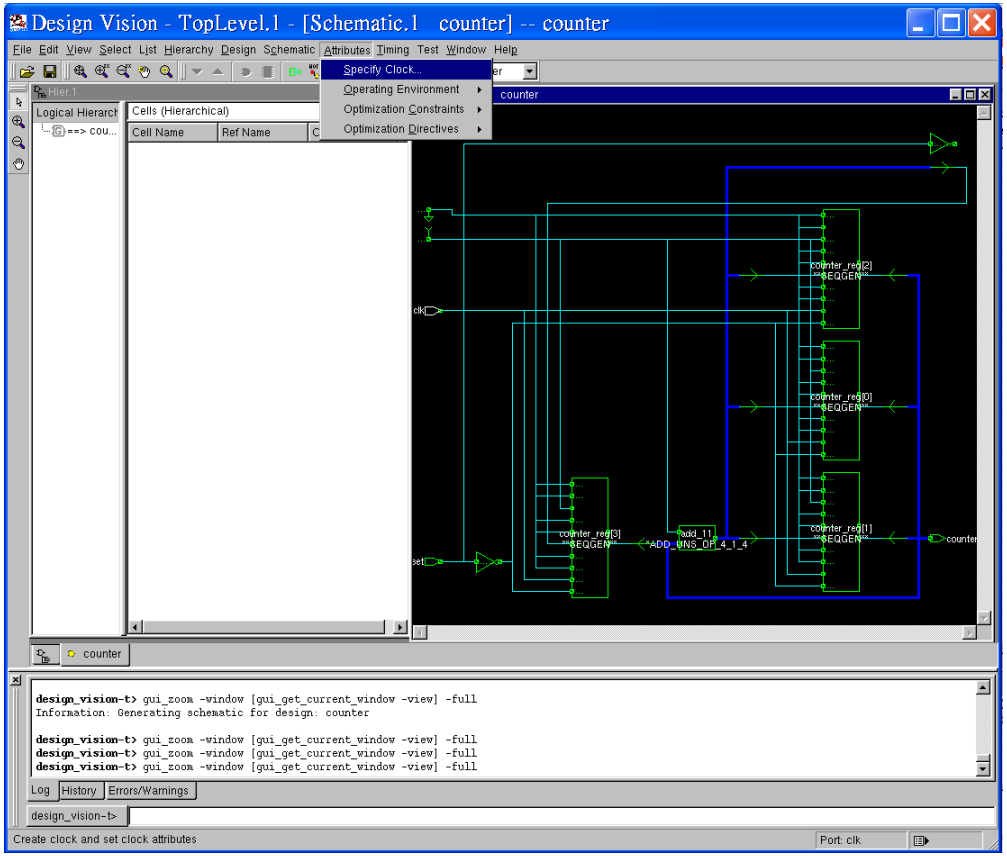
HDL Compiler translates Verilog HDL descriptions into Design Compiler as Synopsys *design block*

HDL Compiler



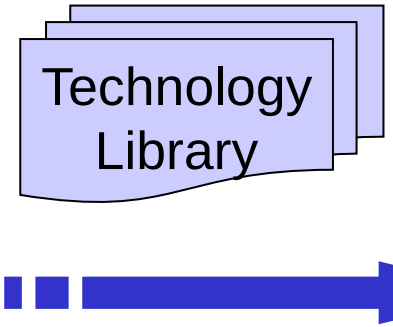
HDL Compiler Translation

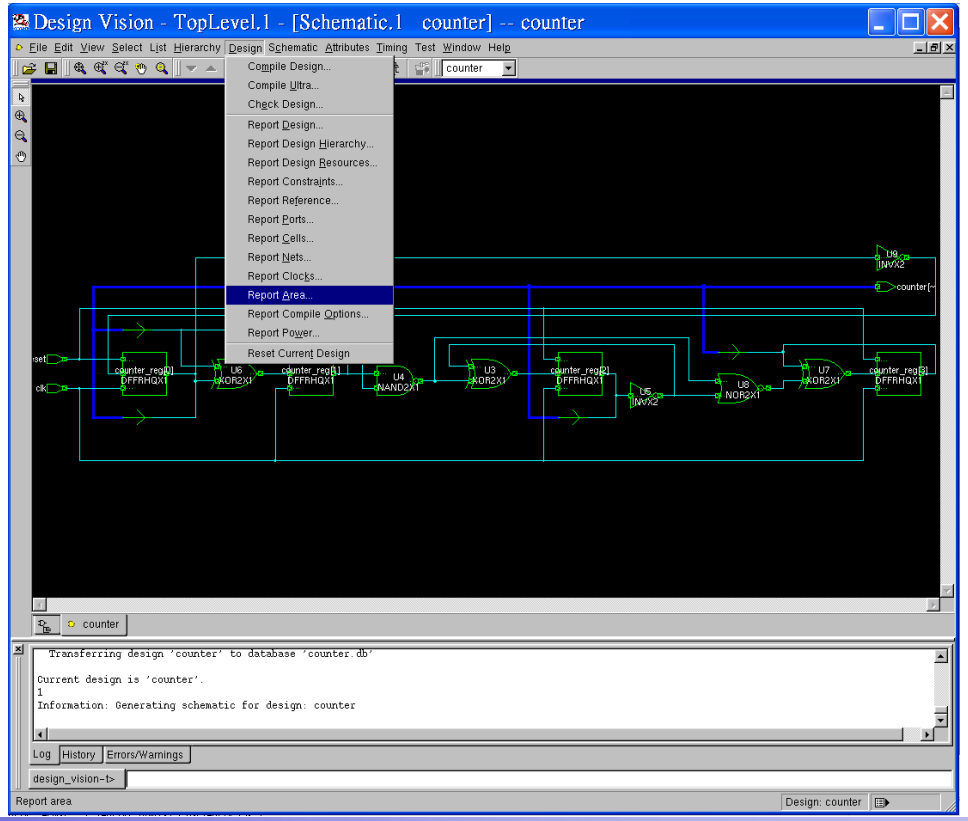
- In schematic view, we can see the Verilog file is translated with a *GTECH* library (the synopsys default)



Design Compiler

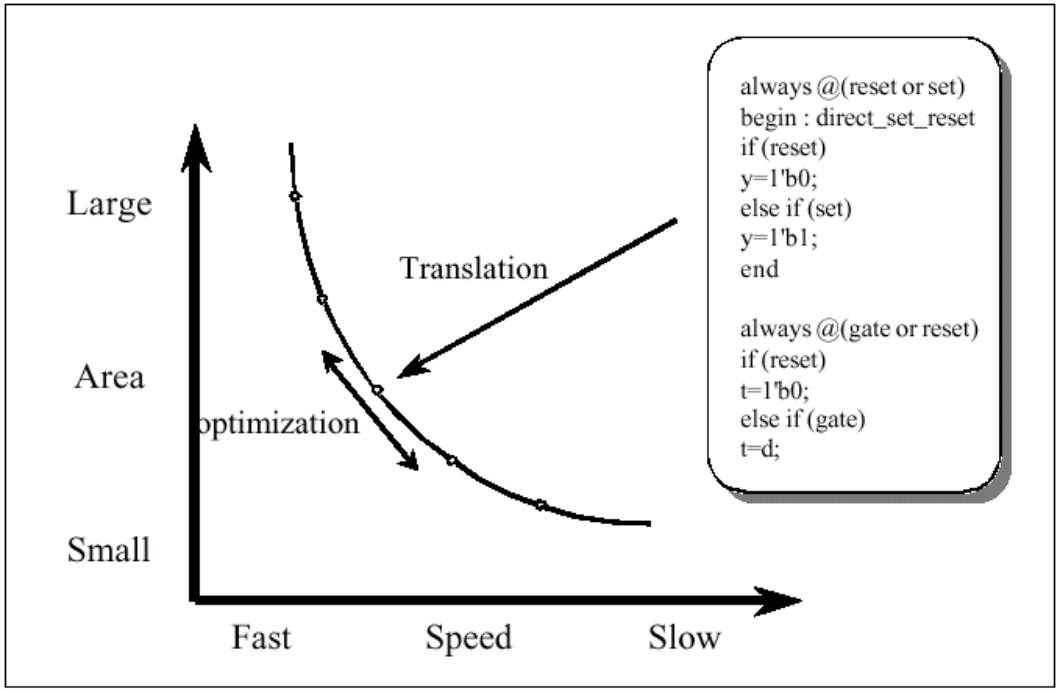
- Design Compiler maps Synopsys design block to gate level design with a user specified library



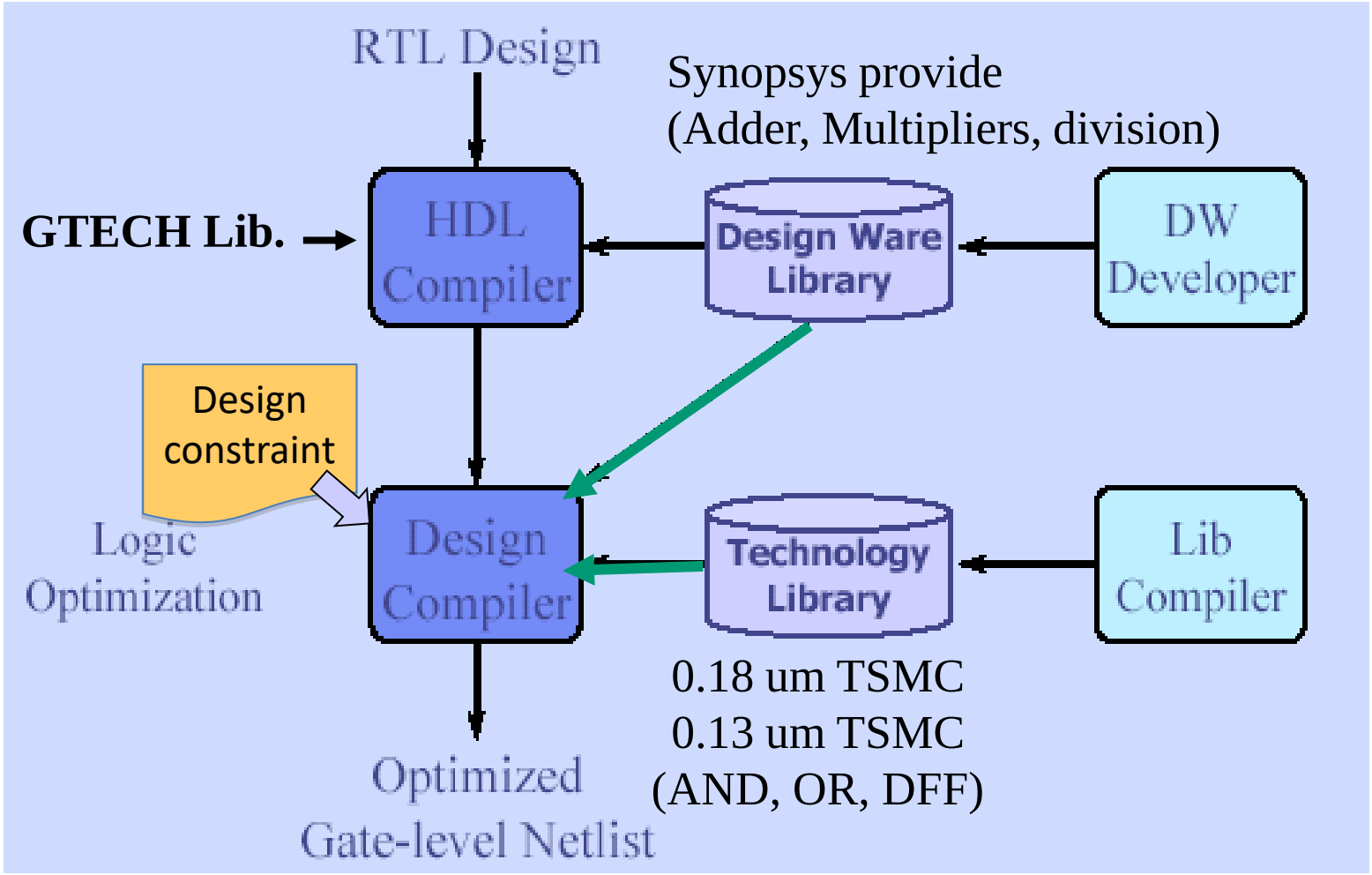


Trade-off between Speed and Area

- Synthesis is Constraint Driven
- Technology Independent



Logic Synthesis Overview



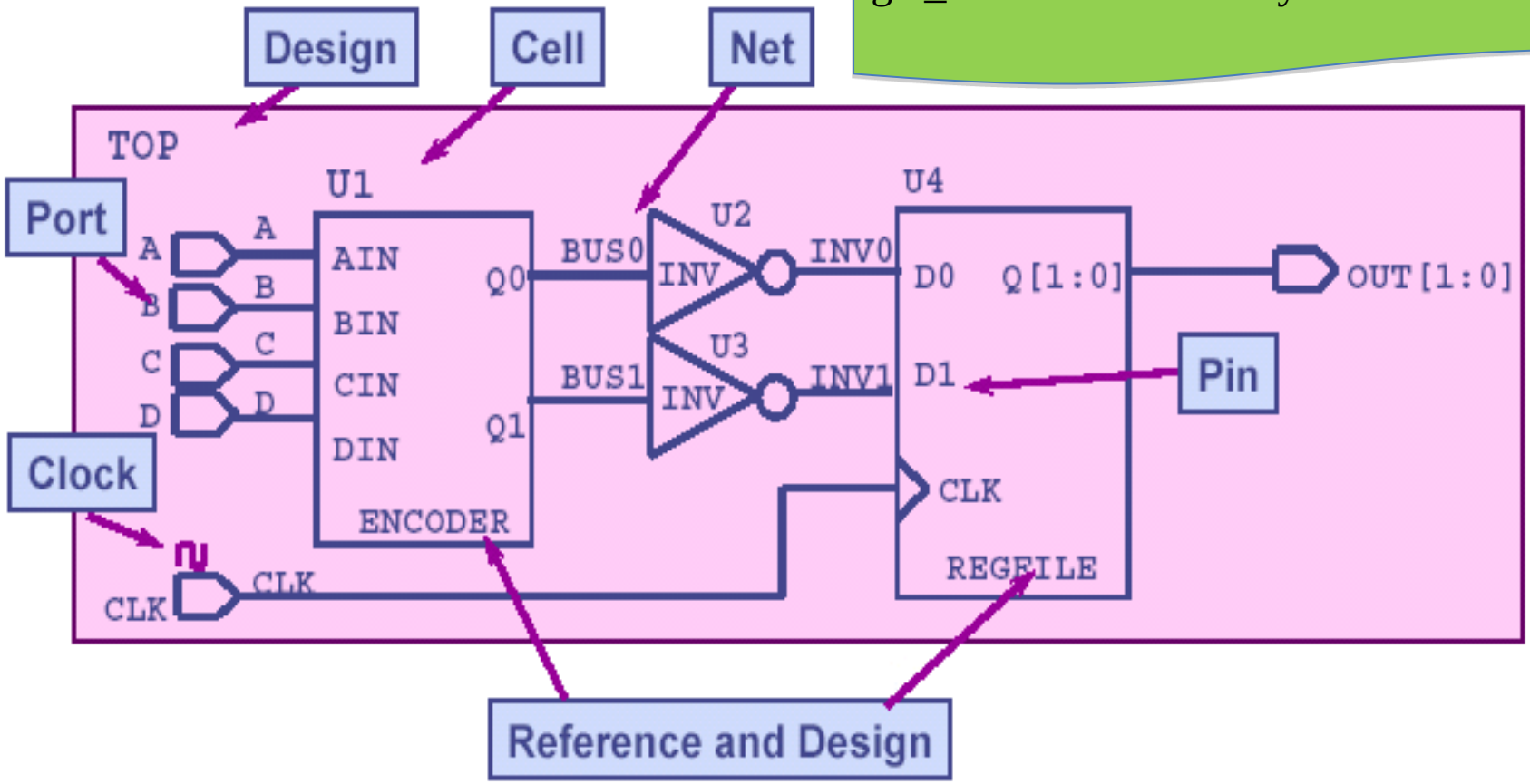
Design Objects

- Seven Types of Design Objects:

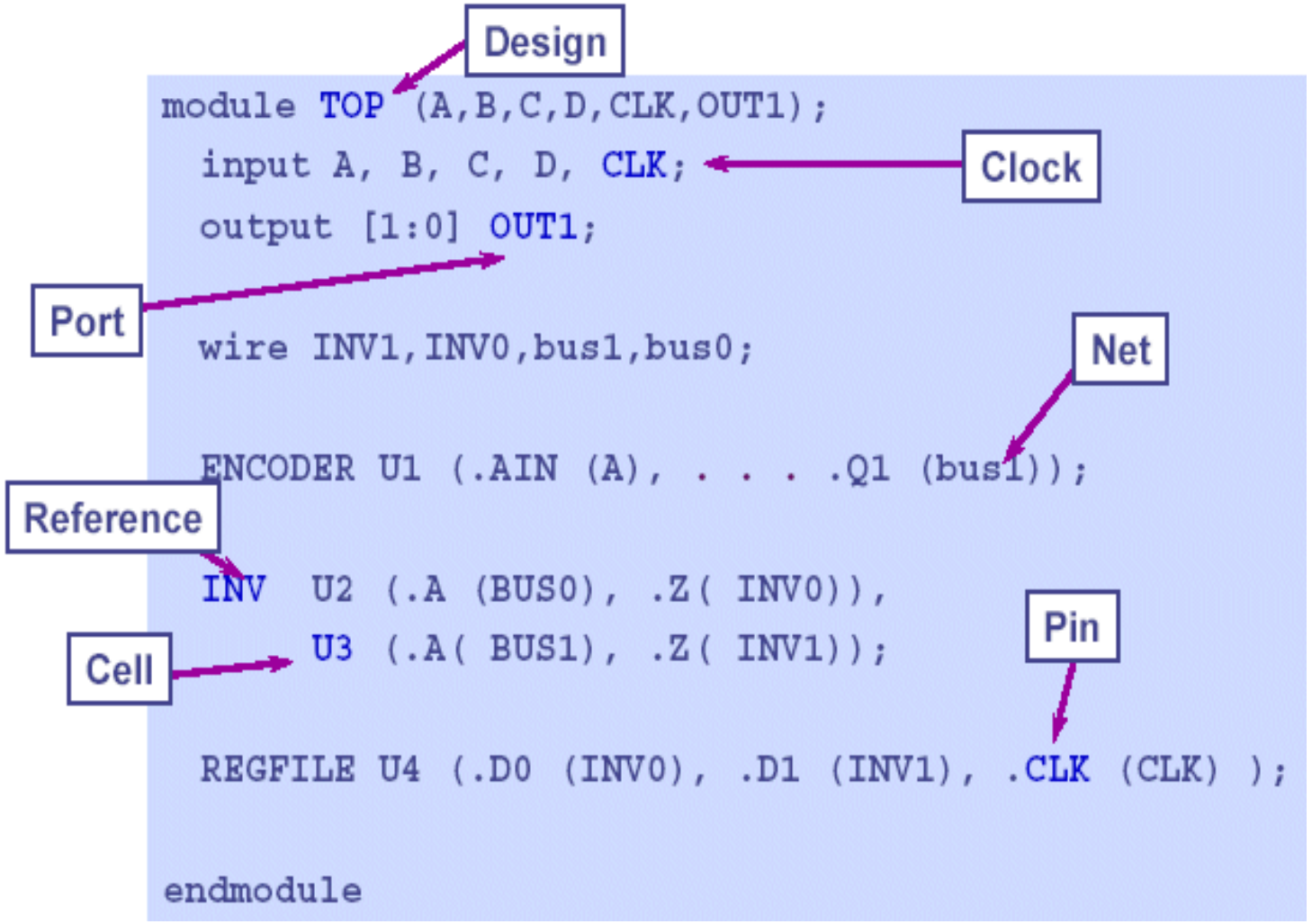
- **Design**: A circuit that performs one or more logical functions (top module)
- **Cell**: An instance of a design or library primitive within a design (instance)
- **Reference**: The name of the original design that a cell instance "points to"
- **Port**: The input or output of **a design**
- **Pin**: The input or output of **a cell**
- **Net**: The wire that connects ports to pins and/or pins to each other
- **Clock**: A timing reference object in DC memory which describes a waveform for timing analysis

Design Objects (Schematic Perspective)

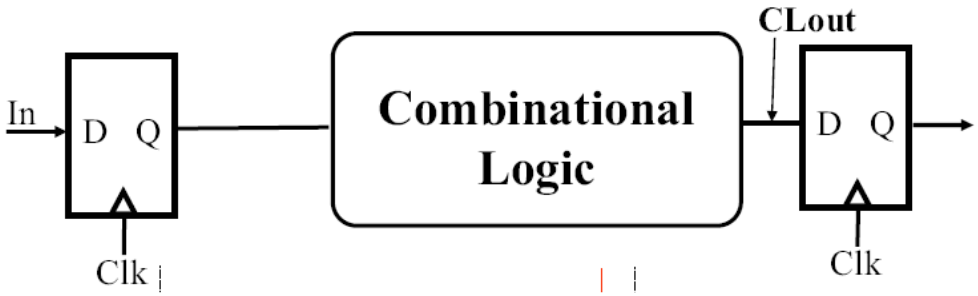
Know this and know how to correctly use get_ command in Tcl syntax!



Design Objects (Verilog Perspective)



Synthesis Target



- Only consider gate delay and approximate wire delay
- Clock network is deeply related to physical layout, thus not considered
  - Induce clock latency/uncertainty to estimate
- Setup time must be met, hold time is optional
  - Hold time violation can be resolved by inserting buffers during APR

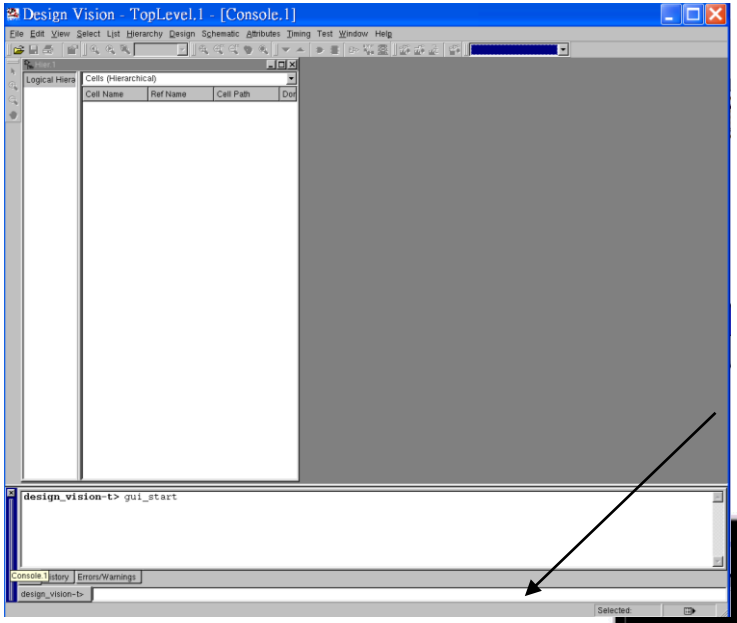
Outline

- Introduction to Synthesis Tools
- File Preparation
- Synthesis Flow
- Design Environment
- Design Constraints
- Gate-Level Simulation
- Synopsys Filters

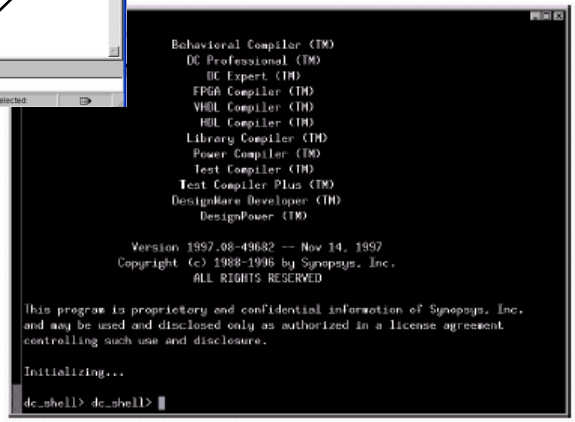
Tools We will Use

Tool	Purpose
Design Compiler	Constraint driven logic optimizer for synthesis
Design Vision	Graphical User Interface (GUI) of synopsys synthesis tool
Presto HDL Compiler	Translate Verilog descriptions into Design Compiler
Design Time	Static Timing Analysis (STA) engine
DFT Compiler	Design for Testing
Design Ware	Enable synthesis using DesignWare library

Design Compiler Interaction

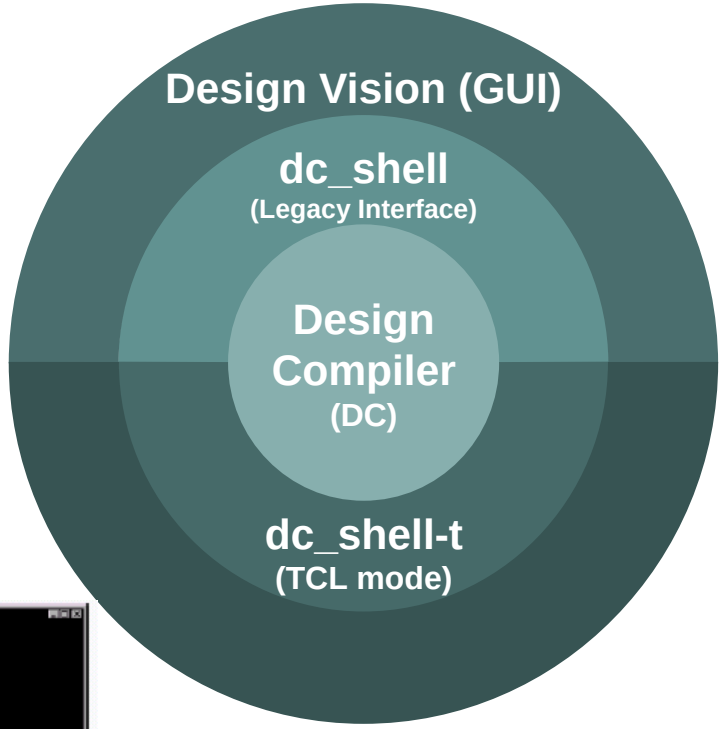


Command line



Unix%> **dc_shell**

Unix%> **dc_shell –gui**



Outline

- Introduction to Synthesis Tools
- File Preparation
- Synthesis Flow
- Design Environment
- Design Constraints
- Gate-Level Simulation
- Synopsys Filters

File Preparation

- .synopsys_dc.setup
- Design file
  - Verilog (.v)
- Technology library file
  - LIB (.lib)
  - DB (.db)
- Script file
  - TCL (.tcl)
- Design constraint file
  - Synopsys design constraints (.sdc)

.synopsys_dc.setup

```
set company {NTUEE}
set designer {Student}

set search_path [concat  [list .
    /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db .]
    $search_path]
set link_library [list "dw_foundation.sldb" "typical.db“
    "slow.db" "fast.db"]
set target_library [list "typical.db" "slow.db" "fast.db"]
set symbol_library [list "generic.sdb"]
set synthetic_library [list "dw_foundation.sldb"]

set default_schematic_options {-size infinite}
set hdlin_translate_off_skip_text "TRUE"
set edifout_netlist_only "TRUE"
set verilogout_no_tri true
set plot_command {lpr -Plw}
set hdlin_auto_save_templates "TRUE"
set compile_fix_multiple_port_nets "TRUE"
```

What .synopsys_dc.setup defined

- search_path: the path to search for unsolved reference library or design
- link_library: the library used for interpreting input description
  - Any cells instantiated in your HDL code (Macro…)
- target_library: the ASIC technology that the design is mapped to (standard cell)
- symbol_library: used during schematic generation
- synthetic_library: designware library to be used

- Other variables

.db versus .lib file

slow.db

```
SOH SOH SOH   BS ETX 6 x8F SOH BS SOH ETX STX EN
taint_license SOH ETX BS contents EOT BEL slow
2000.11-SP1-1 NUL STX DC4 SOH ENQ ACK CAN libr
operations
processors
FF data_classes VT
library_types FF SI implementations
EOT slow ENQ ETX DC2 ENQ SOH SO EOT type ENQ SI DC
references NAK
netlist_cells SYN FF netlist_nets ETB
graphics_view CAN BEL aliases EM VT constrain
ACCSHCINX2 STX NUL NUL NUL NUL EOT NAK SOH SOH C
pin_number STX EOT STX"    pin_class STX ENQ
rise_power
0-STX ENQ . BS template/DC3 energy_template_
FS=!C|=#GSq=$ENQ-=$NUL xFC=#   =""xD5=RS
fall_power VT 06 STX /ETX RS ETX BEL=,BS1=x87
0 NUL SOH ENQ ; STX !B   SO0-STX/ETX RS ETX BEL=
x95=EOT騢=ENQfACK=ENQ   =ENQbxE1=EOT搧=
xB8 =VT|5=VT愔=VT   =
  =BSψ=EOT摞=
}g=VTJxED=VT扈=VTcVT=
wGS=BS   =EOT愁=VT3xDB=FF DC2'=FF   =FFZx
Z\=
Ec=FFx91BS=
椪=BEL0@=SIxC96=DLE舀=DC1;\=DC1/xD3=DL
E<US@xA3<DLExEAx9E<-xA3<<-6/<,dw<*xE2x
6xB5<
```

slow.lib

```
cell (AND2X1) {
  cell_footprint : and2;
  area : 6.789600;
  pin(A) {
    direction : input;
    capacitance : 0.001327;
  }
  pin(B) {
    direction : input;
    capacitance : 0.001542;
  }
  pin(Y) {
    direction : output;
    capacitance : 0.0;
    function : "(A B)";
    internal_power() {
      related_pin : "A";
      rise_power(energy_template_7x7) {
        index_1 ("0.042, 0.066, 0.112, 0.206, 0.
```

Area

Power

```
  timing() {
    related_pin : "A";
    timing_sense : positive_unate;
    cell_rise(delay_template_7x7) {
      index_1 ("0.042, 0.066, 0.112, 0.206, 0.
      index_2 ("0.00079, 0.002054, 0.00474, 0.
      values ( \
```

Timing (delay)

fast.lib versus slow.lib file

fast.lib ➔ for HOLD time

```
cell (AND2X1) {
  cell_footprint : and2;
  area : 6.789600;
```

```
timing() {
  related_pin : "A";
  timing_sense : positive_unate;
  cell_rise(delay_template_7x7) {
    index_1 ("0.02, 0.032, 0.056, 0.102
    index_2 ("0.00079, 0.002054, 0.0047
    values ( \
      "0.045522, 0.050983, 0.061495, 0.
      "0.047351, 0.052798, 0.063319, 0.
      "0.051040, 0.056467, 0.066916, 0.
      "0.055823, 0.061330, 0.071888, 0.
      "0.059203, 0.064842, 0.075497, 0.
      "0.057208, 0.063229, 0.074353, 0.
      "0.039006, 0.045647, 0.057870, 0.
  }
```

slow.lib ➔ for SETUP time

```
cell (AND2X1) {
  cell_footprint : and2;
  area : 6.789600;
```

```
timing() {
  related_pin : "A";
  timing_sense : positive_unate;
  cell_rise(delay_template_7x7) {
    index_1 ("0.042, 0.066, 0.112, 0.2
    index_2 ("0.00079, 0.002054, 0.004
    values ( \
      "0.120617, 0.134063, 0.159187, 0
      "0.125564, 0.138998, 0.164099, 0
      "0.135198, 0.148596, 0.173633, 0
      "0.153769, 0.167154, 0.192143, 0
      "0.177698, 0.191732, 0.217528, 0
      "0.199998, 0.215214, 0.242429, 0
      "0.207585, 0.224996, 0.255239, 0
  }
```

Outline

- Introduction to Synthesis Tools
- File Preparation
- Synthesis Flow
- Design Environment
- Design Constraints
- Gate-Level Simulation
- Synopsys Filters

Synthesis Flow

- Read Design File
- Read Design Constraints
- Compile Design
- Validate Design Constraints
- Output Netlist

} TCL

- **`dc_shell -f syn.tcl | tee syn.log`**

Read Design File

- Method 1
  - `read_file –format verilog <DESIGN FILE>`
- Method 2
  - `analyze –f verilog <DESIGN FILE>`
  - `elaborate <DESIGN NAME>`
  - Performs syntax check during analysis
  - analyze stores cached files, the design can be elaborated directly in the future

Link Design Module

- Link the modules to the link library
  - Module used but not defined in the design
  - e.g., SRAM macro
- `link`

Read Design Constraints

- Apply design constraints file
  - Clock signal
  - Input/Output
  - Area limit
  - DRC constraints
- **`source –echo –verbose <SDC FILE>`**

Compile Design

- Uniquify repeatedly instantiated design for better optimization with longer synthesis time
  - `uniquify`
- Enable the synthesis and start compiling
  - `compile …`
  - `compile_ultra`

Validate Design Constraints

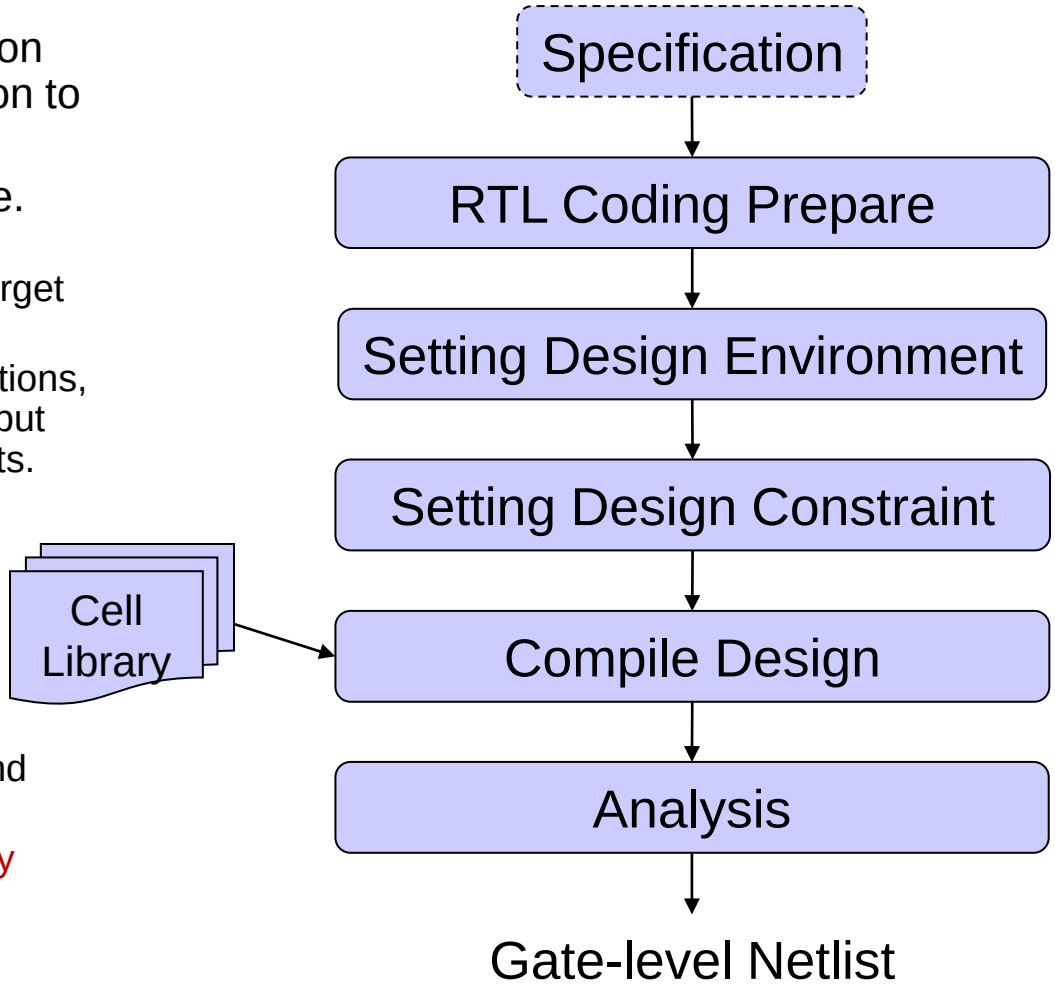
- Report synthesis status
  - `report_area -hierarchy`
  - `report_timing -delay min`
  - `report_timing -delay max`
  - `report_resource`
  - `report_timing`

Output Netlist

- DDC file
  - Save file for design compiler
  - `write –f ddc –hierarchy –output <DDC FILE>`
- Verilog file
  - Synthesized gate-level netlist
  - `write –f Verilog –hierarchy –output <NETLIST FILE>`
- Standard delay format file
  - Delay information of the design
  - `write_sdf –version 2.1 <SDF FILE>`
- Synthesized constraint file
  - `write_sdc –version 1.8 <SDC FILE>`

Synthesis Design Flow

- Develop the HDL design description and simulate the design description to verify that it is correct.
- Set up the .synopsys_dc.setup file.
  - Set the appropriate technology, synthetic, and symbol libraries, target libraries, and link libraries.
  - Set the necessary compilation options, including options to read in the input files and specify the output formats.
- Read the HDL design description.
- **Define the design.**
  - Set design attributes
  - Define environmental conditions
  - Set design rules
  - Set realistic constraints (timing and area goals)
  - Determine a compile methodology



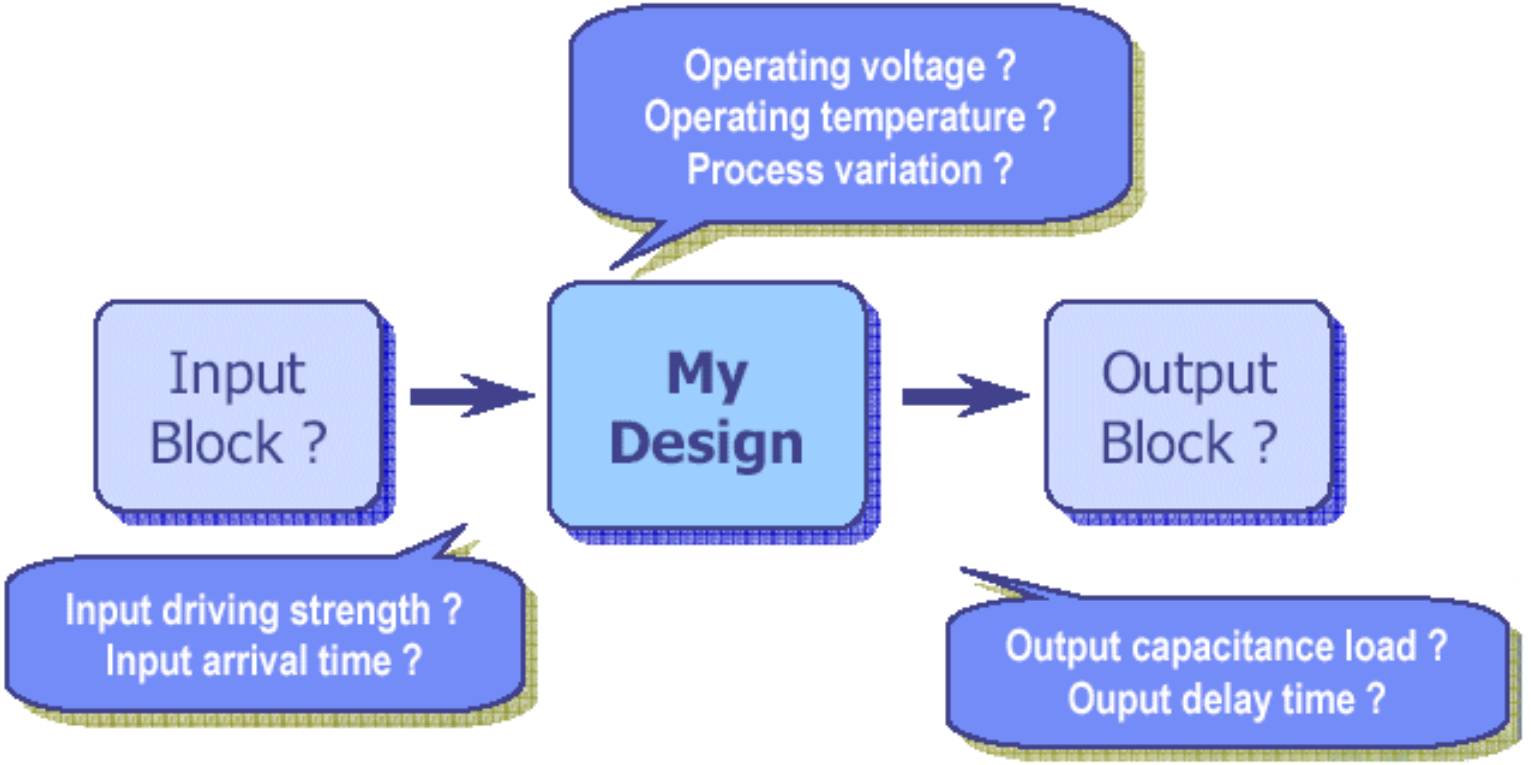
Outline

- Introduction to Synthesis Tools
- File Preparation
- Synthesis Flow
- Design Environment
- Design Constraints
- Gate-Level Simulation
- Synopsys Filters

Why Describes the Real World Environment

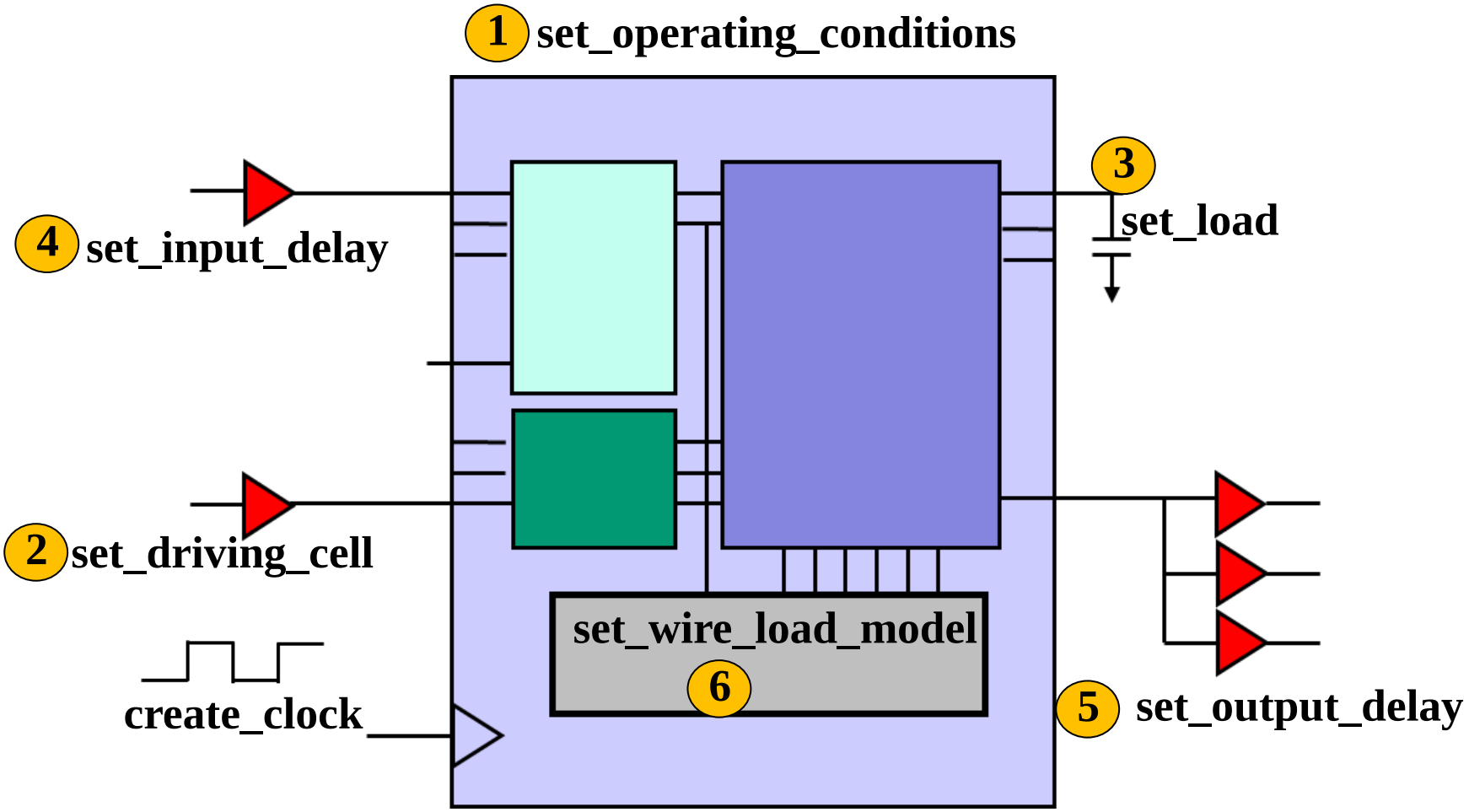
- Beware that the defaults are not realistic conditions.
  - Input drive is not infinite
  - Capacitive loading is usually not zero
  - Consider **process, temperature, and voltage(PVT)** variation
- The operating environment affects the components selected from target library and timing through your design.
- The real world environment you define describes the conditions that the circuit will operate within.

Describing Design Environment (1/2)



Tell dc timing delay and loading at each node
Inside : use slow.lib fast.lib in target library
Outside : sdc instruction about environment

Describing Design Environment (2/2)

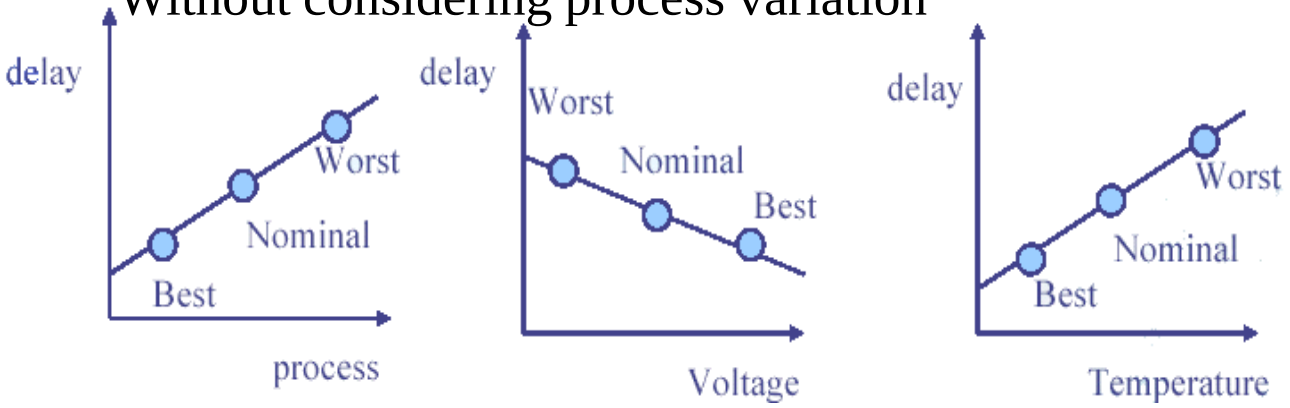


1. Setting Operating Environment

- Operating condition model scales components delay, directs the optimizer to simulate variations in process, temperature, and voltage. (Take 0.13um as example)

Name	Process	Temp	Volt	
slow	1	125	1.62	-10%
typical	1	25	1.8	Normal voltage
fast	1	-40	1.98	+10%

Without considering process variation



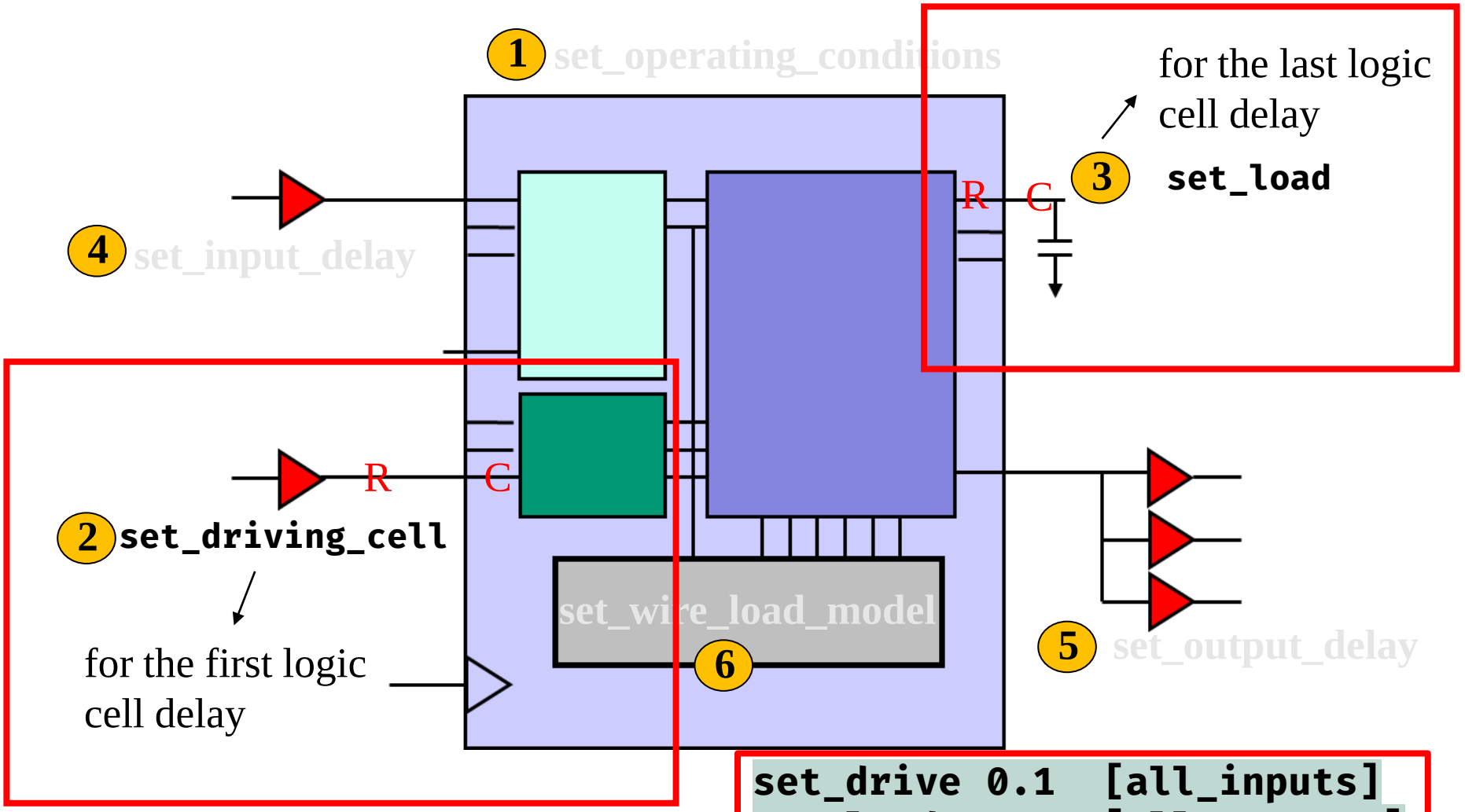
1. Setting Operating Environment

- `set_operating_conditions`
  `–min_library MIN_LIB –min MIN_COND`
  `–max_library MAX_LIB –max MAX_COND`

- With 0.18 technode
  - `set_operating_conditions`
    `-min_library fast –min fast`
    `-max_library slow –max slow`

Name	Process	Temp	Volt
slow	1	125	1.62
typical	1	25	1.8
fast	1	-40	1.98

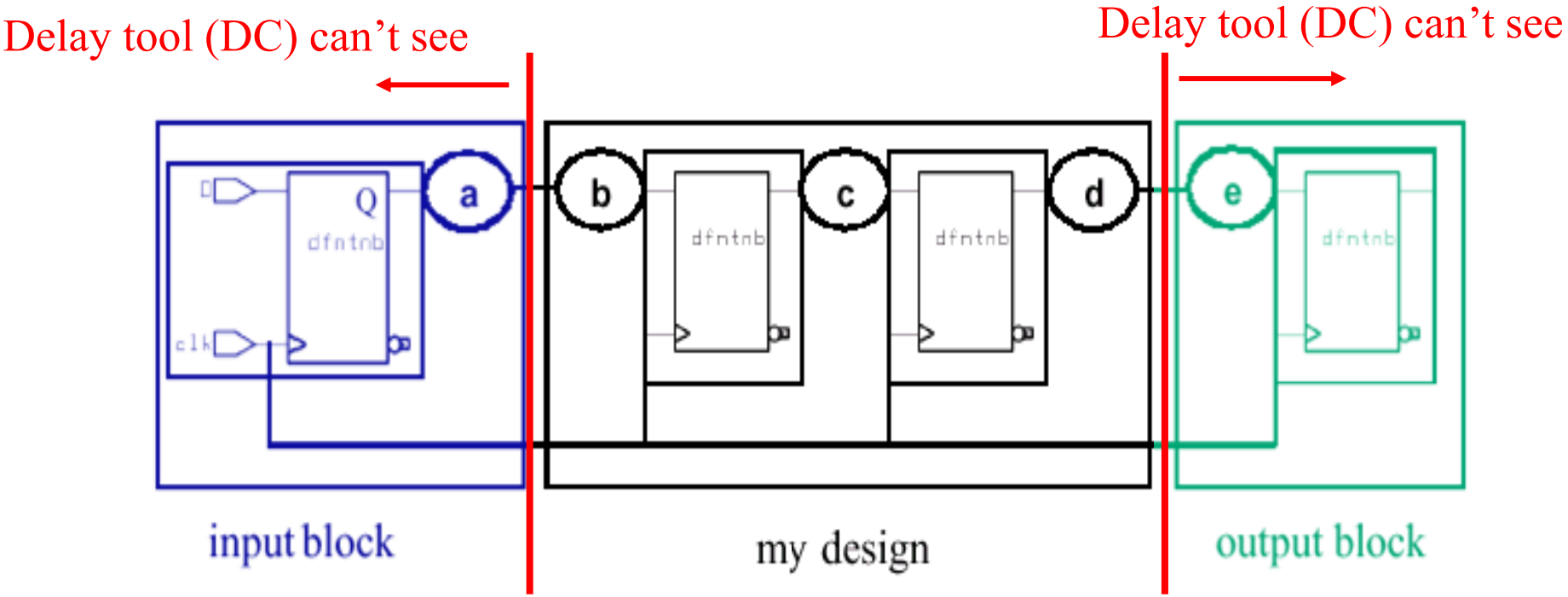
2,3. Setting Driving Cell and Output Load




```
set_drive 0.1  [all_inputs]
set_load  0.05 [all_outputs]
```

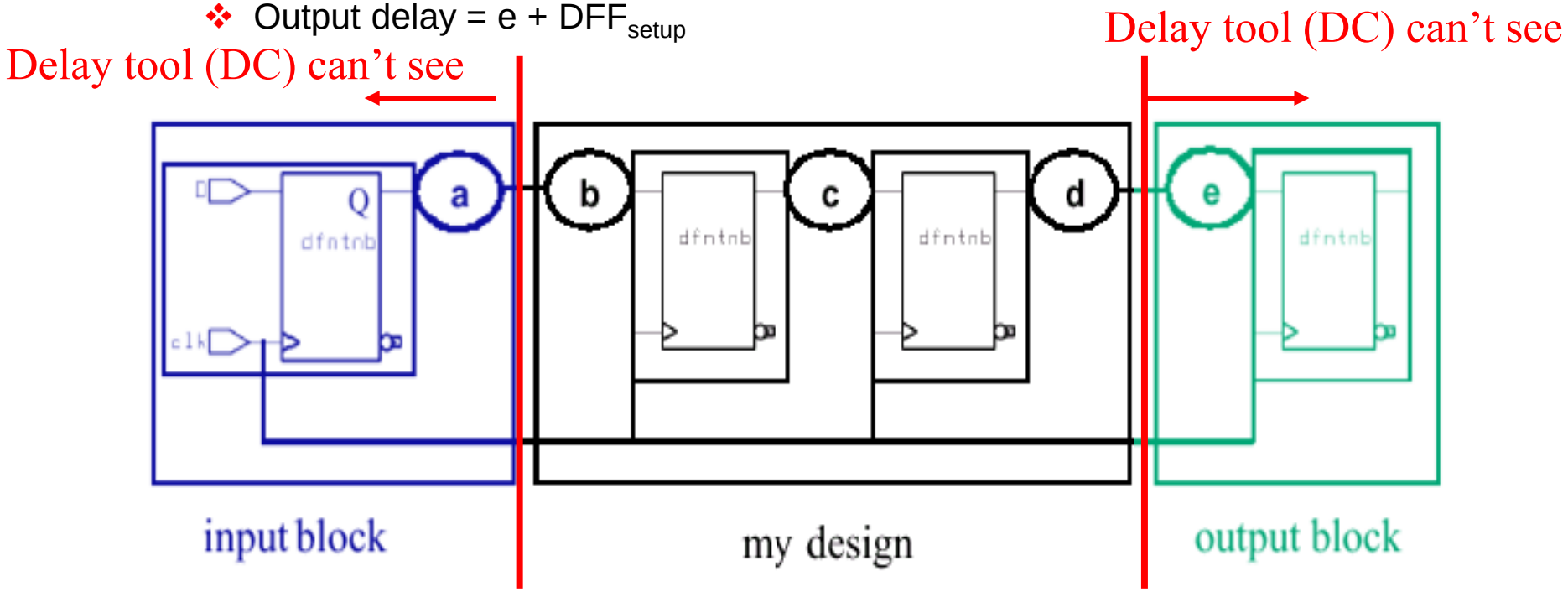
4,5 Input/Output Delay (Not Real)

- Clock cycle ≥ $DFF_{clk\text{-}Qdelay}$ + c + DFF_{setup}
- Clock cycle ≥ b + DFF_{setup}
- Clock cycle ≥ $DFF_{clk\text{-}Qdelay}$ + d



4,5 Input/Output Delay (Real)

- Clock cycle ≥ $DFF_{clk\text{-}Qdelay}$ + c + DFF_{setup}
- Clock cycle ≥ **$DFF_{clk\text{-}Qdelay}$ + a** + b + DFF_{setup}
  - Input delay = $DFF_{clk\text{-}Qdelay}$ + a
- Clock cycle ≥ $DFF_{clk\text{-}Qdelay}$ + d + **e + DFF_{setup}**
  - Output delay = e + DFF_{setup}



4,5. Setting Input/Output Delay

- Input delay
  - `set_input_delay  <T_IN>  -clock <CLOCK> <PORTS>`
- Output delay
  - `set_output_delay <T_OUT> -clock <CLOCK> <PORTS>`

6. Setting Wire Load Model

- WLM estimates wire capacitance based on chip area & cell fanout
- Setting this information during compile in order to model the design more accurately **(timing).**
- If you don't use WLM, you will have **no net delay and net area**

- `set_wire_load_model –name <WLM> –library <LIB>`
  - `set_wire_load_model –name tsmc13_w10 –library slow`

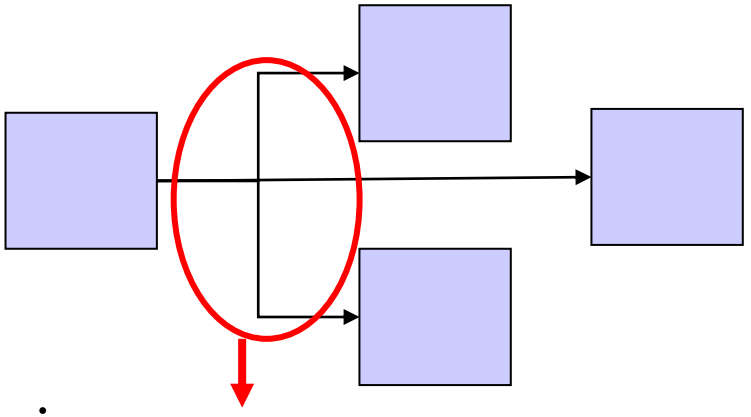
6. Wire Load Model in slow.db

```
wire_load("tsmc13_wl10") {
  resistance  : 8.5e-8;
  capacitance : 1.5e-4;
  area        : 0.7;
  slope       : 66.667;
  fanout_length    (1,66.667);
}
wire_load("tsmc13_wl20") {
  resistance  : 8.5e-8;
  capacitance : 1.5e-4;
  area        : 0.7;
  slope       : 133.334;
  fanout_length    (1,133.334);
}
wire_load("tsmc13_wl30") {
  resistance  : 8.5e-8;
  capacitance : 1.5e-4;
  area        : 0.7;
  slope       : 200.001;
  fanout_length    (1,200.001);
}
```

2x

Estimated distance between gates

Take tsmc13_w110 as example:



Cwire =
(fanout=3 ➔ length=66.667*3)*
capacitance coeff. (1.5e-4) = 0.03pf

Net Area = **(inaccurate)**
(fanout=3 ➔ length=66.667*3)*
area coeff. (0.7) = 140 um^2

SDC about Setting Design Environment

```
set CYCLE 10

create_clock -period $CYCLE [get_ports  clk]
set_dont_touch_network      [get_clocks clk]
set_clock_uncertainty  0.1  [get_clocks clk]
set_clock_latency      0.5  [get_clocks clk]

set_input_delay 1 -clock clk \
    [remove_from_collection [all_inputs] [get_ports clk]]
set_output_delay 0.5 -clock clk [all_outputs]
set_load         1              [all_outputs]
set_drive        1              [all_inputs]

set_operating_conditions -max_library slow -max slow
set_wire_load_model -name tsmc13_wl10 -library slow
set_max_fanout   20             [all_inputs]
```

Outline

- Introduction to Synthesis Tools
- File Preparation
- Synthesis Flow
- Design Environment
- Design Constraints
- Gate-Level Simulation
- Synopsys Filters

Define Clock Specification

- We need to accurately specify the clock including clock routing details in the early design stage to achieve timing convergence

- What should be defined?
  - **Clock period**
  - **Waveform**
  - **Uncertainty**
    - **Skew**
  - Latency
    - Source latency
    - Network latency
  - Transition
    - Input transition
    - Clock transition

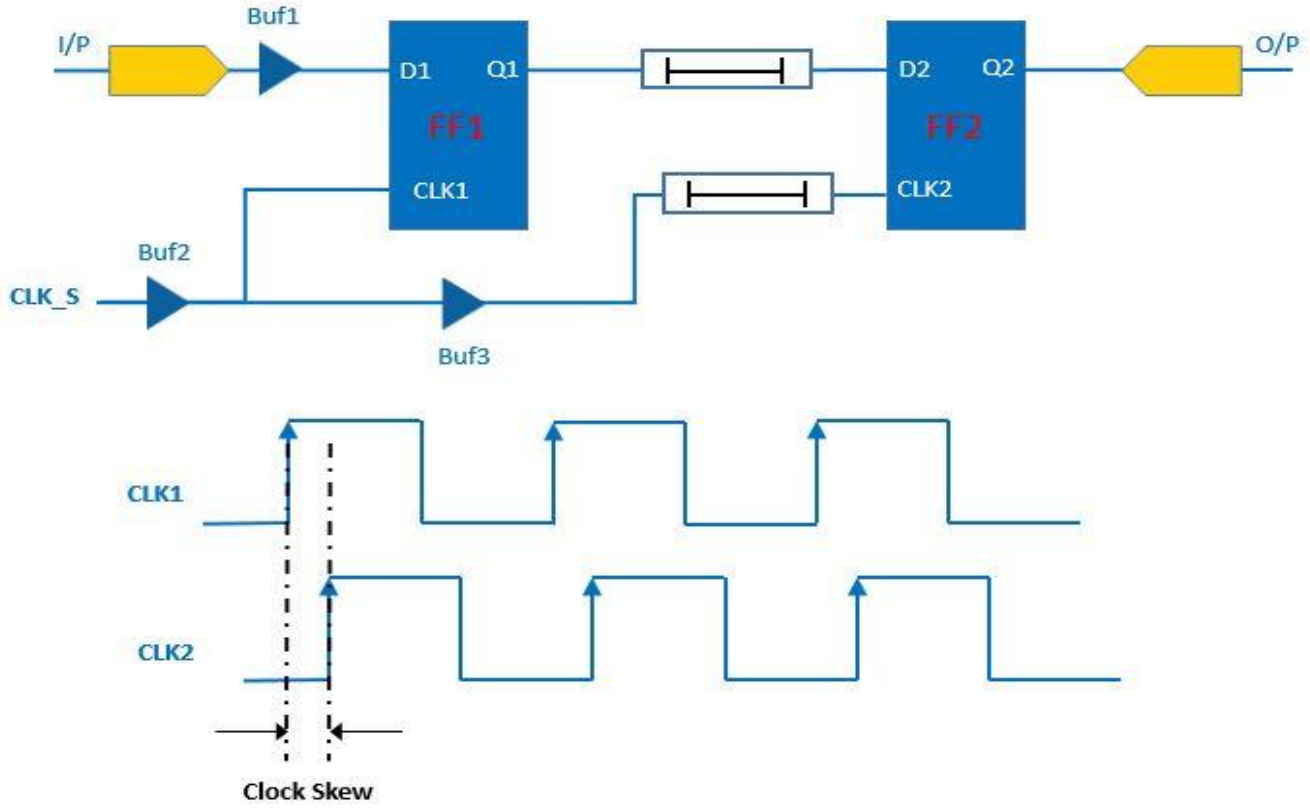
⇐ Focus on these three specs

Specify Clock Constrains

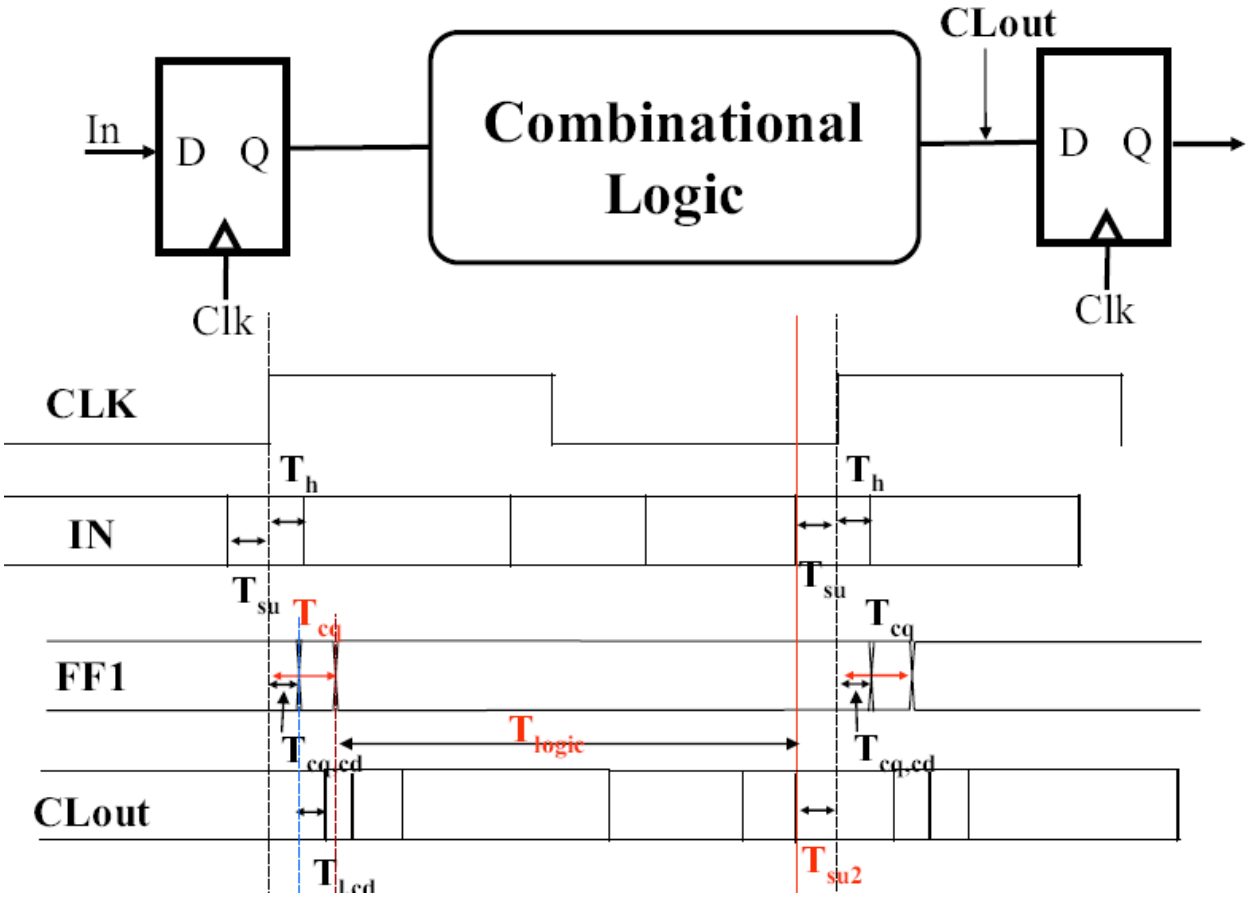
- Define your clock's waveform
  - `create_clock –name <CLOCK NAME> –period <CYCLE> [get_ports <CLOCK PORT>]`
- Respect the hold time requirement of all clocked flip-flops
  - `set_fix_hold [get_clocks <CLOCK NAME>]`
- Do not add buffer to clock network
  - `set_dont_touch_network [get_clocks <CLOCK NAME>]`
- No net delay in clock path
  - `set_ideal_network [get_ports <CLOCK PORT>]`

Clock Uncertainty (Skew)

- The maximum difference between the arrival of clock signals at sequential cells
  - `set_clock_uncertainty <UNCERTAINTY> [get_clocks <CLOCK NAME>]`

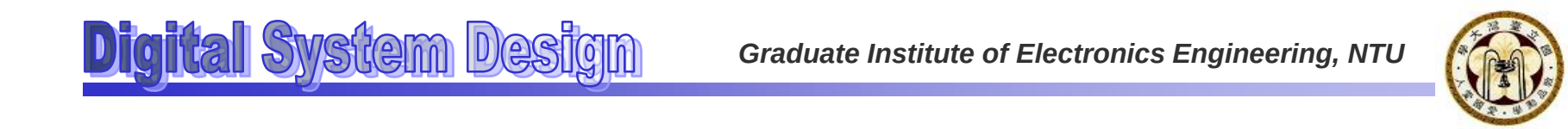


System Timing: Maximum Delay

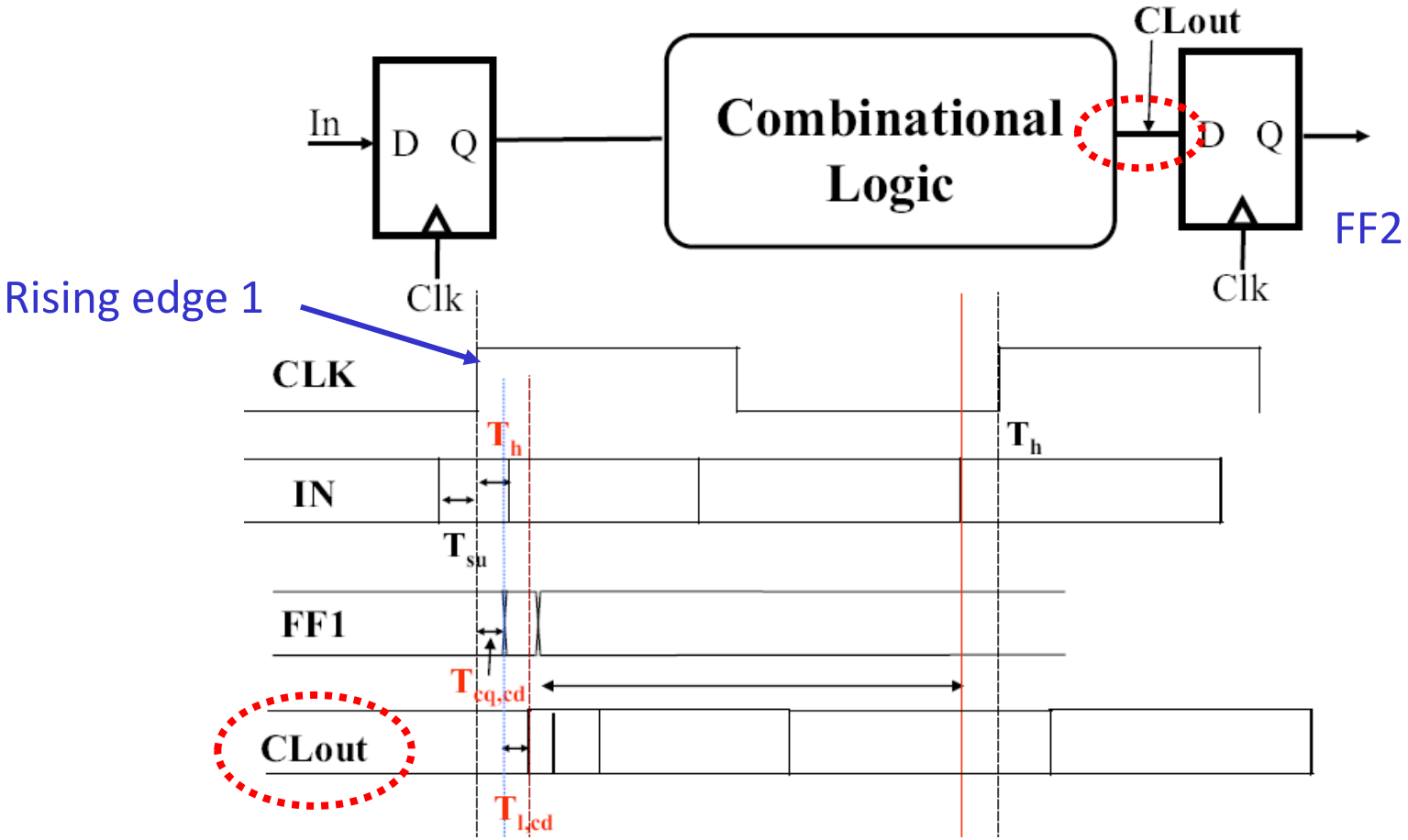


$$T_{logic} < T - T_{cq} - T_{su2}$$

$$Slack_{setup} = T - T_{cq} - T_{su2} - T_{logic}$$

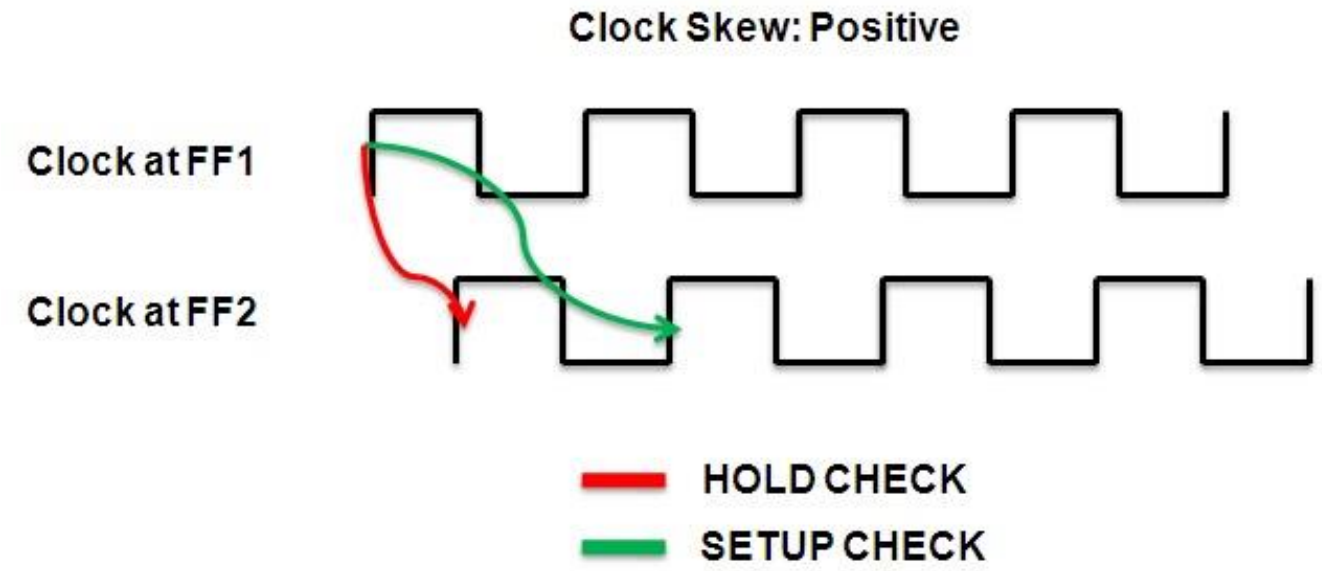


System Timing: Minimum Delay



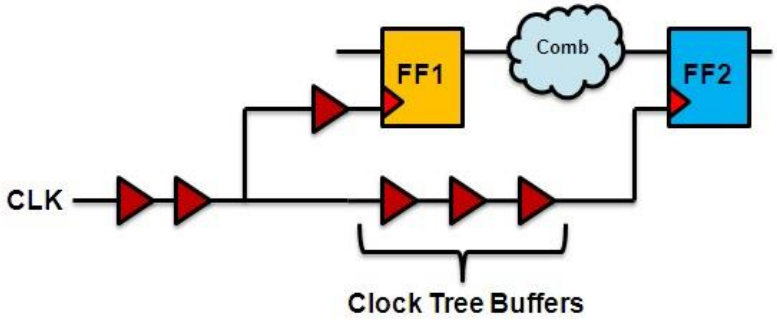
$$T_{cq,cd} + T_{logic,cd} > T_{hold2} \Rightarrow Slack_{hold} = T_{cq,cd} + T_{logic,cd} - T_{hold2}$$

Clock Uncertainty (Positive Skew)

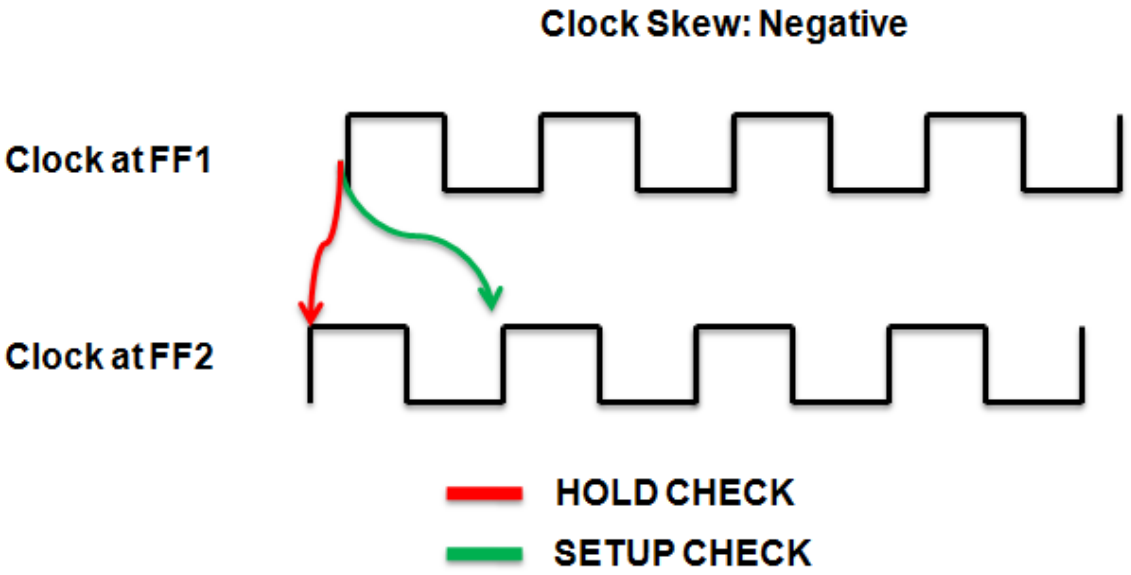


$$\text{Setup Slack} = T_{clock} - T_{(clk\text{-}q)FF1} - T_{comb} - T_{su,FF2} + T_{skew}$$

$$\text{Hold Slack} = T_{(clk\text{-}q)FF1} + T_{comb} - T_{ho,FF2} - T_{skew}$$

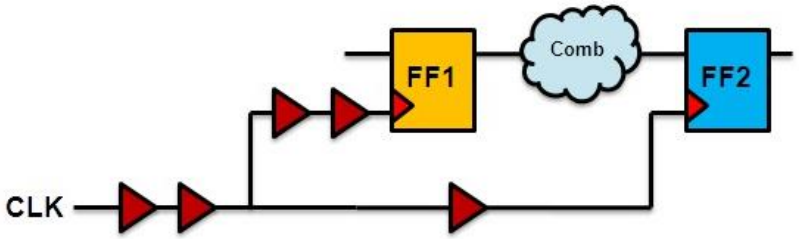


Clock Uncertainty (Negative Skew)



$$\text{Setup Slack} = T_{clock} - T_{(clk\text{-}to\text{-}q)FF1} - T_{comb} - T_{su,FF2} - T_{skew}$$

$$\text{Hold Slack} = T_{(clk\text{-}to\text{-}q)FF1} + T_{comb} - T_{ho,FF2} + T_{skew}$$



Setting Area Constraint

- Area Unit : um^2 (follow unit defined in library)
  - **`set_max_area <AREA LIMIT>`**
  - Set area limit to 0 for minimized area

Design Rule Constraints

- **Design rules constraints can't be violated at any cost**, even if it will violate the timing and area goal.

- Three kinds of design rule constraints are set:
  - `set_max_transition`
  - `set_max_fanout`
  - `set_max_capacitance`

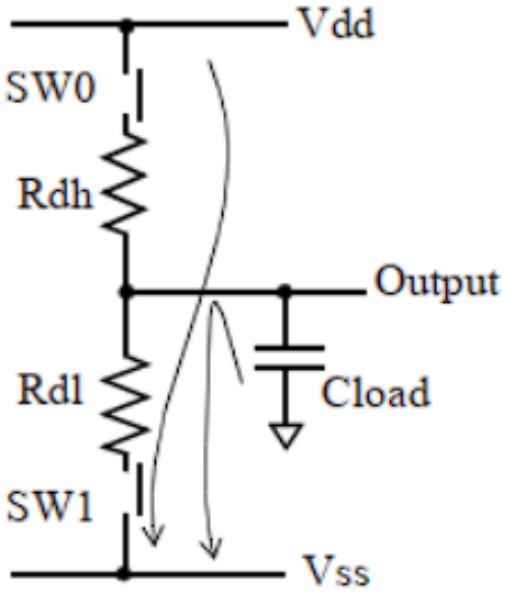
Setting the Fanout Load

- Vendors impose design rules that restrict how many cells are connected to one another based on capacitance, transition, and fanout

- Fanout
  - Number of load gates connected to the output of the driving gate

```
set_max_capacitance 0.2 [all_inputs]
set_max_transition  0.2 [all_inputs]
set_max_fanout      6   [all_inputs]
```

The functions among these three command are same, you just need to use one



Constraint Priority

- During the optimization, there exists a constraint priority relationship
  - Design Rule Constraint
  - Timing Constraint
  - Power Constraint
  - Area Constraint

SDC about Setting Design Constraint

```
set CYCLE 10

create_clock -period $CYCLE [get_ports  clk]
set_dont_touch_network      [get_clocks clk]
set_clock_uncertainty  0.1  [get_clocks clk]
set_clock_latency      0.5  [get_clocks clk]

set_input_delay 1 -clock clk \
    [remove_from_collection [all_inputs] [get_ports clk]]
set_output_delay 0.5 -clock clk [all_outputs]
set_load         1              [all_outputs]
set_drive        1              [all_inputs]

set_operating_conditions -max_library slow -max slow
set_wire_load_model -name tsmc13_wl10 -library slow
set_max_fanout   20             [all_inputs]
```

Outline

- Introduction to Synthesis Tools
- File Preparation
- Synthesis Flow
- Design Environment
- Design Constraints
- Gate-Level Simulation
- Synopsys Filters

gate-level .v file and .sdf file

```
module CONV_DW01_dec_0 ( A, SUM );
  input [11:0] A;
  output [11:0] SUM;
  wire   n1, n2, n3, n4, n5, n6, n7, n8, n9

  OR2X1 U1 ( .A(A[1]), .B(A[0]), .Y(n10) );
  CLKINVX1 U2 ( .A(A[9]), .Y(n1) );
  OAI21XL U3 ( .A0(n2), .A1(n1), .B0(n3), .
  AO21X1 U4 ( .A0(n4), .A1(A[8]), .B0(n2),
  OAI2BB1X1 U5 ( .A0N(n5), .A1N(A[7]), .B0(
  OAI2BB1X1 U6 ( .A0N(n6), .A1N(A[6]), .B0(
  OAI2BB1X1 U7 ( .A0N
  OAI2BB1X1 U8 ( .A0N
  OAI2BB1X1 U9 ( .A0N
  OAI2BB1X1 U10 ( .A0
  OAI2BB1X1 U11 ( .A0
  XOR2X1 U12 ( .A(A[1
  NOR2X1 U13 ( .A(A[1
  XNOR2X1 U14 ( .A(A[
  NAND2X1 U15 ( .A(n2
```

syn.v file

Timing info. (delay)

```
(CELL
  (CELLTYPE "OR2X1")
  (INSTANCE U1100)
  (DELAY
    (ABSOLUTE
    (IOPATH A Y (0.266:0.267:0.267) (0.340:0.342:0.342))
    (IOPATH B Y (0.271:0.273:0.273) (0.377:0.378:0.378))
    )
  )
```

syn.sdf file

Rise delay

Fall delay

min typ max

gate-level .v file and tsmc13 file

syn.v file

```
module CONV_DW01_dec_0 ( A, SUM );
  input [11:0] A;
  output [11:0] SUM;
  wire   n1, n2, n3, n4, n5, n6, n7,

  OR2X1 U1 ( .A(A[1]), .B(A[0]), .Y
  CLKINVX1 U2 ( .A(A[9]), .Y(n1) );
  OAI21XL U3 ( .A0(n2), .A1(n1), .B0
  AO21X1 U4 ( .A0(n4), .A1(A[8]), .B
  OAI2BB1X1 U5 ( .A0N(n5), .A1N(A[7
  OAI2BB1X1 U6 ( .A0N(n6), .A1N(A[6
  OAI2BB1X1 U7 ( .A0N(n7), .A1N(A[5
  OAI2BB1X1 U8 ( .A0N(n8), .A1N(A[4
  OAI2BB1X1 U9 ( .A0N(n9), .A1N(A[3
  OAI2BB1X1 U10 ( .A0N(n10), .A1N(A
  OAI2BB1X1 U11 ( .A0N(A[0]), .A1N(A
  XOR2X1 U12 ( .A(A[11]), .B(n11),
  NOR2X1 U13 ( .A(A[10]), .B(n3), .Y
  XNOR2X1 U14 ( .A(A[10]), .B(n3),
  NAND2X1 U15 ( .A(n2), .B(n1), .Y(n
```

tsmc13.v

```
`timescale 1ns/1ps
`celldefine
module OR2X1 (Y, A, B);
output Y;
input A, B;

or (Y, A, B);
```

Gate function

```
specify
  // delay parameters
  specparam
    tplh$A$Y = 1.0,
    tphl$A$Y = 1.0,
    tplh$B$Y = 1.0,
    tphl$B$Y = 1.0;

  // path delays
  (A *> Y) = (tplh$A$Y, tphl$A$Y);
  (B *> Y) = (tplh$B$Y, tphl$B$Y);
endspecify

endmodule // OR2X1
```

Default delay (Wrong!)

Gate-Level Simulation (Verilog)

- Modify your testbench file to include timing delay

  $sdf_annotate ("SDF_FILE_NAME", top_module_instance_name);

```
`define SDFFILE   "CHIP_syn.sdf"
initial $sdf_annotate(`SDFFILE, chip0);
```

- Gate level simulation with timing information:

  >> vcs testbench.v design_syn.v -v cell_model.v +define+SDF –full64 –R –debug_access+all +v2k

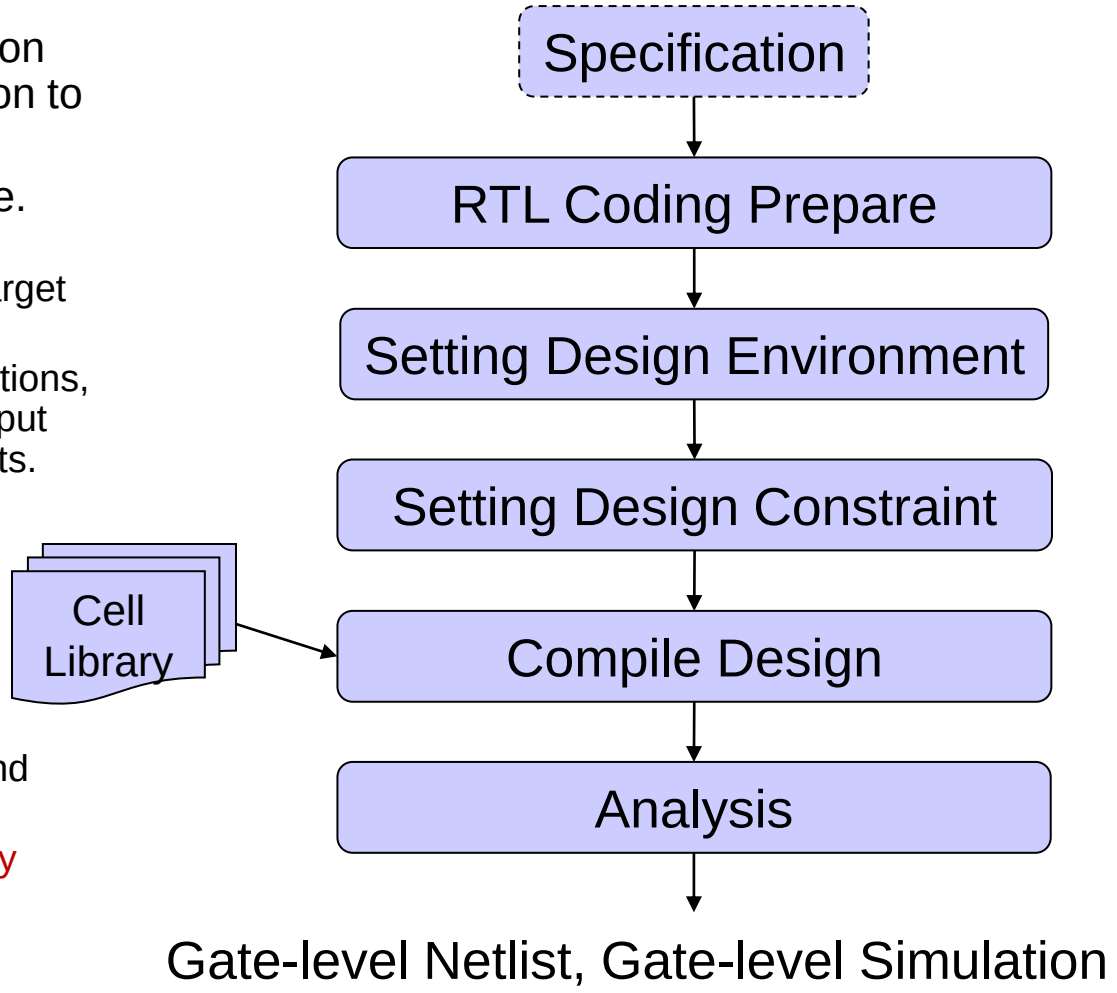
Without this, simulator cant find the according gates in your synthesized file

Without this, simulator will use the delay info in cell_model.v, which must be wrong

```
➢ vcs testbench.v design_syn.v  -v tsmc13.v  +define+SDF
  –full64 –R –debug_access+all +v2k
```

Synthesis Design Flow

- Develop the HDL design description and simulate the design description to verify that it is correct.
- Set up the .synopsys_dc.setup file.
  - Set the appropriate technology, synthetic, and symbol libraries, target libraries, and link libraries.
  - Set the necessary compilation options, including options to read in the input files and specify the output formats.
- Read the HDL design description.
- **Define the design.**
  - Set design attributes
  - Define environmental conditions
  - Set design rules
  - Set realistic constraints (timing and area goals)
  - Determine a compile methodology



Outline

- Introduction to Synthesis Tools
- File Preparation
- Synthesis Flow
- Design Environment
- Design Constraints
- Gate-Level Simulation
- Synopsys Filters

Common Collection Command

all_clocks	get_cells
all_ideal_ents	get_clock
all_inputs	get_clocks
all_outputs	get_designs
all_registers	get_nets
	get_pins
	get_ports

```
set_input_delay 1 –clock clk \
    [remove_from_collection [all_inputs] [get_ports clk]]

set_output_delay 1 –clock clk [all_outputs]

foreach_in_collection latch [get_cells –filter \
    “full_name=~*latch*”] {
    echo “[get_object_name $latch]”
}
```

Synopsys Filter

- get_* commands usually allow **-filter** option
  - `get_cells -filter "full_name=~*latch*"`
  - `get_cells -filter "full_name=~*reg* && ref_name~=*BUF*"`
  - `get_cells -filter "area>3 && number_of_pins==2"`

  - ```
    get_pins \
      -of [get_cells -filter "number_of_pins>2"] \
      -filter "pin_direction==out && full_name!~*BUF*"
    ```
    ```