Logic Synthesis with Design Compiler

Speaker: Yi-Ping Huang

Advisor: Prof. Shao-Yi Chien

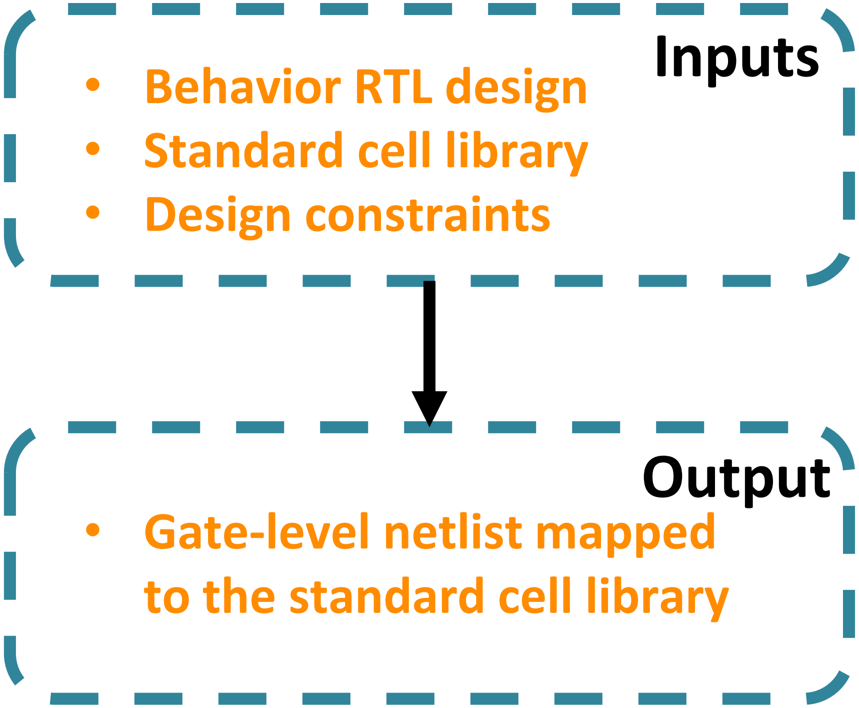
Date: 2025/08/04

Outline

- Introduction
- What is Logic Synthesis?
- Synopsys Design Compiler
- Setting Design Environment
- Setting Design Constraints
- Synthesis Report and Analysis
- Gate-Level Simulation

What is Logic Synthesis?

- Synthesis is the process to convert RTL into a gate-level netlist optimized with a set of design constraints.

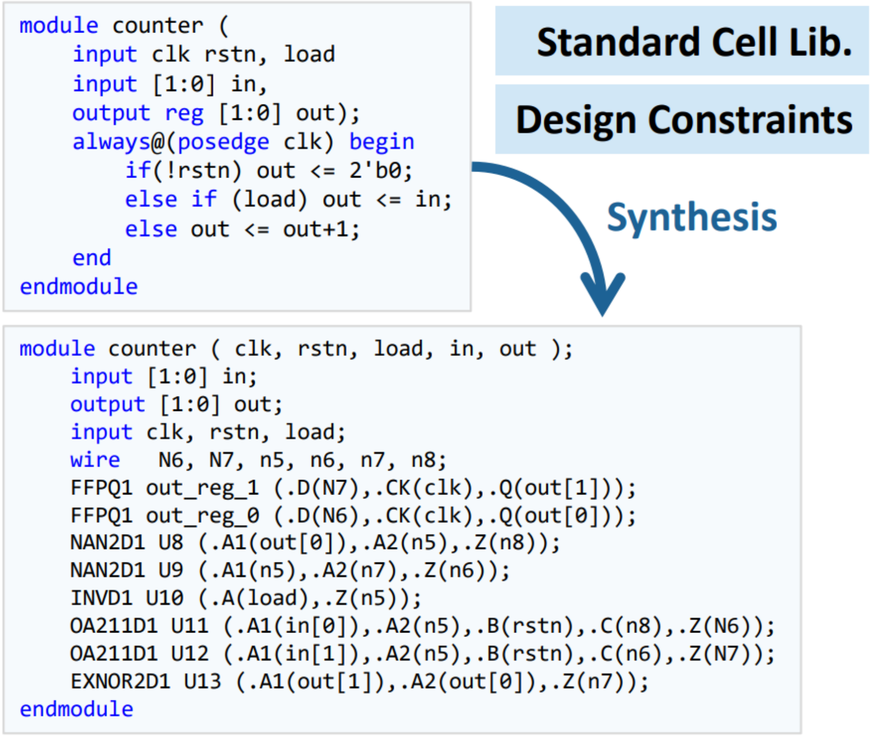


Inputs

- Behavior RTL design
- Standard cell library
- Design constraints

Output

- Gate-level netlist mapped to the standard cell library



```
module counter (
    input clk rstn, load
    input [1:0] in,
    output reg [1:0] out);
    always@(posedge clk) begin
        if(!rstn) out <= 2'b0;
        else if (load) out <= in;
        else out <= out+1;
    end
endmodule
```

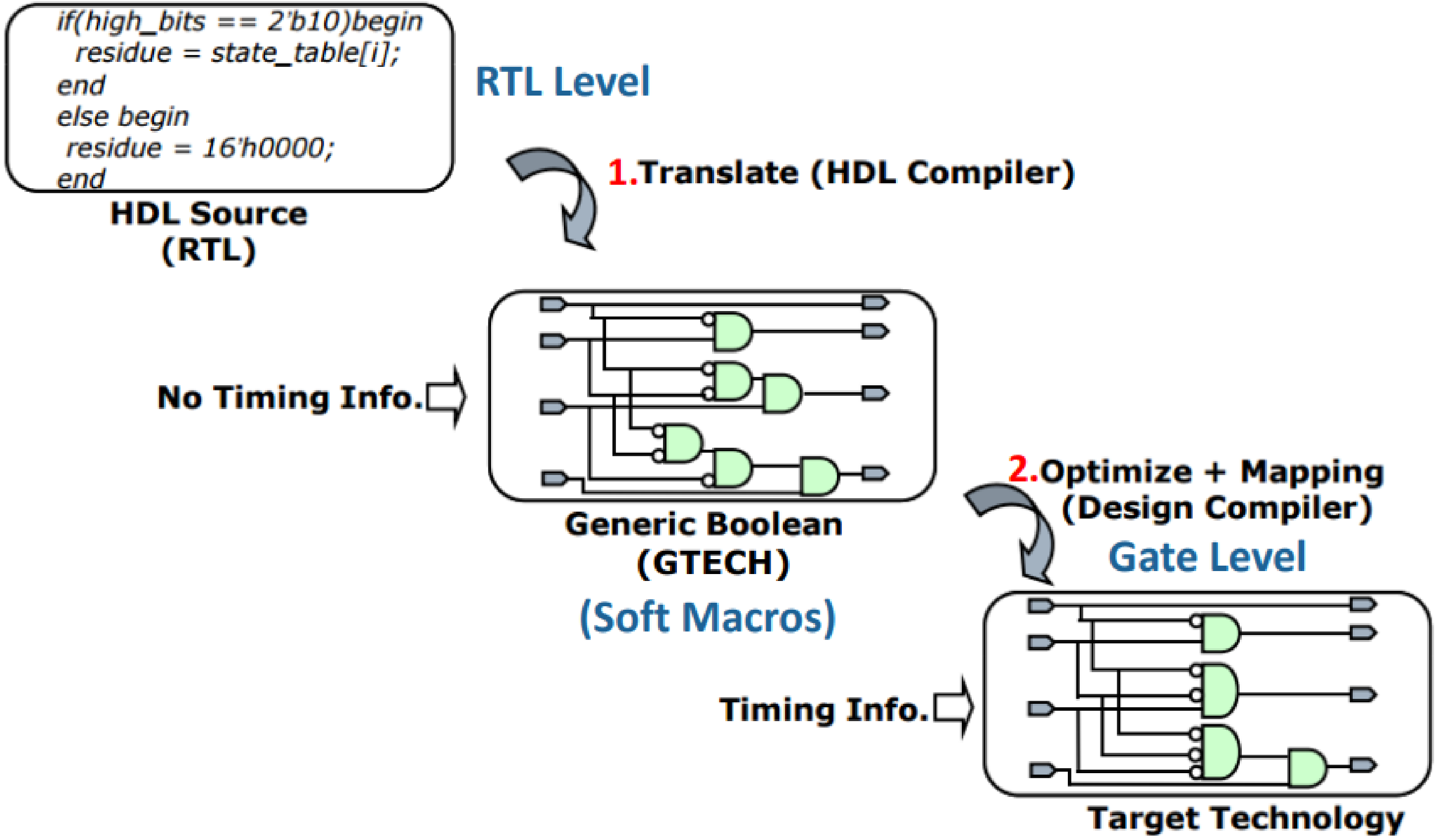
Standard Cell Lib.

Design Constraints

Synthesis

```
module counter ( clk, rstn, load, in, out );
    input [1:0] in;
    output [1:0] out;
    input clk, rstn, load;
    wire   N6, N7, n5, n6, n7, n8;
    FFPQ1 out_reg_1 (.D(N7),.CK(clk),.Q(out[1]));
    FFPQ1 out_reg_0 (.D(N6),.CK(clk),.Q(out[0]));
    NAN2D1 U8 (.A1(out[0]),.A2(n5),.Z(n8));
    NAN2D1 U9 (.A1(n5),.A2(n7),.Z(n6));
    INVD1 U10 (.A(load),.Z(n5));
    OA211D1 U11 (.A1(in[0]),.A2(n5),.B(rstn),.C(n8),.Z(N6));
    OA211D1 U12 (.A1(in[1]),.A2(n5),.B(rstn),.C(n6),.Z(N7));
    EXNOR2D1 U13 (.A1(out[1]),.A2(out[0]),.Z(n7));
endmodule
```

What is Logic Synthesis?



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

HDL Source
(RTL)

RTL Level

1.Translate (HDL Compiler)

No Timing Info.

Generic Boolean
(GTECH)

(Soft Macros)

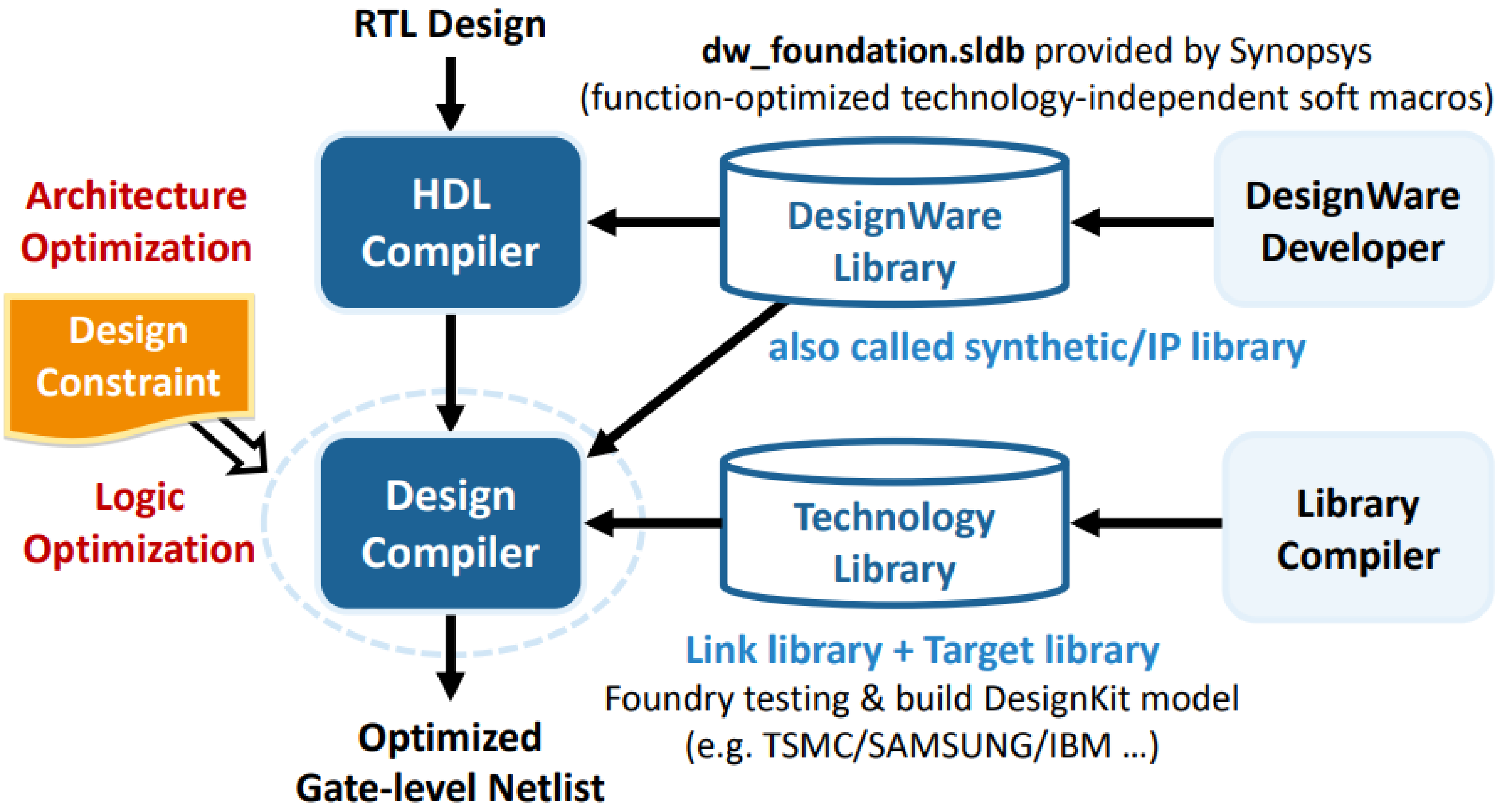
2.Optimize + Mapping
(Design Compiler)

Gate Level

Timing Info.

Target Technology

What is Logic Synthesis?



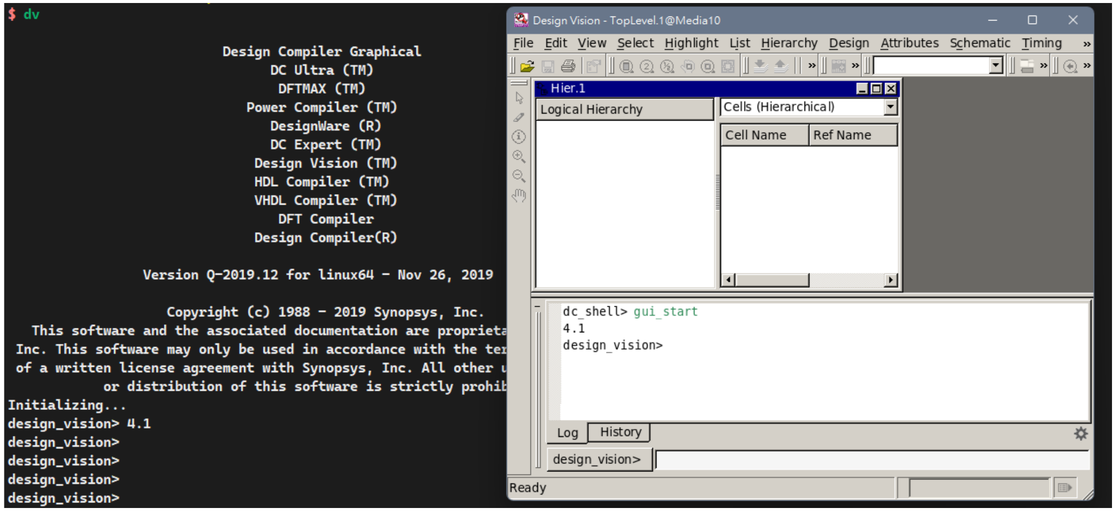
dw_foundation.sldb provided by Synopsys
(function-optimized technology-independent soft macros)

also called synthetic/IP library

Link library + Target library
Foundry testing & build DesignKit model
(e.g. TSMC/SAMSUNG/IBM …)

How to launch Design Compiler

- GUI Mode
  - **Command: dv**



```
$ dv

                         Design Compiler Graphical
                               DC Ultra (TM)
                                DFTMAX (TM)
                            Power Compiler (TM)
                              DesignWare (R)
                              DC Expert (TM)
                            Design Vision (TM)
                             HDL Compiler (TM)
                            VHDL Compiler (TM)
                               DFT Compiler
                            Design Compiler(R)

                 Version Q-2019.12 for linux64 - Nov 26, 2019

                    Copyright (c) 1988 - 2019 Synopsys, Inc.
   This software and the associated documentation are proprieta
 Inc. This software may only be used in accordance with the ter
 of a written license agreement with Synopsys, Inc. All other u
            or distribution of this software is strictly prohib
Initializing...
design_vision> 4.1
design_vision>
design_vision>
design_vision>
design_vision>
```

```
dc_shell> gui_start
4.1
design_vision>
```

Design Vision - TopLevel.1@Media10
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing
Hier.1
Logical Hierarchy
Cells (Hierarchical)
Cell Name
Ref Name
Log
History
design_vision>
Ready

- DC-TCL command Mode **(recommended*)**
  - **Command:**

    %dc_shell

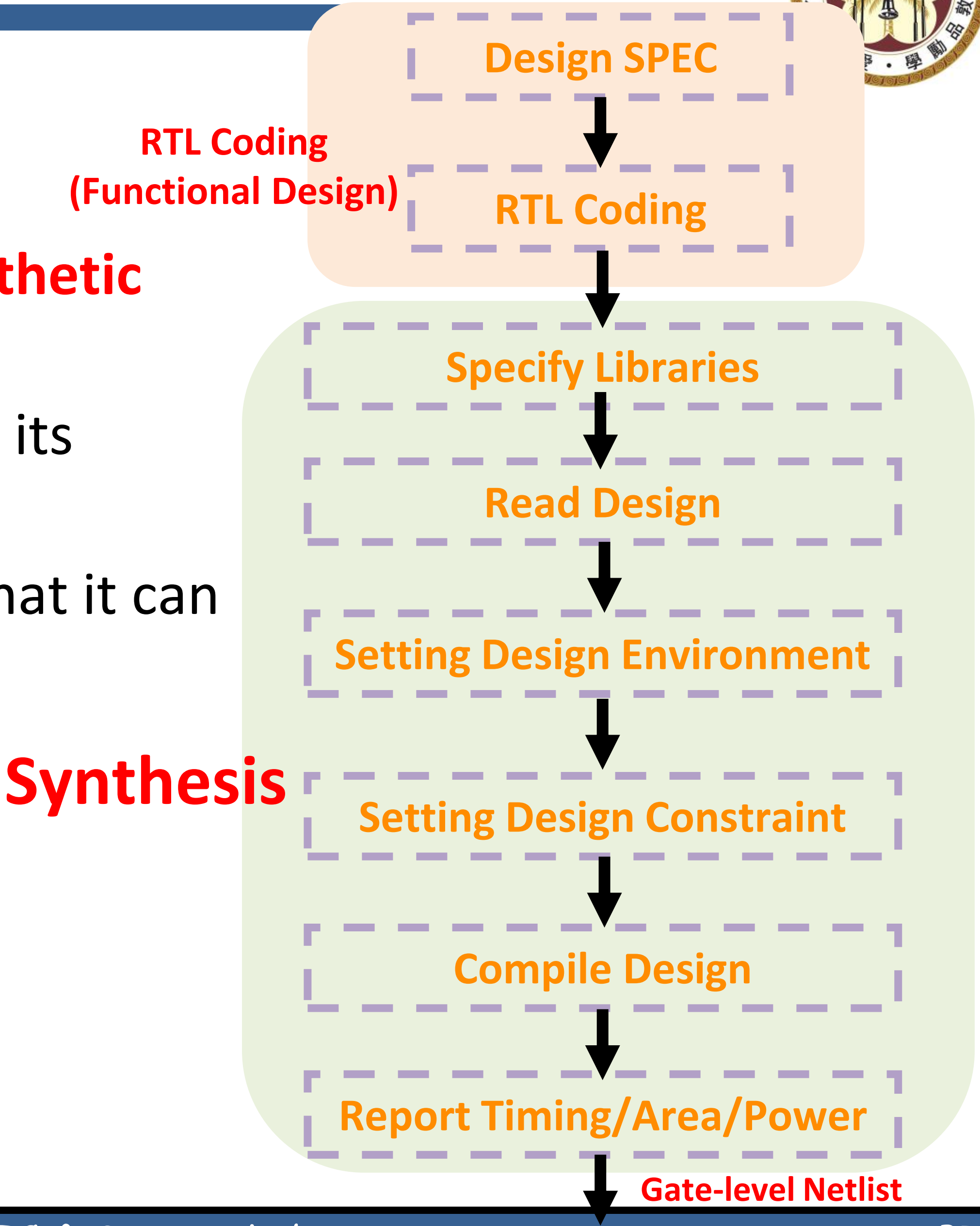
    %dc_shell> gui_start

    %dc_shell> gui_stop

- Before synthesis, we usually prepare a tcl file.
  - **Command: dc_shell –f xxxxxxxxx.tcl**

Synthesis Design Flow

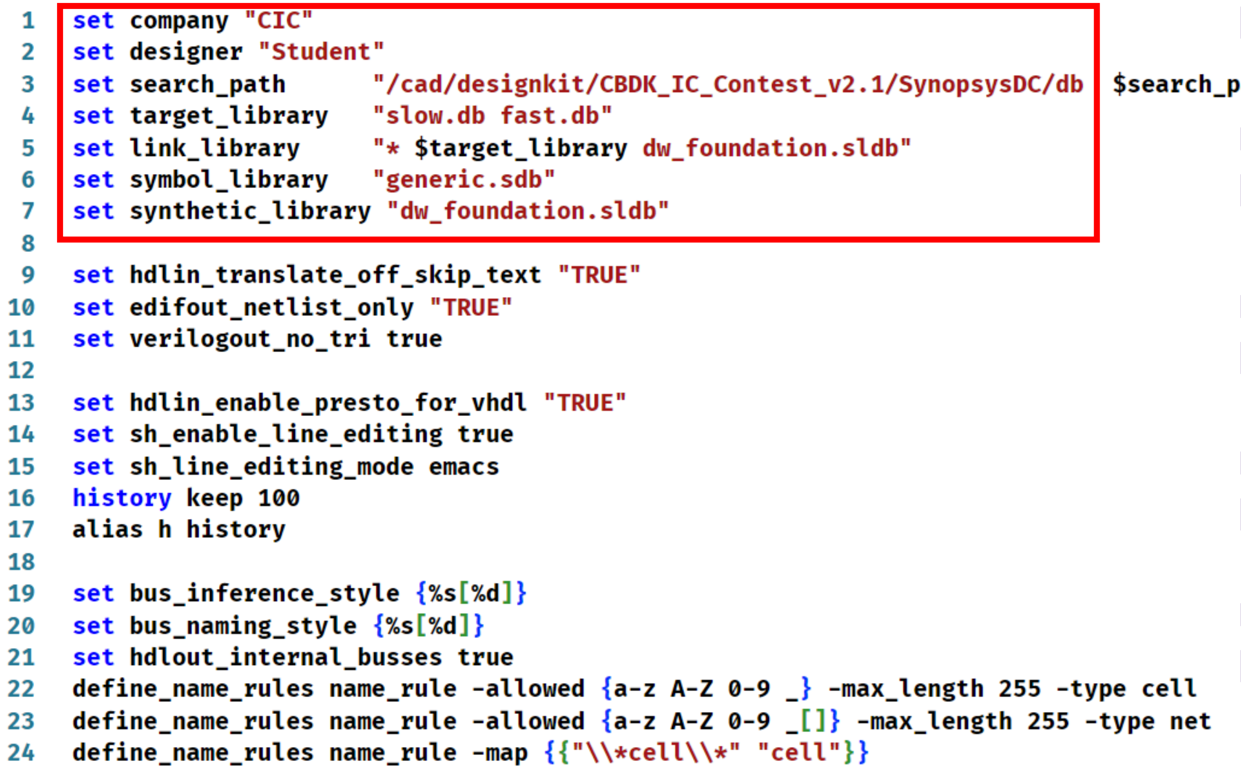
- Develop the RTL design (**Synthetic design**)
- Simulate the design to verify its functionality.
- Run the Synthesis to verify that it can meet the specification (Timing/Power/Area).



Synthesis Design Flow - Specify Libraries

- Set up the .synopsys_dc.setup file.

Need to modify search_path in different environment (media10/11, NTU IC Design Lab …)

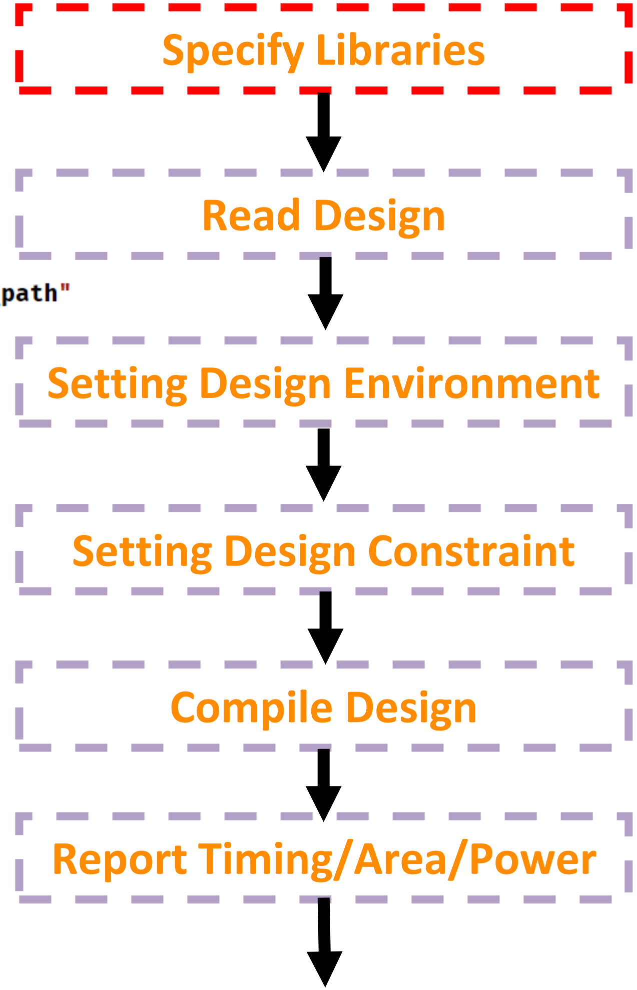


```
set company "CIC"
set designer "Student"
set search_path      "/cad/designkit/CBDK_IC_Contest_v2.1/SynopsysDC/db $search_path"
set target_library   "slow.db fast.db"
set link_library     "* $target_library dw_foundation.sldb"
set symbol_library   "generic.sdb"
set synthetic_library "dw_foundation.sldb"

set hdlin_translate_off_skip_text "TRUE"
set edifout_netlist_only "TRUE"
set verilogout_no_tri true

set hdlin_enable_presto_for_vhdl "TRUE"
set sh_enable_line_editing true
set sh_line_editing_mode emacs
history keep 100
alias h history

set bus_inference_style {%s[%d]}
set bus_naming_style {%s[%d]}
set hdlout_internal_busses true
define_name_rules name_rule -allowed {a-z A-Z 0-9 _} -max_length 255 -type cell
define_name_rules name_rule -allowed {a-z A-Z 0-9 _[]} -max_length 255 -type net
define_name_rules name_rule -map {{"\\*cell\\*" "cell"}}
```



Specify Libraries

- Search path
  - lists all the related directories of design and libraries.
- Link library
  - lists all technology process libraries with various timing condition.
- Target library
  - selects libraries from link_library
- Symbol library
  - defines symbols of schematic view for Design Vision
- DesignWare library
  - defines built-in operators in Synopsys

Synthesis Design Flow - Read Design

1.Method(1): Read the RTL design file

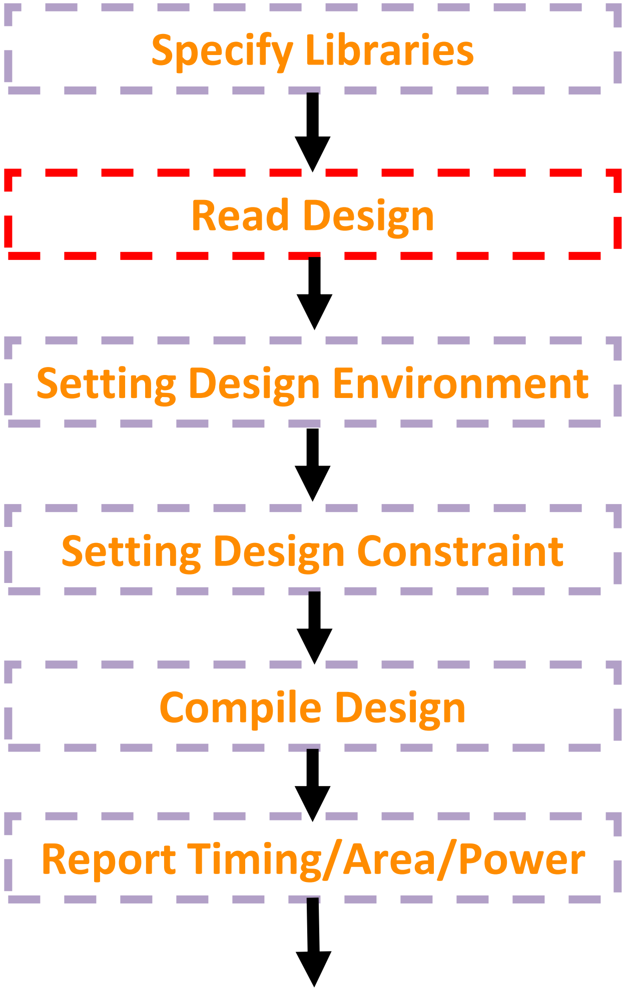
- **read_verilog design.v**

(Note: If there is parameter in your design, use Method(2))

1.Method(2): Analyze & Elaborate

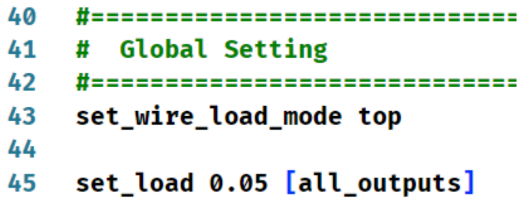
- **sh mkdir DESIGN**
- **define_design_lib DESIGN -path ./DESIGN**
- **analyze -library DESIGN -format verilog design.v**

- **elaborate top -architecture verilog -library DESIGN**



Synthesis Design Flow - Setting Design Environment

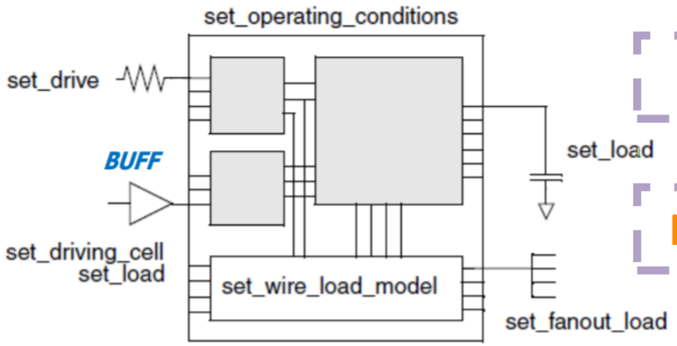
- Beware that the defaults are not realistic conditions.
  - Input drive is not infinite
  - Capacitive loading is usually not zero
  - Consider process, voltage, and temperature (PVT)

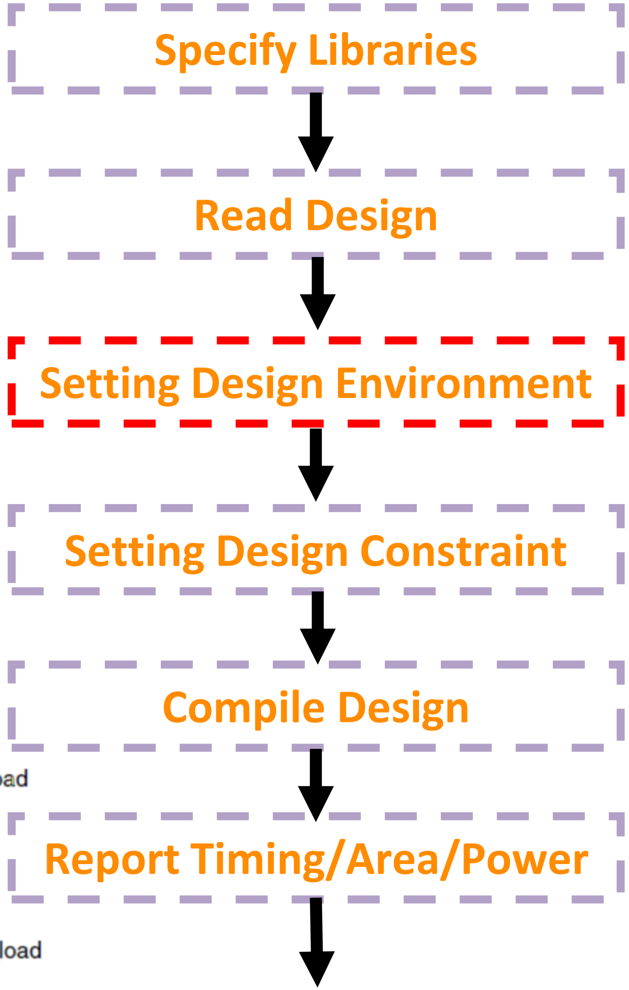


```
#=============================
#  Global Setting
#=============================
set_wire_load_mode top

set_load 0.05 [all_outputs]
```

- set_operating_conditions
- set_drive set_driving_cell
- set_load set_fanout_load
- set_wire_load_model
- set_min_library

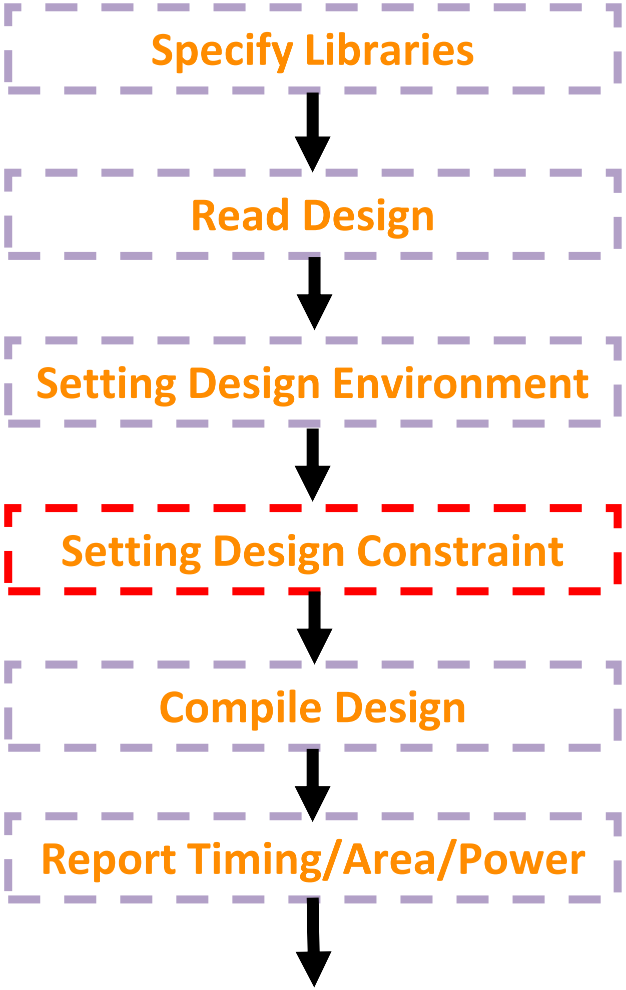




Synthesis Design Flow - Setting Design Environment

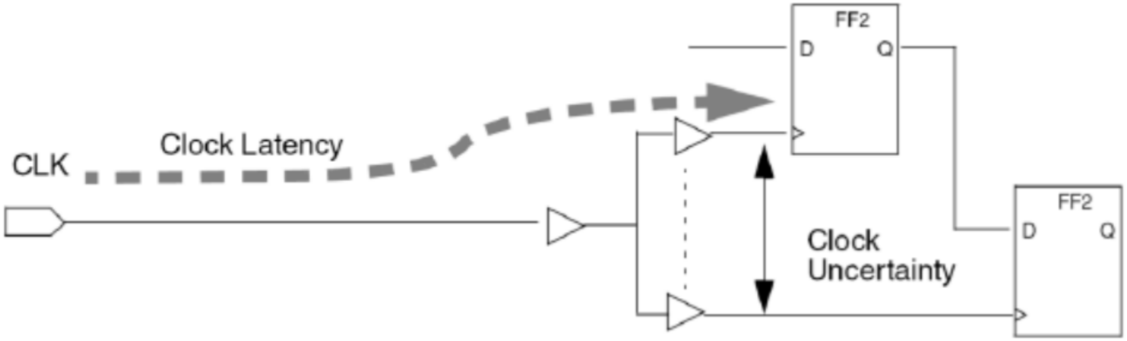
- Set design rule constraints & design optimization constraints

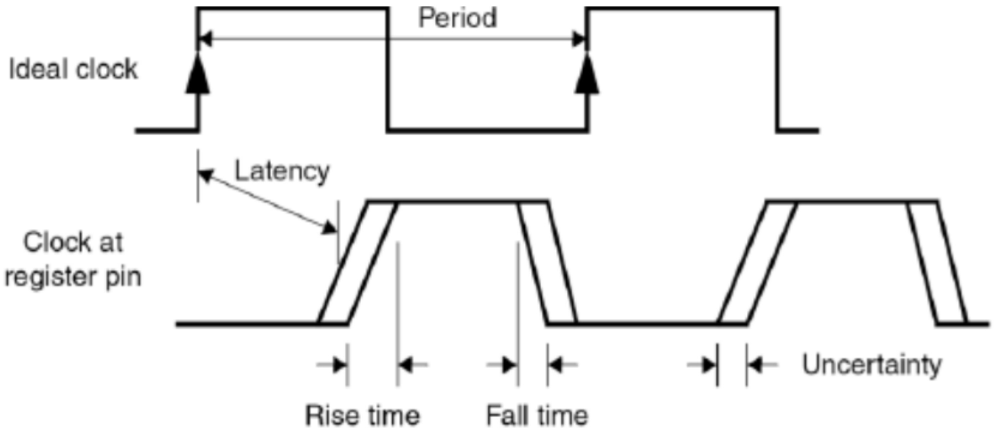
- create_clock
- set_clock_latency
- set_clock_uncertainty
- set_clock_transition
- set_input_delay
- set_output_delay
- set_max_area



Basic Clock Constraints

- Period
- Waveform
- Uncertainty
  - Skew
- Latency
  - Source latency (option)
  - Network latency
- Transition
  - Input transition
  - Clock transition





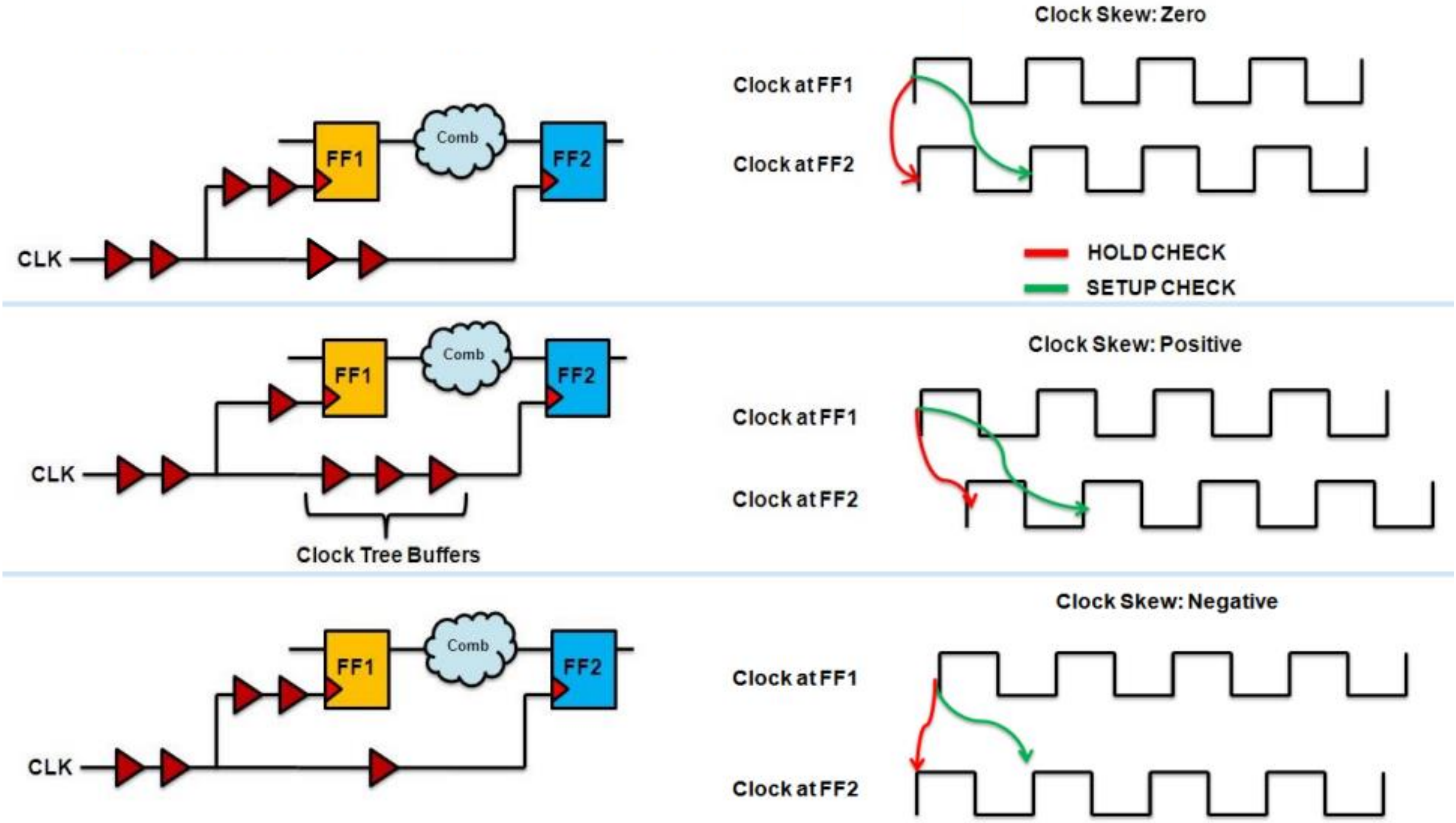
Setting Design Constraint – Create Clock

- Default clock characteristics (ideal clock):
  - 0 delay at clock port
  - 0 propagation delay
  - 0 transition delay
  - 0 uncertainty
- Create Clock

```
create_clock -period 10 [get_ports clk]
```

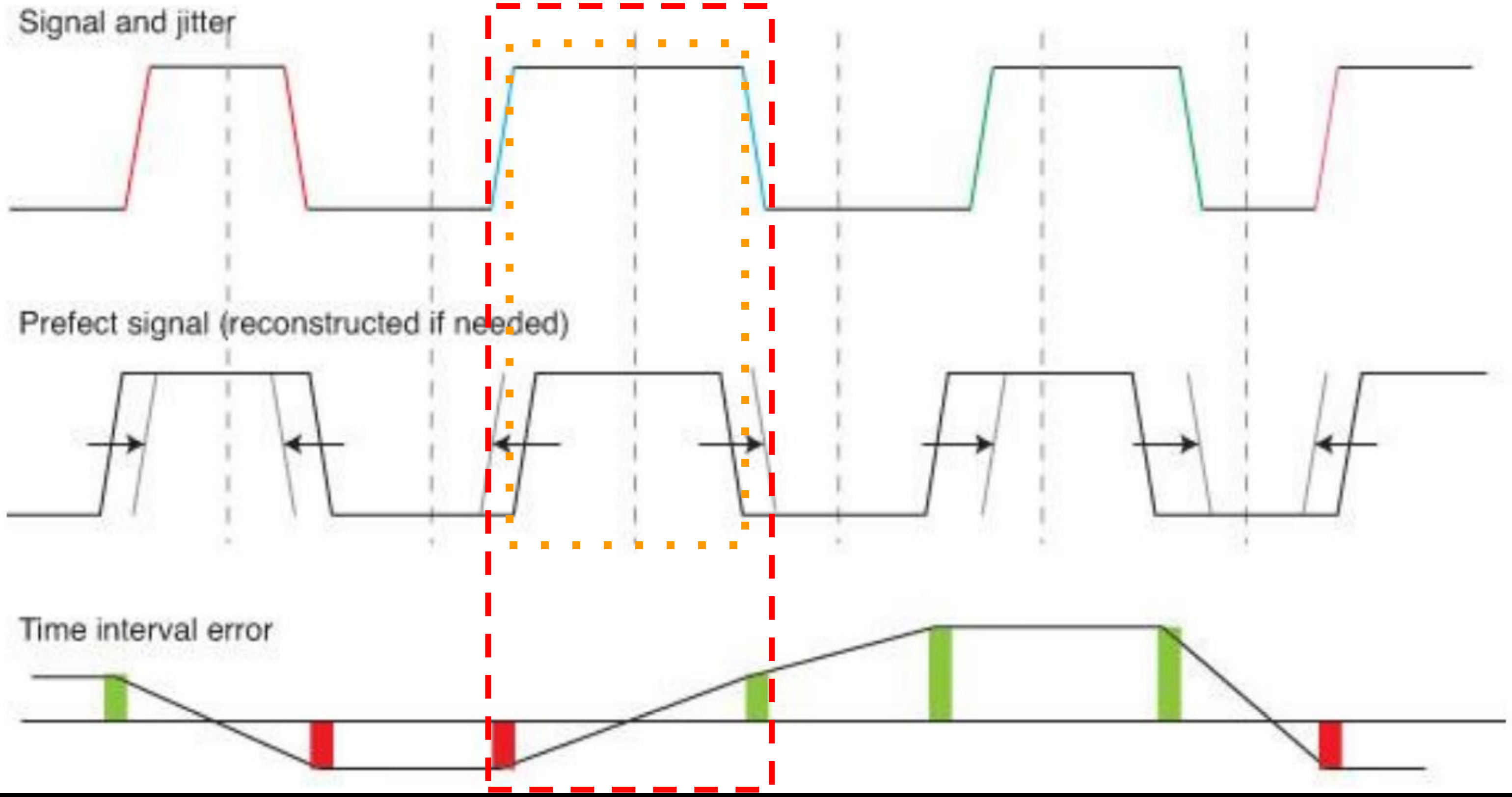
Clock Uncertainty: Skew

- The **spatial variation** in arrival time of a clock transition on an integrated circuit. (different drive & load)



Clock Uncertainty: Jitter

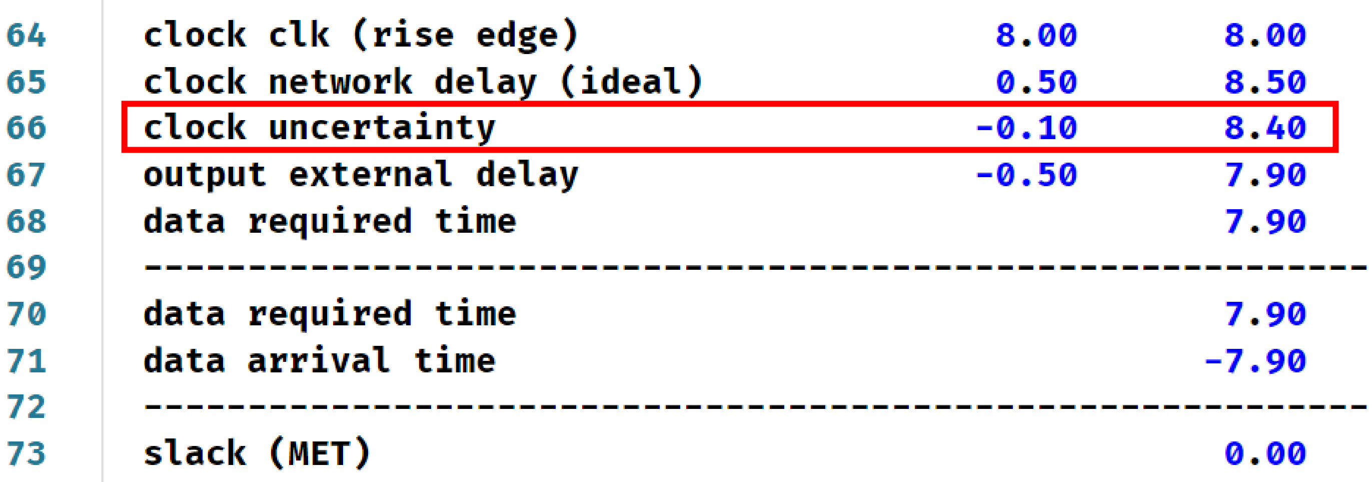
- The **temporal variation** of the clock period at a given point on an integrated circuit. (crystal oscillator variation)



Setting Clock Uncertainty

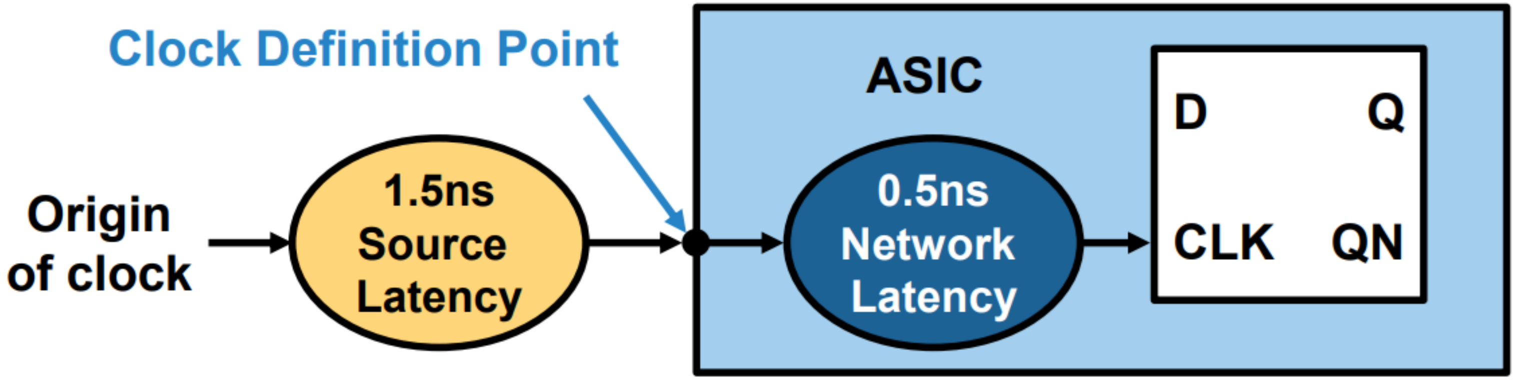
- Different clock arrival time
  - Models clock skew + jitter effects on the clock
  - After clock tree synthesis (CTS) at P&R, real propagated skew is considered!
- Experience
  - Small circuits: 0.1ns
  - Large circuits: 0.3ns

```
set_clock_uncertainty 0.1 [get_ports clk]
```



```
clock clk (rise edge)                     8.00       8.00
clock network delay (ideal)               0.50       8.50
clock uncertainty                        -0.10       8.40
output external delay                    -0.50       7.90
data required time                                   7.90
-----------------------------------------------------------
data required time                                   7.90
data arrival time                                   -7.90
-----------------------------------------------------------
slack (MET)                                          0.00
```

Setting Clock Latency



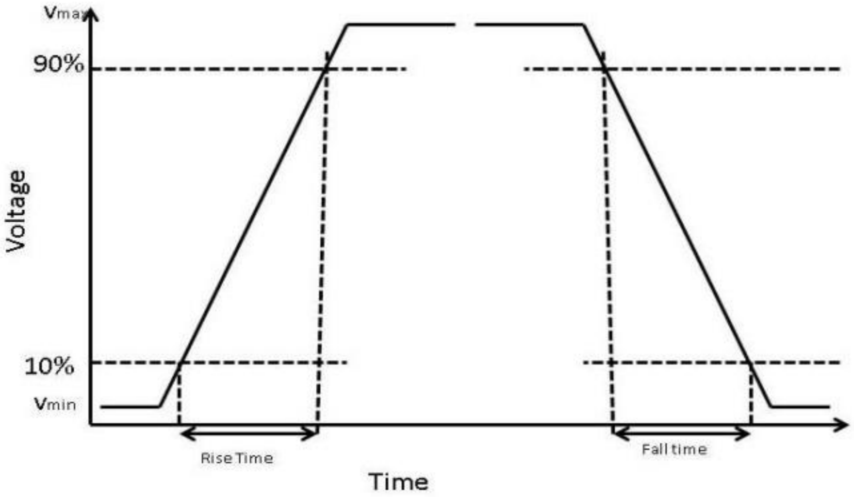
- Network latency

```
set_clock_latency 0.5       [get_clocks clk]
set_clock_latency -fall 0.5 [get_clocks clk]
set_clock_latency -rise 0.5 [get_clocks clk]
```

- Source latency (optional)

```
set_clock_latency 1.5 -source        [get_clocks clk]
set_clock_latency 1.5 -source -early [get_clocks clk]
set_clock_latency 1.5 -source -late  [get_clocks clk]
```

Setting Clock Transition

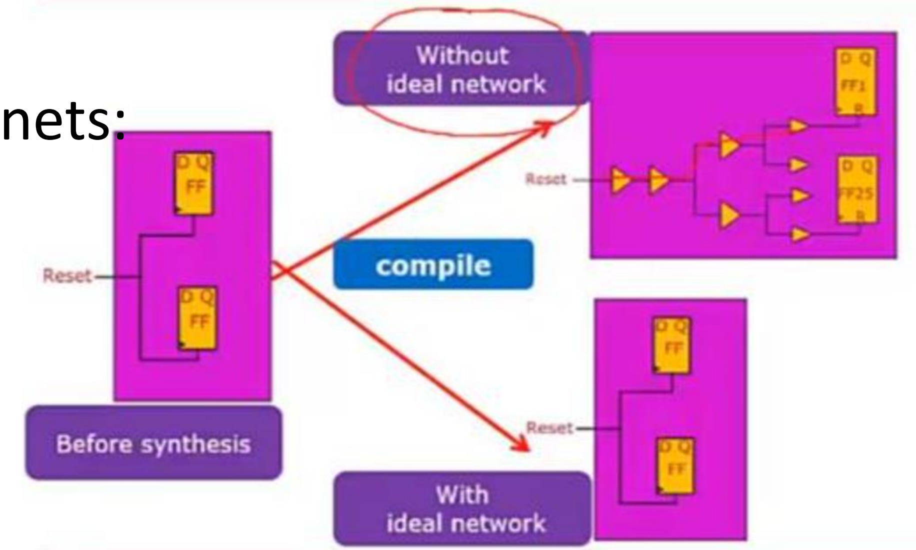


- Rise: transition time setting to only rising edges of clocks
- Fall: transition time setting to only falling edges of clocks

```
set_clock_transition delay       [get_clocks clk]
set_clock_transition delay -fall [get_clocks clk]
set_clock_transition delay -rise [get_clocks clk]
```

Set Ideal Network

- Avoid DC optimization to special nets:
  - Clock
  - Asynchronous Reset
  - High fanout nets



- Effect
  - No delay on clock network or asynchronous reset network
  - Build these clock network tree in place & route(APR)

```
set_ideal_network [get_ports clk]
```

https://www.programmersought.com/article/14253513030/

Specify Clock Constraints

- Set fix hold
  - Set_fix_hold informs compile that hold time violations of the specified clocks should be fixed.
  - To fix a hold violation requires slowing down data signals.
  - Design Compiler considers the minimum delay cost only if the set_fix_hold command is used.

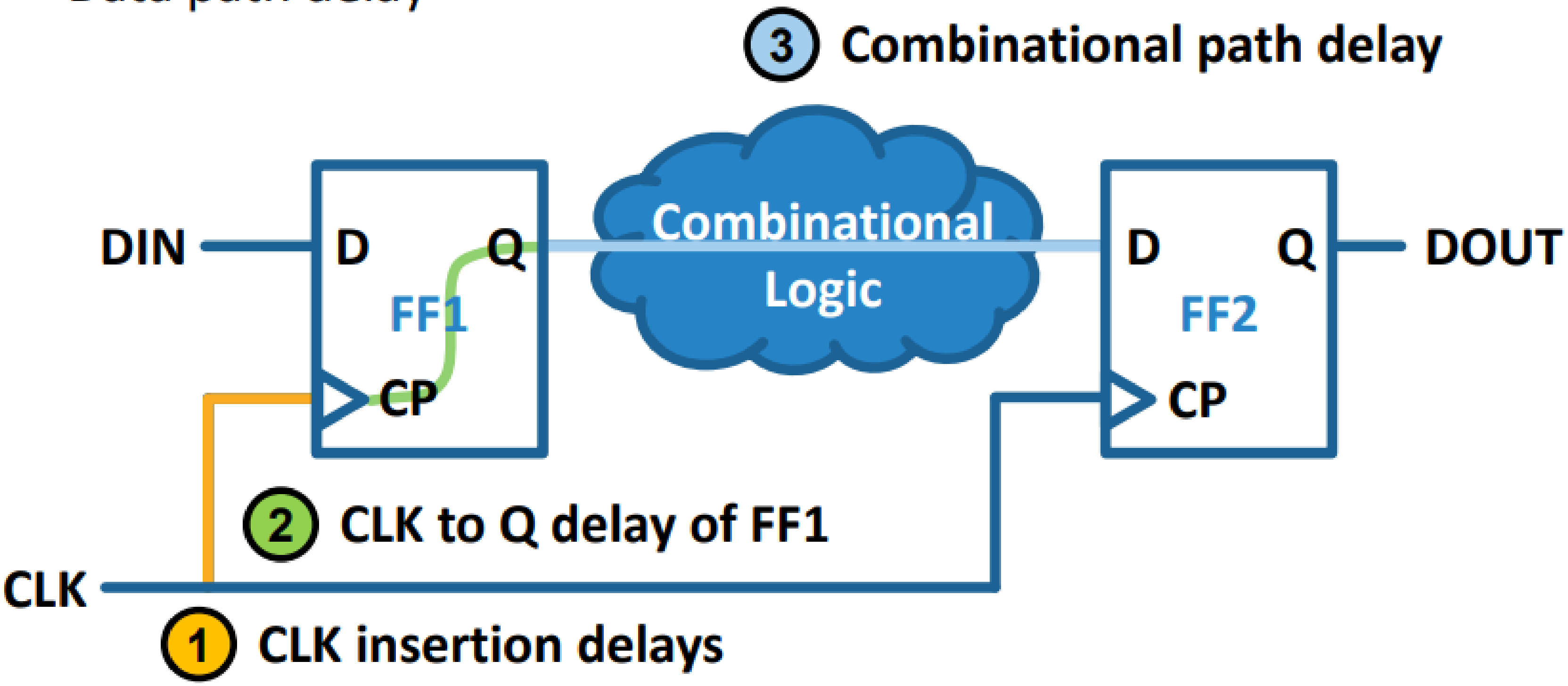
```
set_fix_hold [get_ports clk]
```

- Set dont touch network
  - Do not re-buffer clk network
  - Clock tree synthesis (CTS) is implemented in place & route stage

```
set_dont_touch_network [get_ports clk]
```

STA (Static Timing Analysis)

- Data path delay



③ Combinational path delay

② CLK to Q delay of FF1

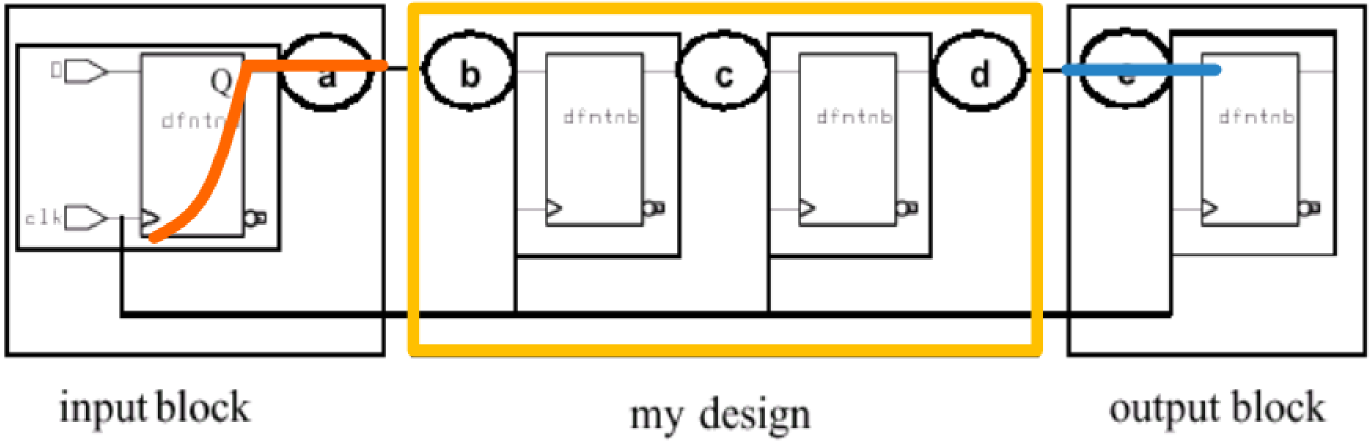
① CLK insertion delays

Actual data path delay (Data Arrival Time): ① + ② + ③

Input / Output Delay

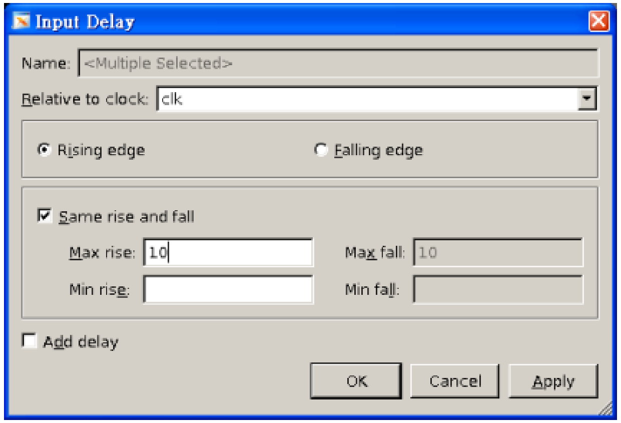
- Clock cycle >= $\text{DFF}_{\text{clk-Qdelay}} + a + b + \text{DFF}_{\text{setup}}$
- **Input delay = $\text{DFF}_{\text{clk-Qdelay}} + a$**

- Clock cycle >= $\text{DFF}_{\text{clk-Qdelay}} + d + e + \text{DFF}_{\text{setup}}$
- **Output delay = $e + \text{DFF}_{\text{setup}}$**



Setting Input Delay

- Select input ports
- Attributes / Operating Environment / Input Delay
  - Relative to clock trigger time



Input Delay

Name: <Multiple Selected>

Relative to clock: clk

● Rising edge ○ Falling edge

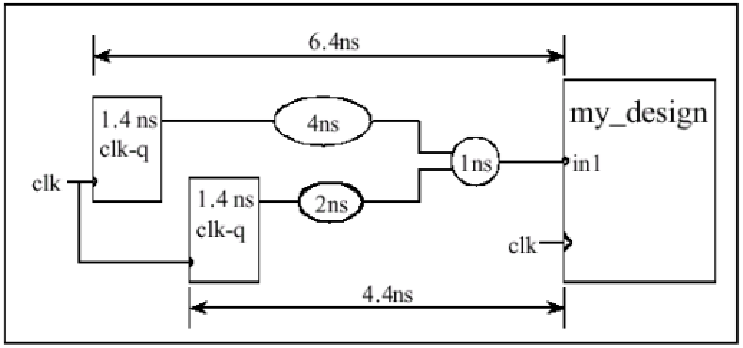
☑ Same rise and fall

Max rise: 10 Max fall: 10

Min rise: Min fall:

□ Add delay

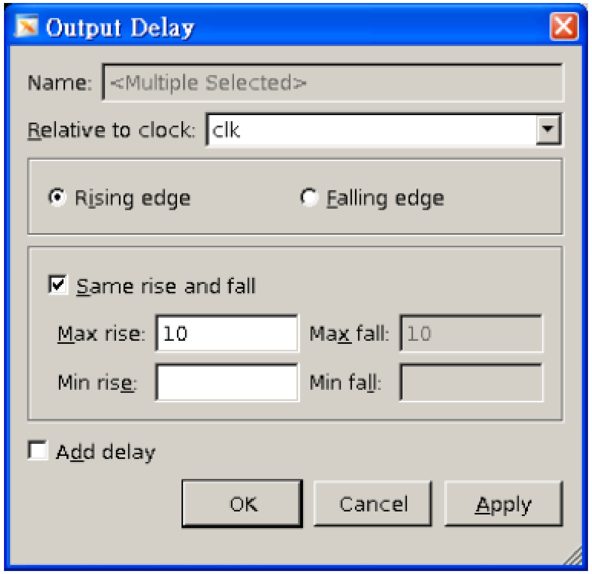
◆ Example



```
set_input_delay -clock clk -max 6.4 [get_ports in1]
set_input_delay -clock clk -min 4.4 [get_ports in1]
```

Setting Output Delay

- Select output ports
- Attributes / Operating Environment / Output Delay
  - Relative to clock trigger time



Output Delay

Name: <Multiple Selected>

Relative to clock: clk

● Rising edge ○ Falling edge

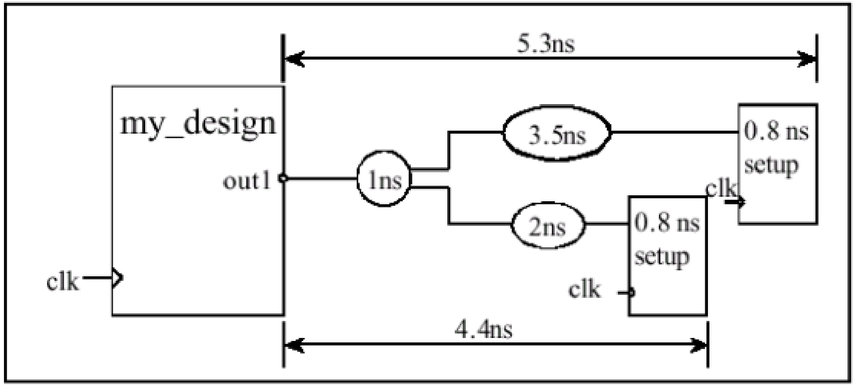
☑ Same rise and fall

Max rise: 10 Max fall: 10

Min rise: Min fall:

□ Add delay

◆ Example

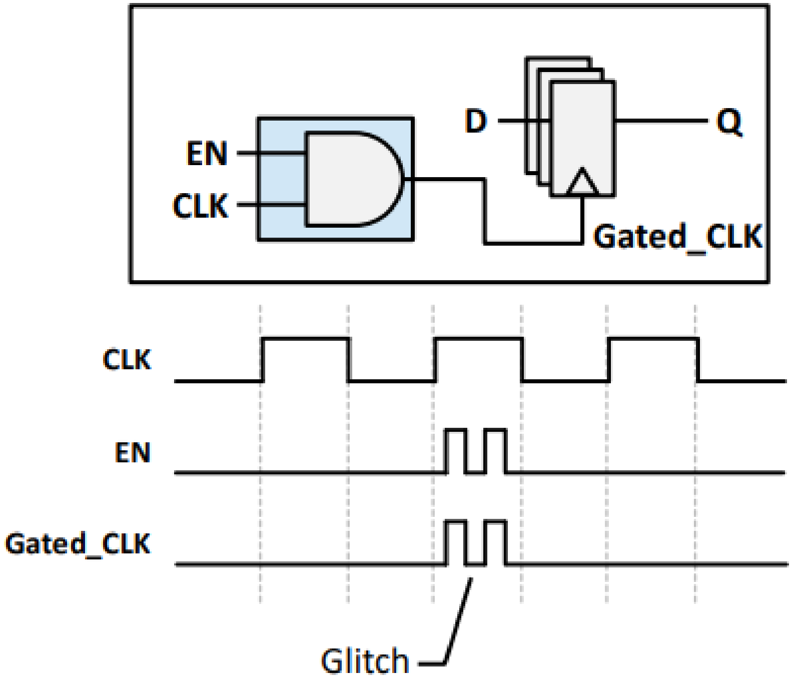


```
set_output_delay -clock clk -max 5.3 [get_ports in1]
```

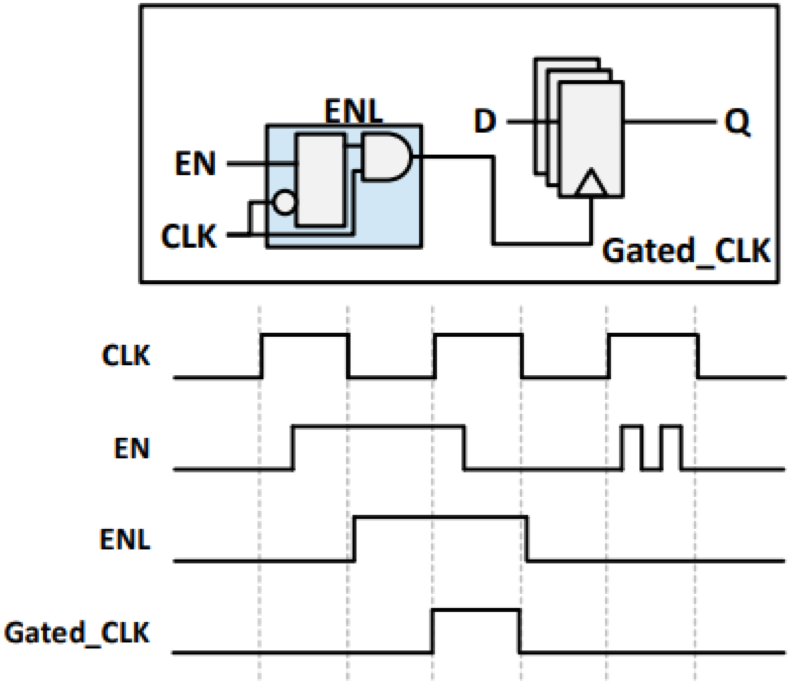
Special Circuits: Clock Gating

- Reduce dynamic power dissipation

CG with AND Gate



Glitch-Preventing CG



Clock Gating (Auto CG)

- If Statement

```
always@(posedge clk) begin
     if (enable) Q <= D_in;
     else Q <= Q;
end
```

- Conditional Assignment

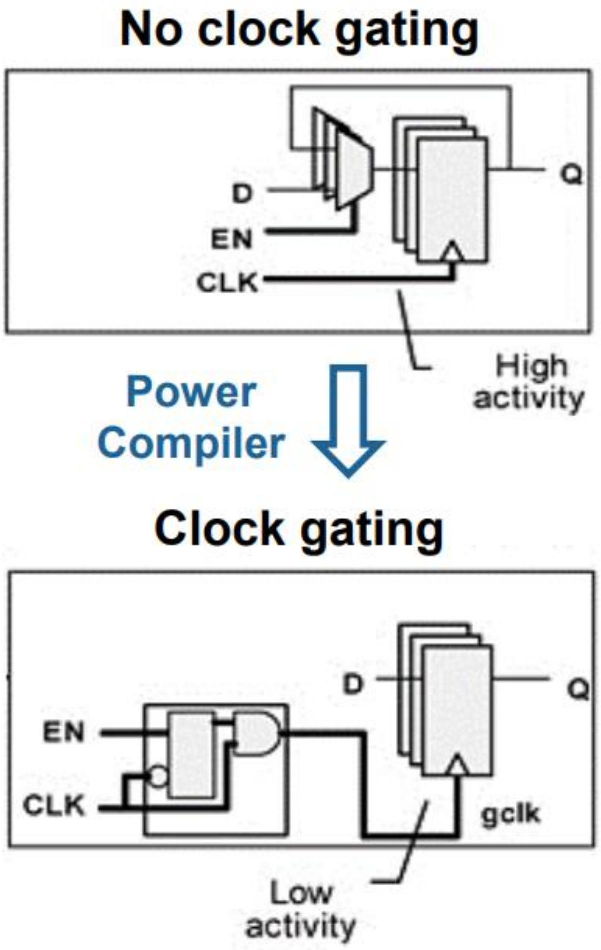
```
always@(posedge clk) begin
     Q <= (enable)? D_in:Q;
end
```

- Clock gating script style 1

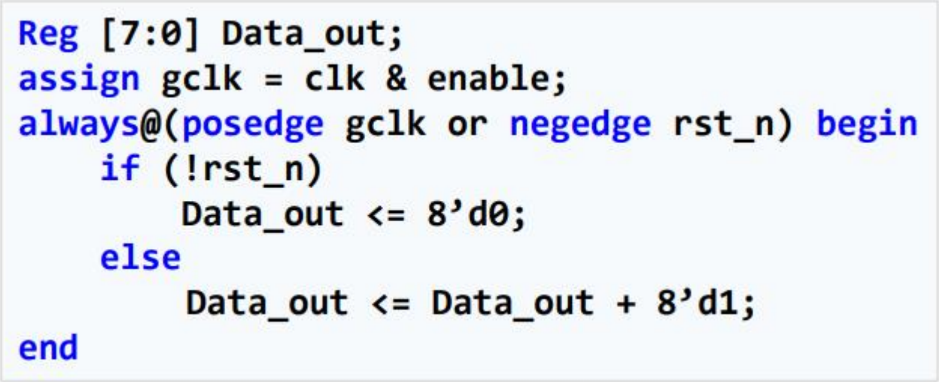
```
insert_clock_gating
compile
```

- Clock gating script style 2

```
compile -gate_clock
```



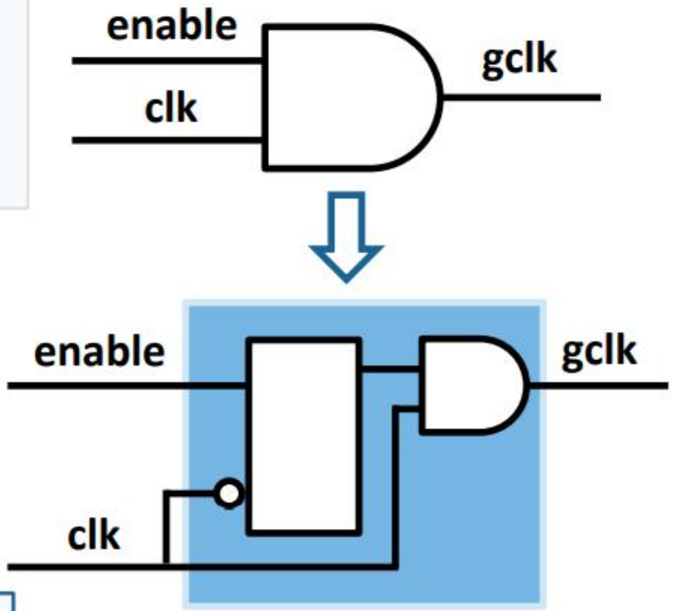
Clock Gating (Manual)



```
Reg [7:0] Data_out;
assign gclk = clk & enable;
always@(posedge gclk or negedge rst_n) begin
    if (!rst_n)
        Data_out <= 8’d0;
    else
        Data_out <= Data_out + 8’d1;
end
```

Recommended version*

Manual Clock Gating



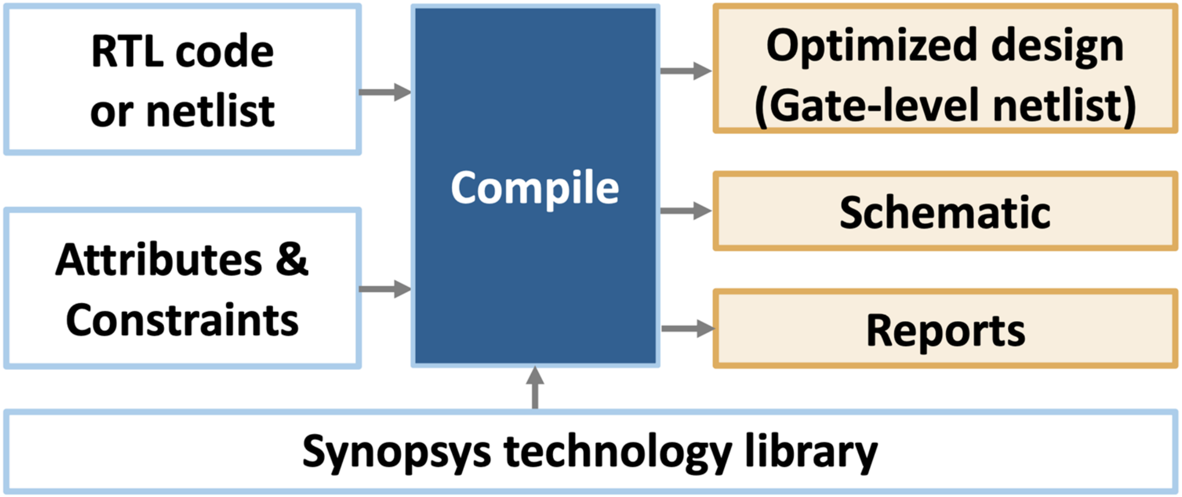
- Clock gating script

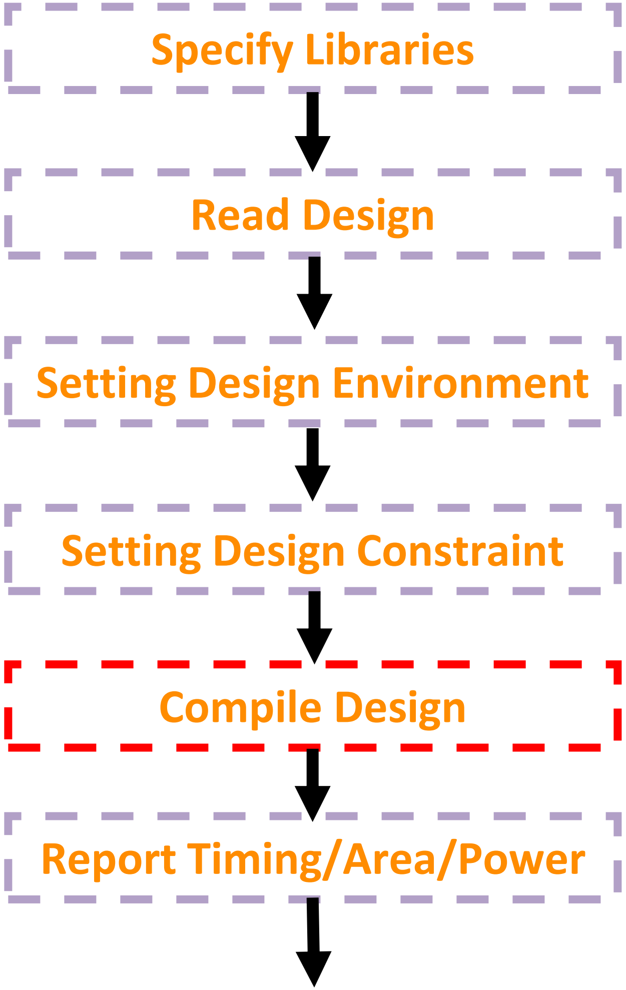
```
replace_clock_gates
compile
```

- Report clock gating

```
report_clock_gating -gating_elements
```

Synthesis Design Flow - Compile Design





Compile Design - Perform optimization

Perform optimization

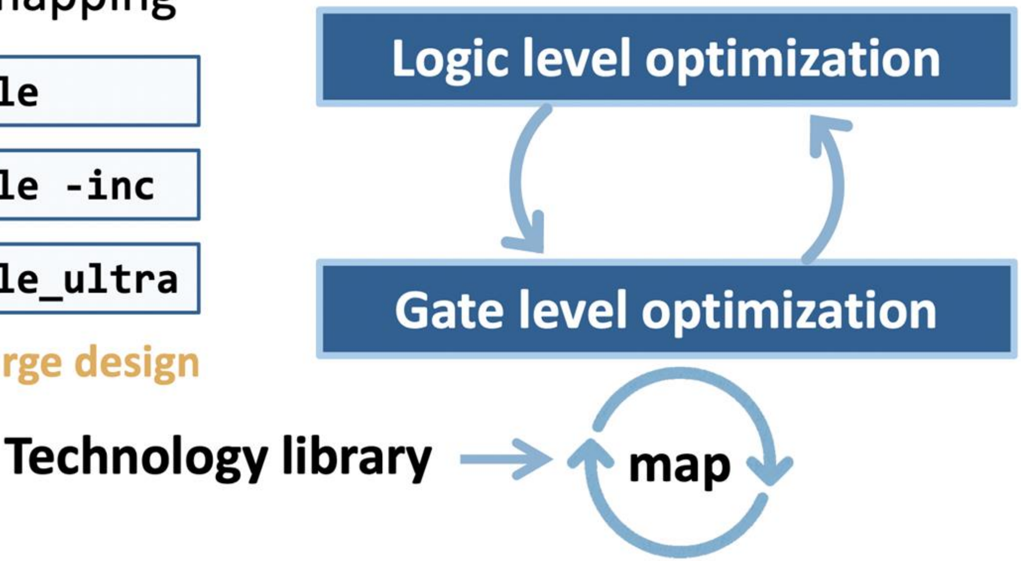
- Logic level optimization
- Gate level optimization (w/ technology library)
  - Combinational mapping
  - Sequential mapping

```
dc_shell>compile
```

```
dc_shell>compile -inc
```

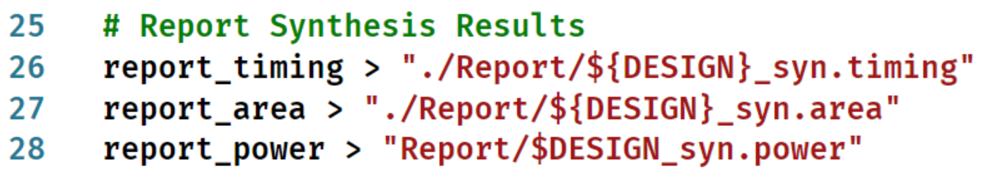
```
dc_shell>compile_ultra
```

*Careful use for large design

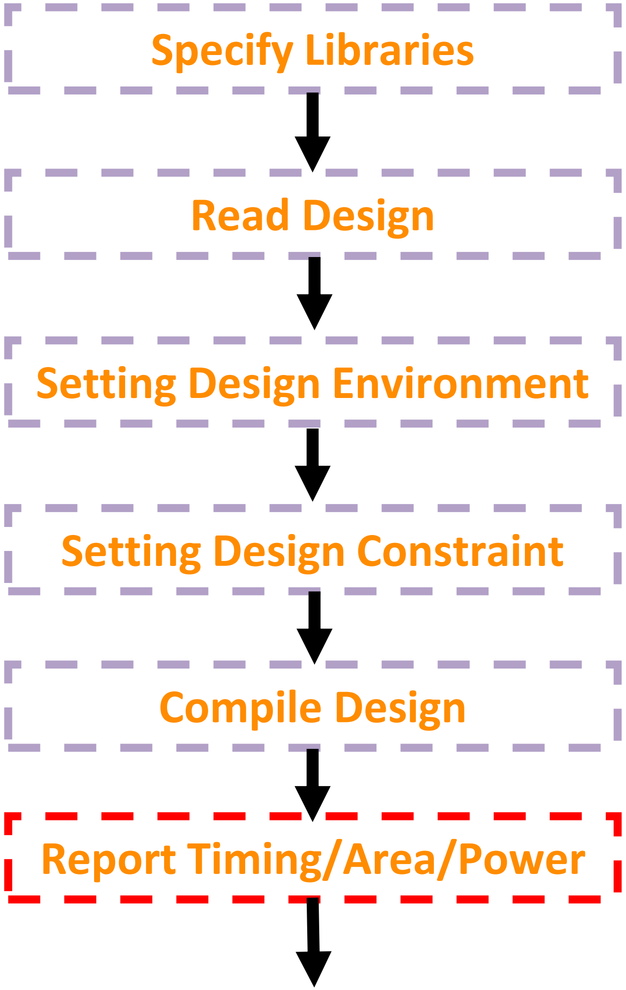


Synthesis Design Flow - Synthesis Report and Analysis

- Timing
  - The default is to display one maximum delay path
- Area
  - Area report shows the um^2 of the design
- Power
  - Report both dynamic and static power
  - PrimeTime is more accurate



```
# Report Synthesis Results
report_timing > "./Report/${DESIGN}_syn.timing"
report_area > "./Report/${DESIGN}_syn.area"
report_power > "Report/$DESIGN_syn.power"
```



Specify Libraries

Read Design

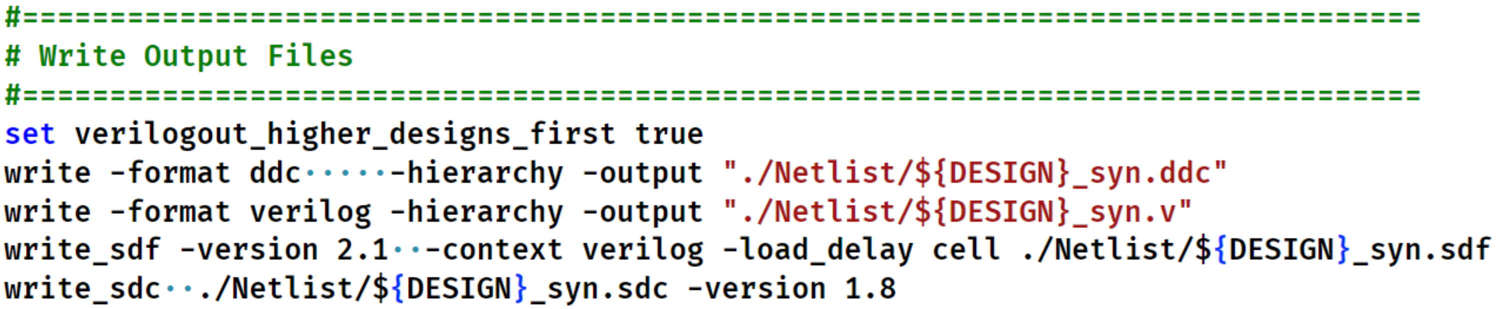
Setting Design Environment

Setting Design Constraint

Compile Design

Report Timing/Area/Power

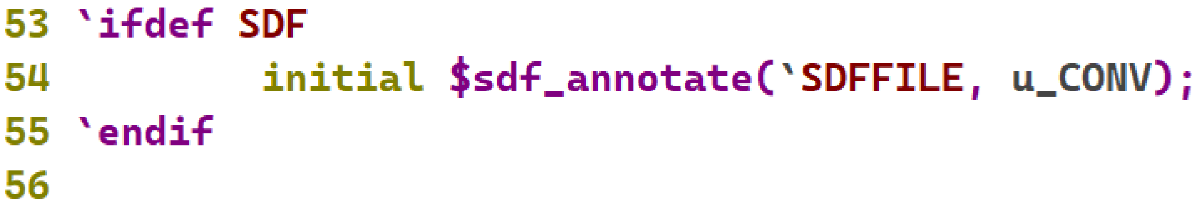
Save Design

- Synopsys Design Constraints (.sdc)
  - A format used to specify the design intent, including the timing, power and area constraints for a design.
- Standard Delay Format (.sdf)
  - Estimate timing data for each cell in the design, important in gate-sim
- Gate-Level Netlist (.v)
  - Description of the connectivity of an electronic circuit, containing all of the logic and delays of the entire circuit.
- Synopsys encrypted binary file (*.ddc)
  - 

```
#================================================================================
# Write Output Files
#================================================================================
set verilogout_higher_designs_first true
write -format ddc     -hierarchy -output "./Netlist/${DESIGN}_syn.ddc"
write -format verilog -hierarchy -output "./Netlist/${DESIGN}_syn.v"
write_sdf -version 2.1  -context verilog -load_delay cell ./Netlist/${DESIGN}_syn.sdf
write_sdc  ./Netlist/${DESIGN}_syn.sdc -version 1.8
```

Preparations for gate level simulation

- Write out gate-level netlist
  - **write -format verilog -output Netlist/$DESIGN_SYN.v -hierarchy**
- Get SDF(Standard Delay Format)
  - **write_sdf -version 2.1 -context verilog -load_delaycell ./Netlist/${DESIGN}_syn.sdf**
- Modify your testbench file to include timing delay
  - **sdf_annotate ("SDF_FILE_NAME", top_module_instance_name);**
- Gate level simulation with timing information
  - **vcs testbench.v design_syn.v -v cell_model.v -full64 -R -debug_access+all +v2k +define+SDF**



```
`ifdef SDF
        initial $sdf_annotate(`SDFFILE, u_CONV);
`endif

```

RTL Practice

- **Basic Practice**
  - Stack Module Design
  - RTL level practice
  - Deadline: Today

- **Advance Practice**
  - IC contest 2023
  - RTL + Synthesis
  - No Deadline
  - Self Practice

Reference

- 2023 Fall NTU CVSD Slides
- 2022 Spring NYCU ICLAB Slides
- 2020 TSRI Logic Synthesis Class Handouts
- 2024 Fall NTU ICD course materials