Logic Synthesis with Design Compiler

Speaker: Kuan-Ting Tu

Advisor: Prof. Shao-Yi Chien

Date: 2024/08/13

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
```

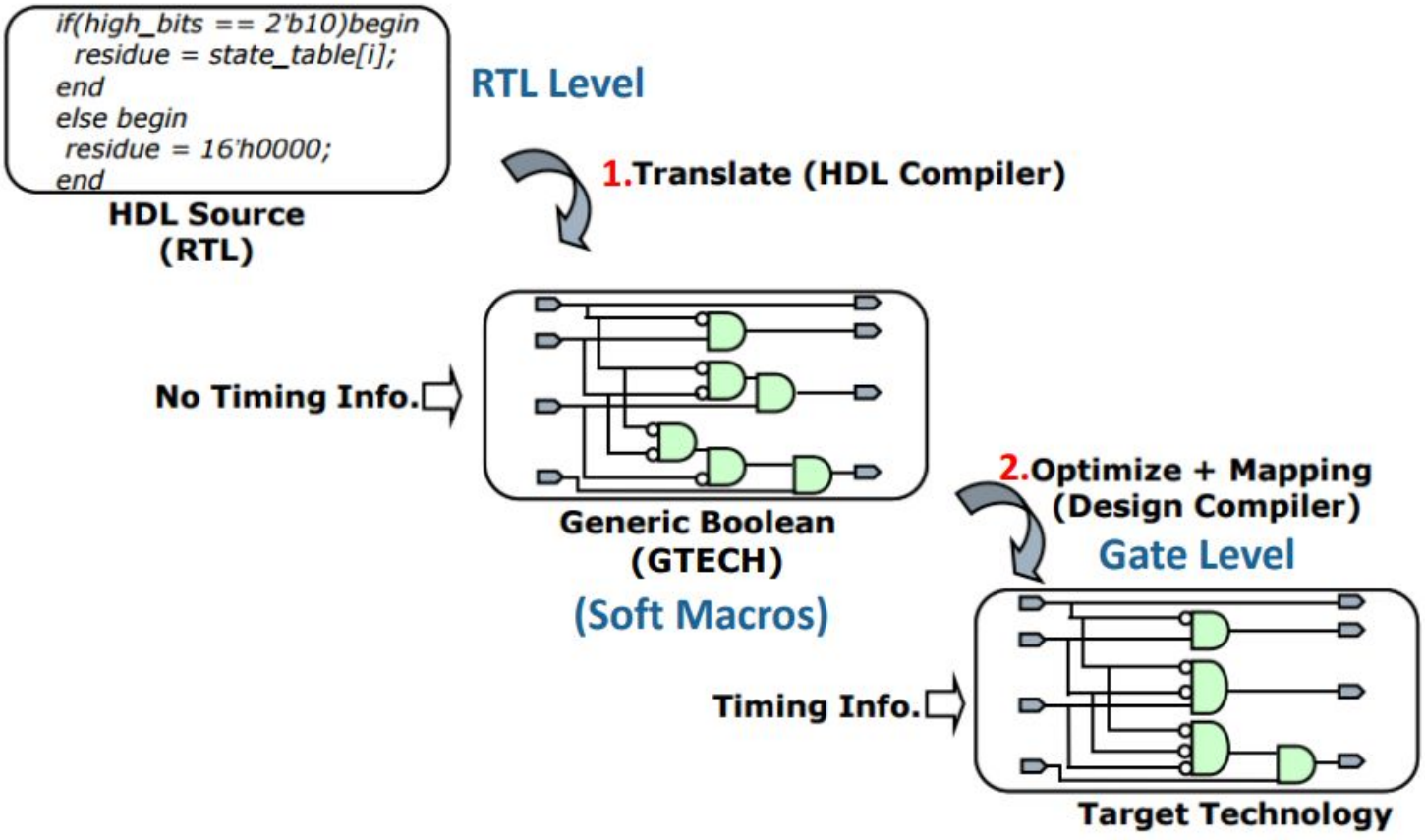
Standard Cell Lib.

Design Constraints

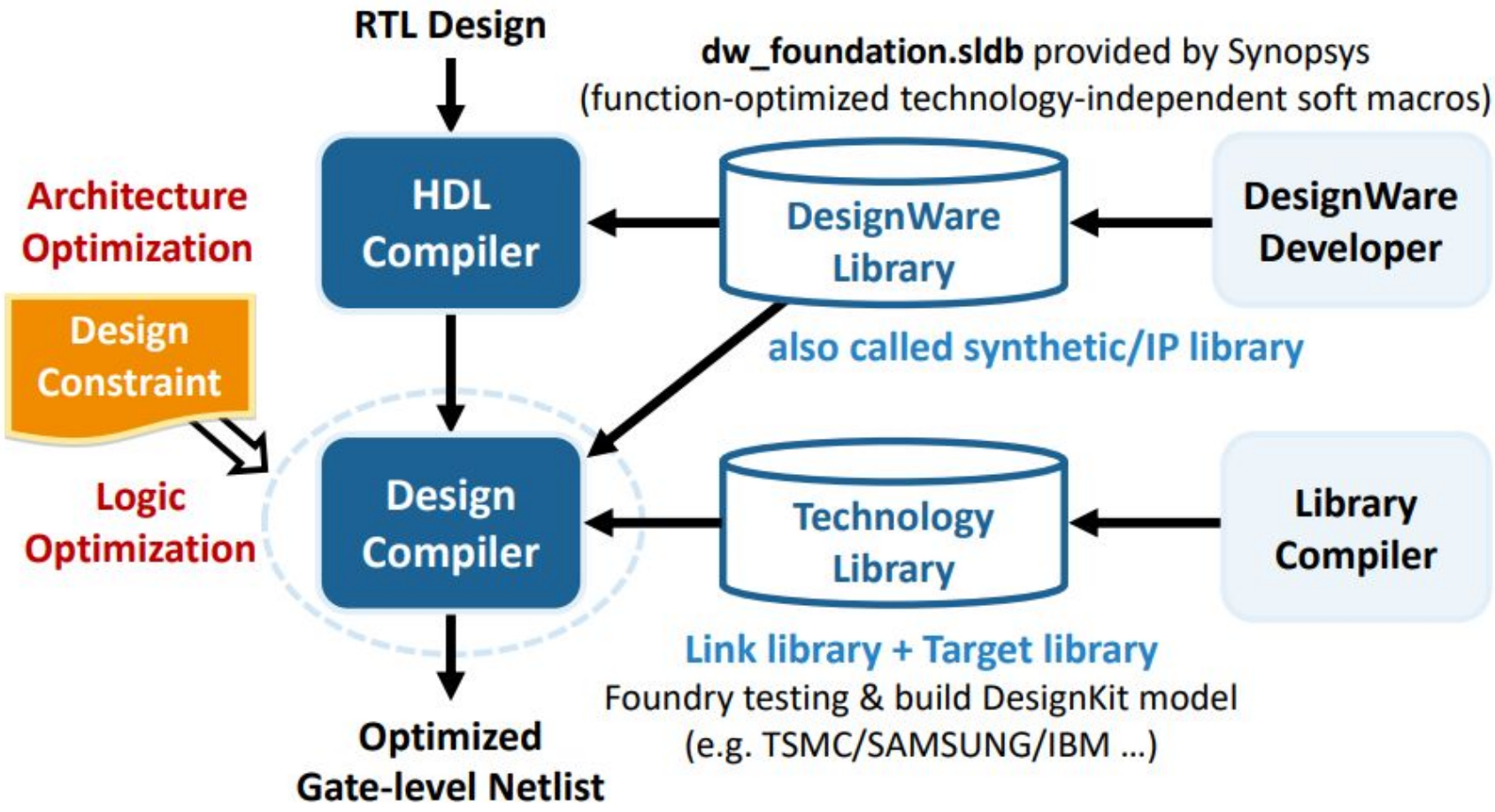
Synthesis

```
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?



What is Logic Synthesis?



Synopsys Design Compiler

SYNOPSYS®

Solutions Products Support Company

Home ▼ / Design ▼ / RTL Design and Synthesis ▼ / Design Compiler

Design Compiler

Concurrent Timing, Area, Power, and Test Optimization

Design Compiler® RTL synthesis solution enables users to meet today's design challenges with concurrent optimization of timing, area, power and test. Design Compiler includes innovative topographical technology that enables a predictable flow resulting in faster time to results. Topographical technology provides timing and area prediction within 10% of the results seen post-layout enabling designers to reduce costly iterations between synthesis and physical implementation. Design Compiler also includes a scalable infrastructure that delivers 2X faster runtime on quad-core platforms.

Design Compiler is the core of Synopsys' comprehensive RTL synthesis solution, including Power Compiler™, DesignWare®, PrimeTime®, and DFTMAX™. Design Compiler NXT is also available and includes includes best-in-class quality-of-results, congestion prediction and alleviation capabilities, physical viewer, and floorplan exploration. Additionally Design Compiler NXT produces physical guidance to IC Compiler, place-and-route solution for tighter correlation to layout and faster placement runtime.

Download Datasheet

NEWS

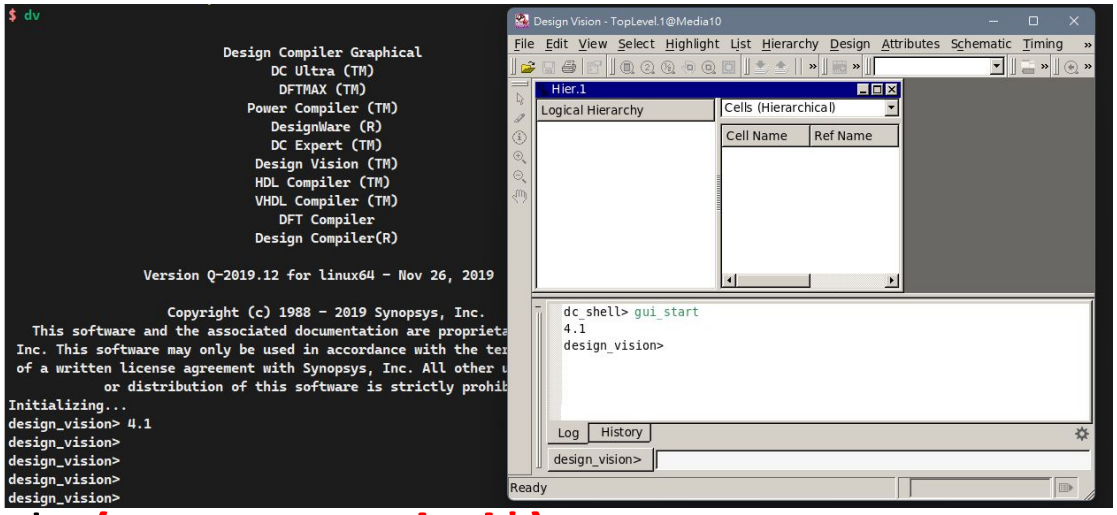
Synopsys Digital and Custom Design Platforms Certified for TSMC's Latest 3nm Process Technology

Synopsys Enables First-Pass Silicon Success for Early Adopters of Next-Generation Armv9 Architecture-based SoCs

Synopsys and Samsung Foundry Collaboration Delivers Optimized Reference Methodology for High-Performance Compute Designs

How to launch Design Compiler

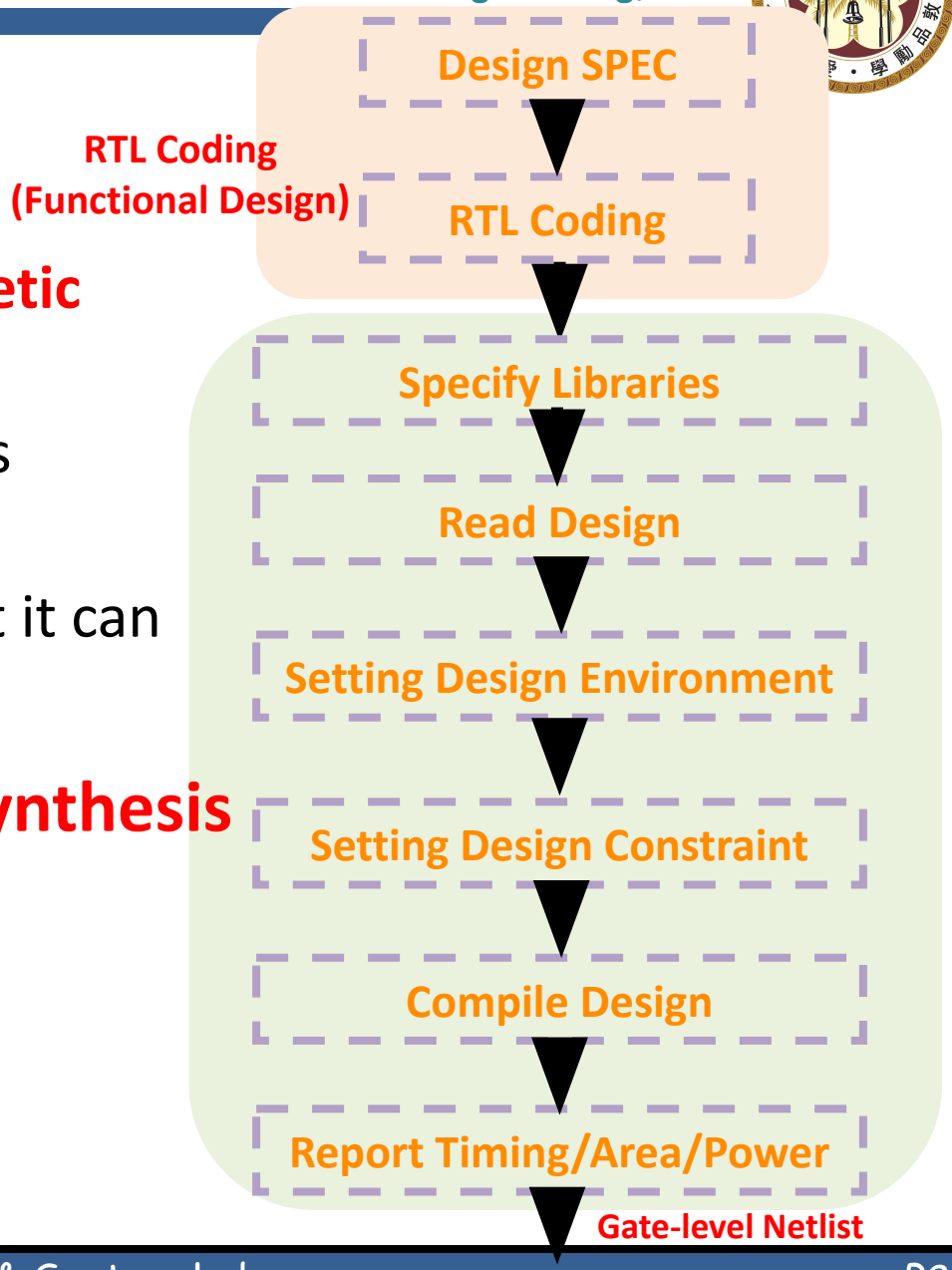
- GUI Mode
  - **Command: dv**



- 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).

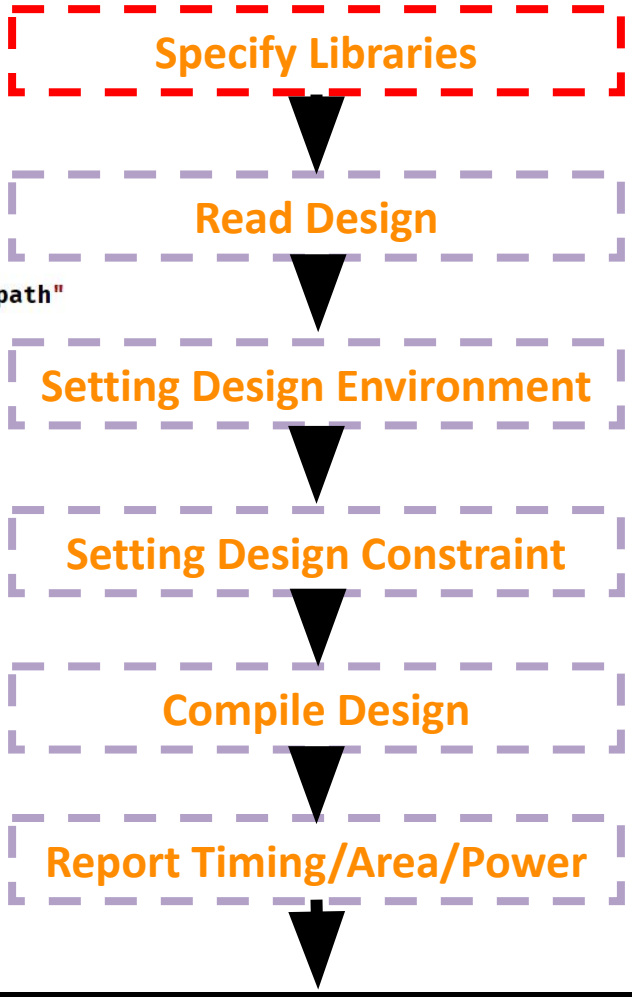


Synthesis Design Flow - Specify Libraries

- Set up the .synopsys_dc.setup file.

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

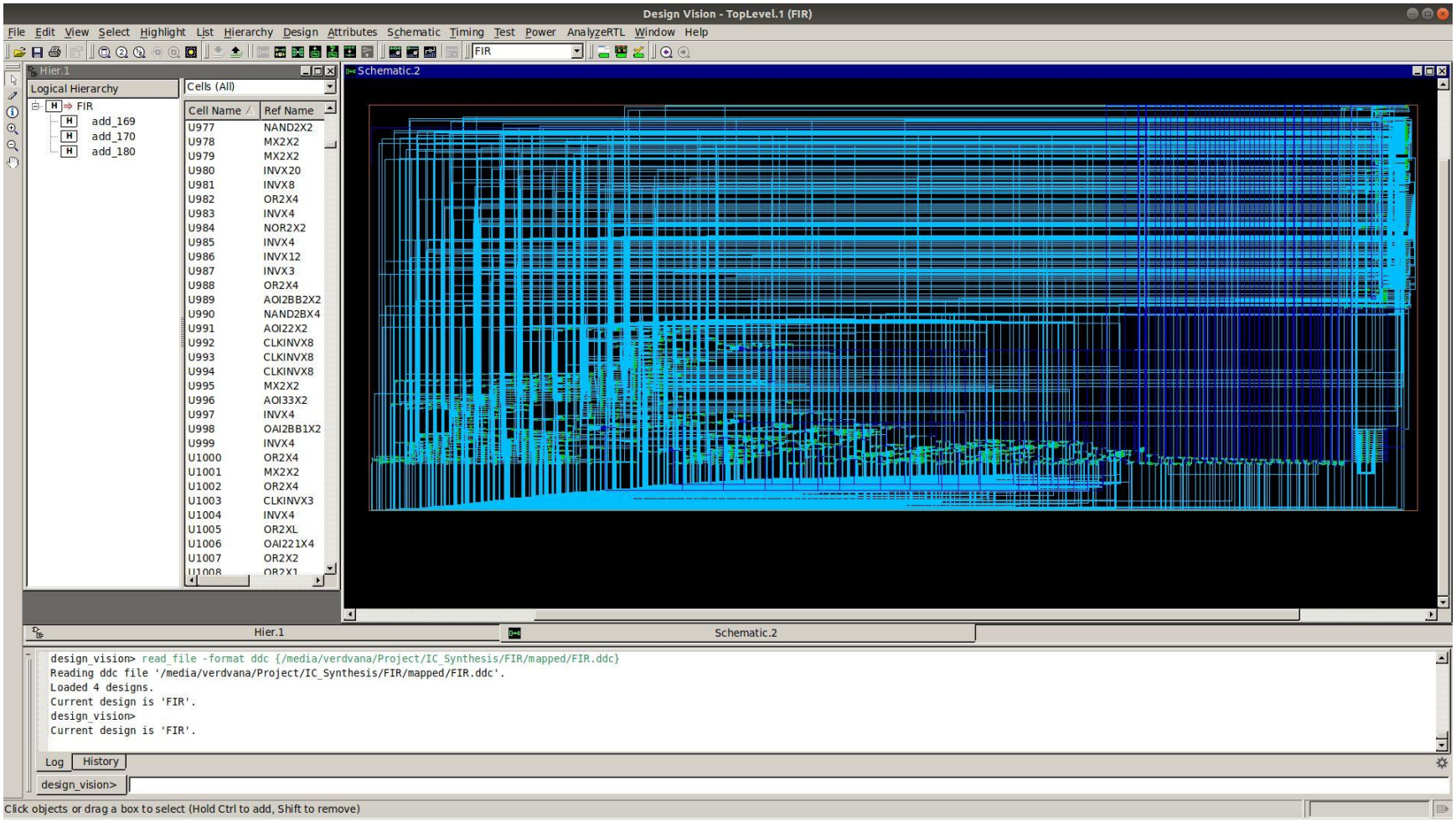
```
1  set company "CIC"
2  set designer "Student"
3  set search_path       "/cad/designkit/CBDK_IC_Contest_v2.1/SynopsysDC/db  $search_path"
4  set target_library    "slow.db fast.db"
5  set link_library      "* $target_library dw_foundation.sldb"
6  set symbol_library    "generic.sdb"
7  set synthetic_library "dw_foundation.sldb"
8
9  set hdlin_translate_off_skip_text "TRUE"
10 set edifout_netlist_only "TRUE"
11 set verilogout_no_tri true
12
13 set hdlin_enable_presto_for_vhdl "TRUE"
14 set sh_enable_line_editing true
15 set sh_line_editing_mode emacs
16 history keep 100
17 alias h history
18
19 set bus_inference_style {%s[%d]}
20 set bus_naming_style {%s[%d]}
21 set hdlout_internal_busses true
22 define_name_rules name_rule -allowed {a-z A-Z 0-9 _} -max_length 255 -type cell
23 define_name_rules name_rule -allowed {a-z A-Z 0-9 _[]} -max_length 255 -type net
24 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

Design Vision - Schematic

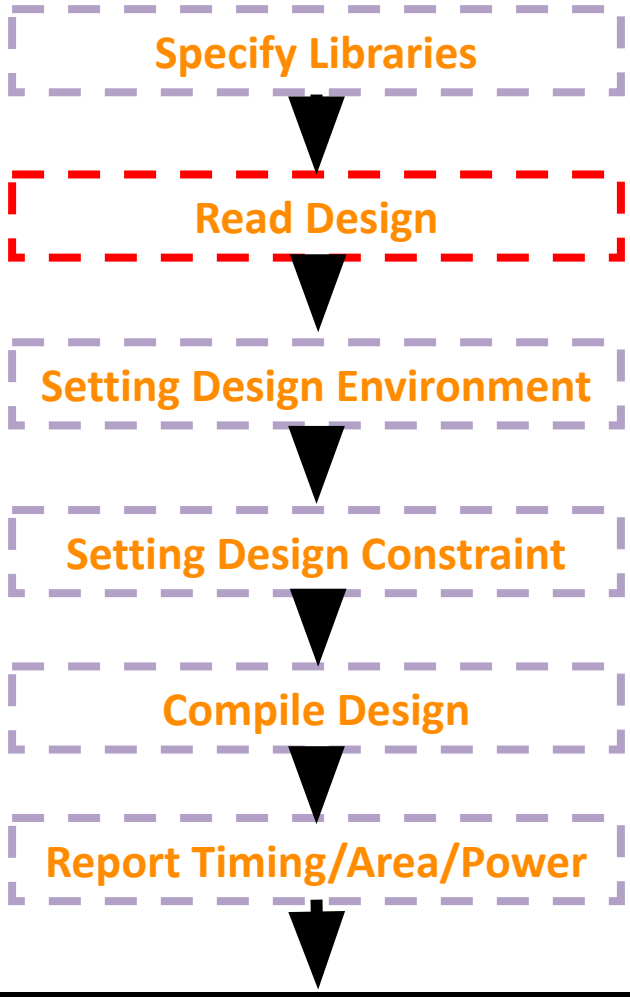


Synthesis Design Flow - Read Design

1. Method(1): Read the RTL design file
   - **read_verilog design.v**

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

2. 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**

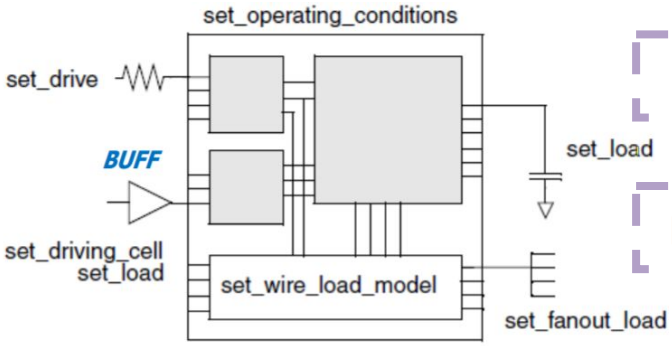


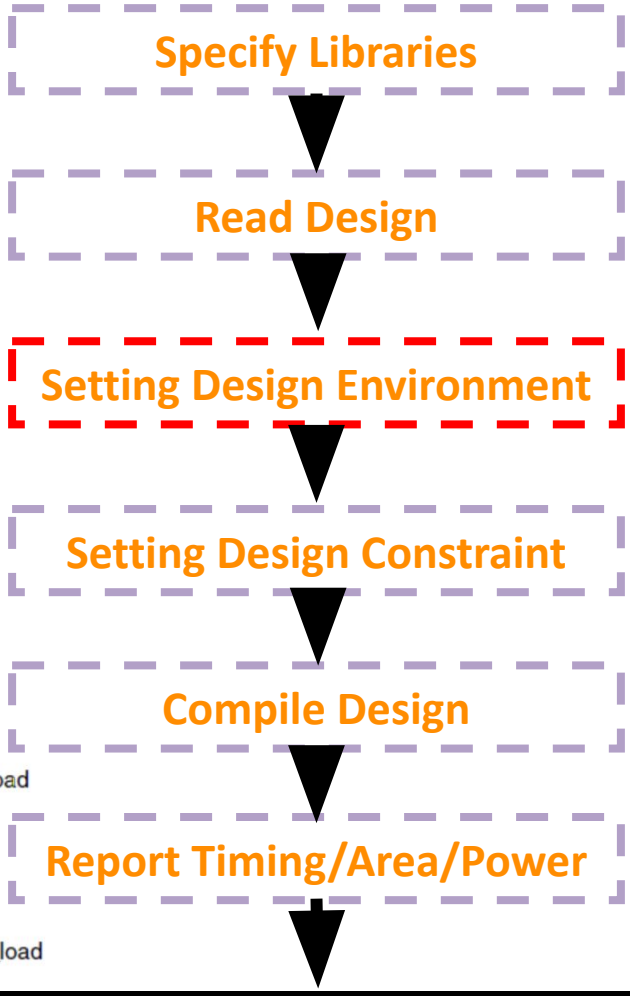
Synthesis Design Flow - Setting Design Environment

- 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) variation

```
40  #============================
41  #  Global Setting
42  #============================
43  set_wire_load_mode top
44
45  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

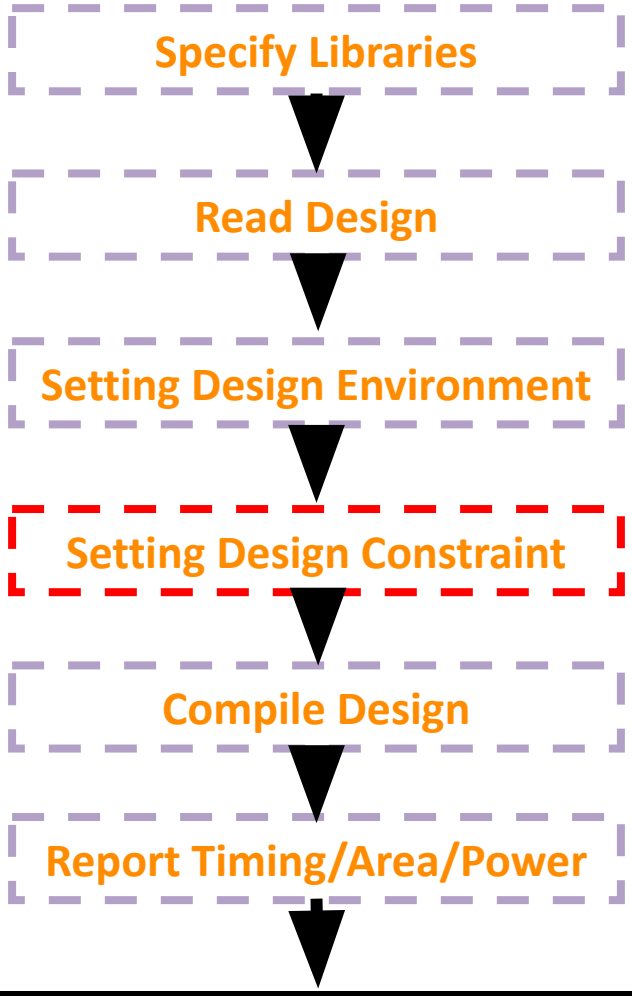




Synthesis Design Flow - Setting Design Environment

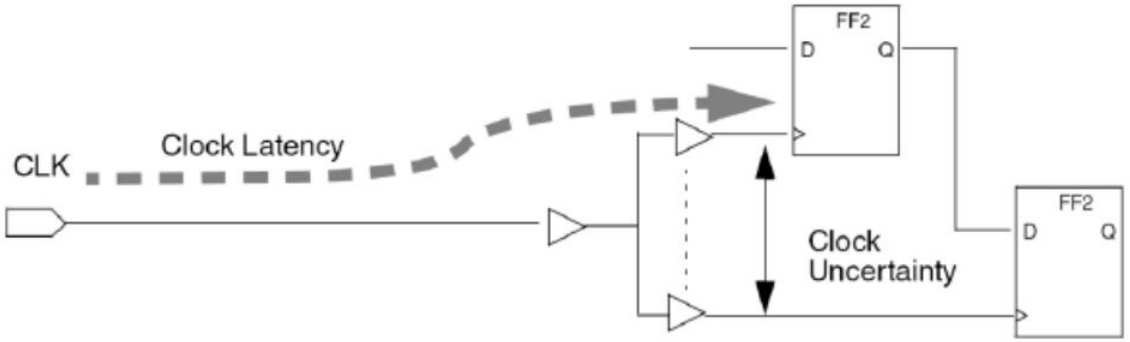
- Set design rule constraints & design optimization constraints

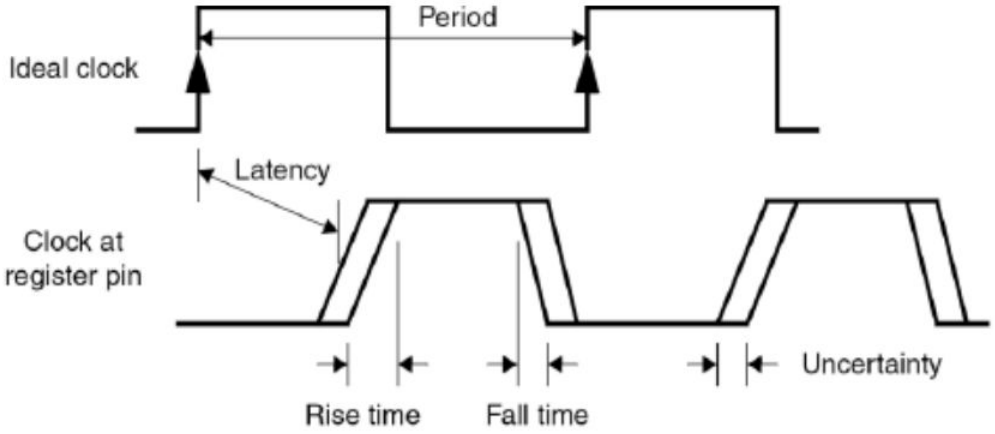
- create_clock
- set_clock_latency
- set_clock_uncertainty
- set_clock_transition
- set_input_delay
- set_output_delay
- set_max_area



Basic Clock Constraints

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





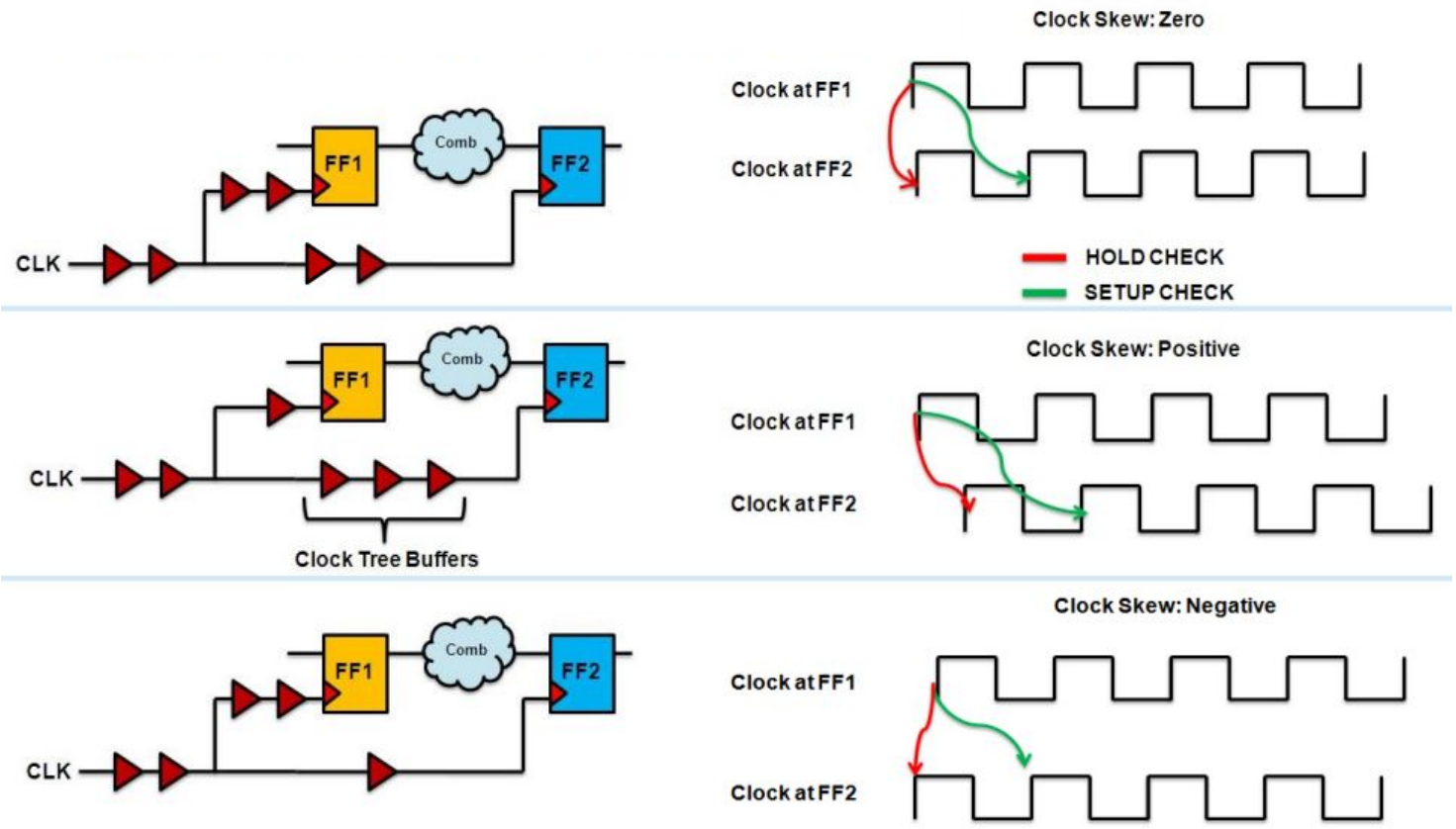
Setting Design Constraint – Create Clock

- 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]
```

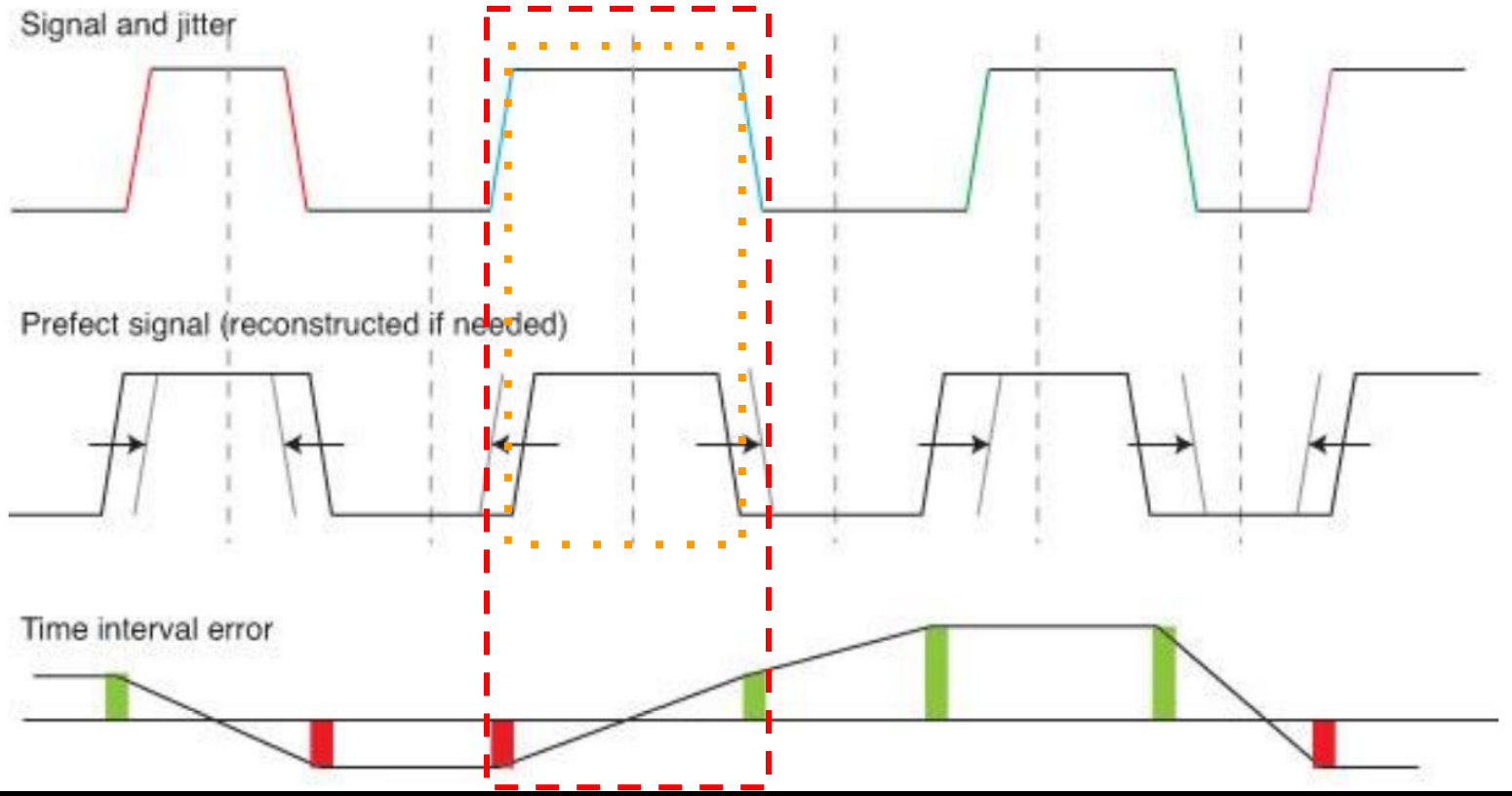
Clock Uncertainty: Skew

- 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)



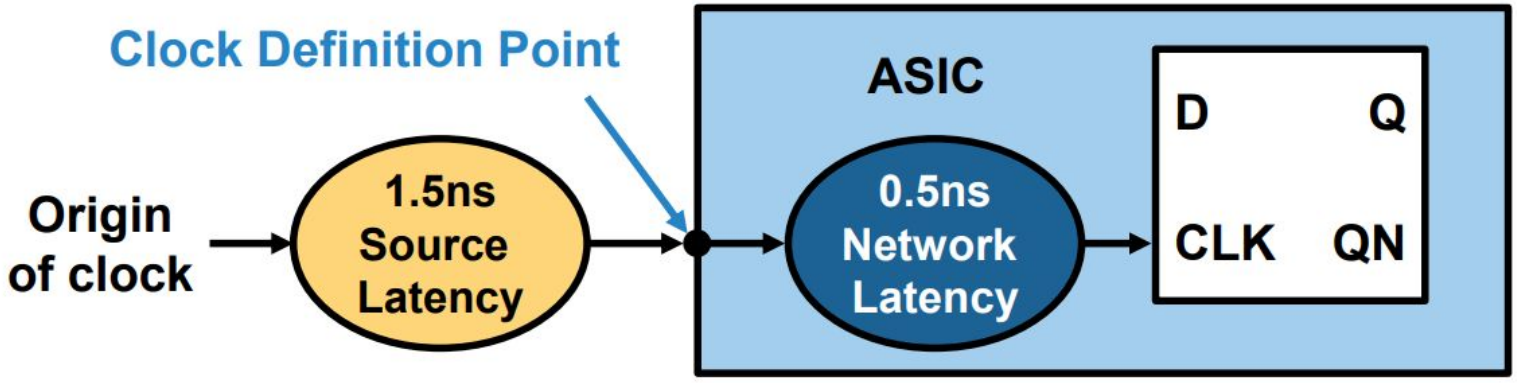
Setting Clock Uncertainty

- 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]
```

```
64   clock clk (rise edge)                    8.00        8.00
65   clock network delay (ideal)              0.50        8.50
66   clock uncertainty                       -0.10        8.40
67   output external delay                   -0.50        7.90
68   data required time                                   7.90
69   ---------------------------------------------------------
70   data required time                                   7.90
71   data arrival time                                   -7.90
72   ---------------------------------------------------------
73   slack (MET)                                          0.00
```

Setting Clock Latency



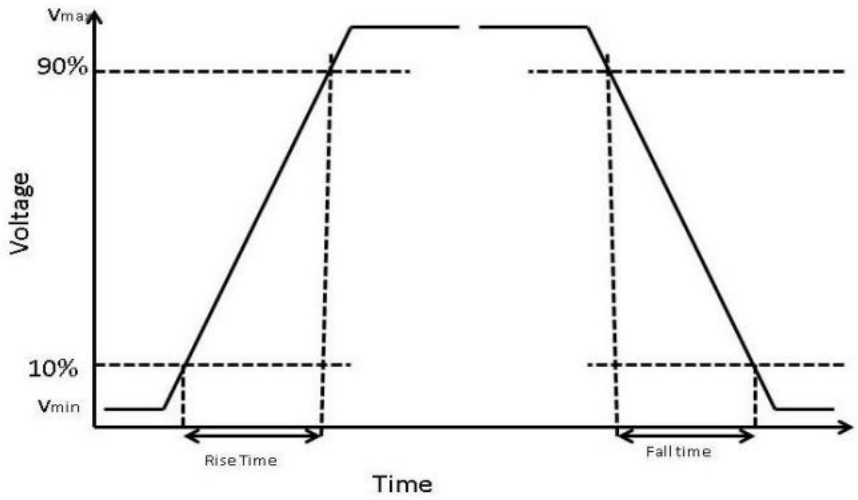
- Network latency

```
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

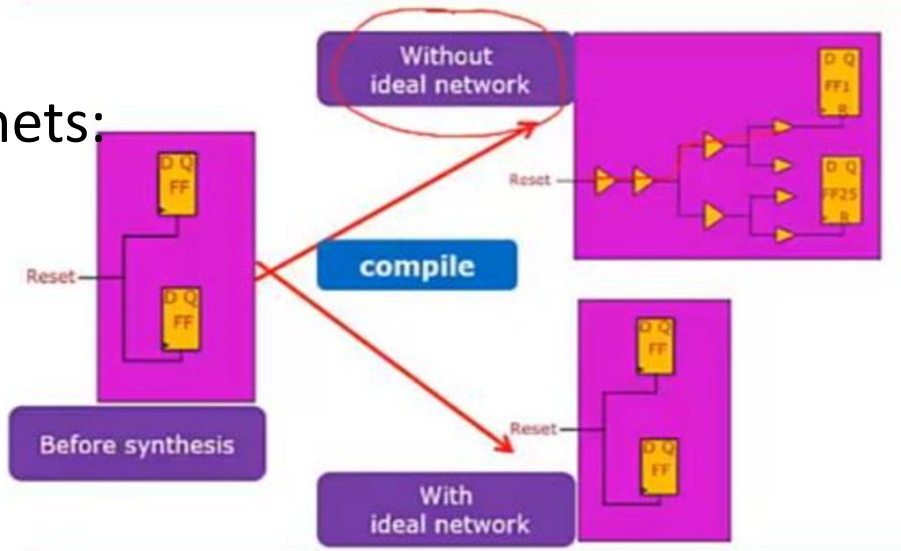


- Rise: transition time setting to only rising edges of clocks
- Fall: transition time setting to only falling edges of clocks

```
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.

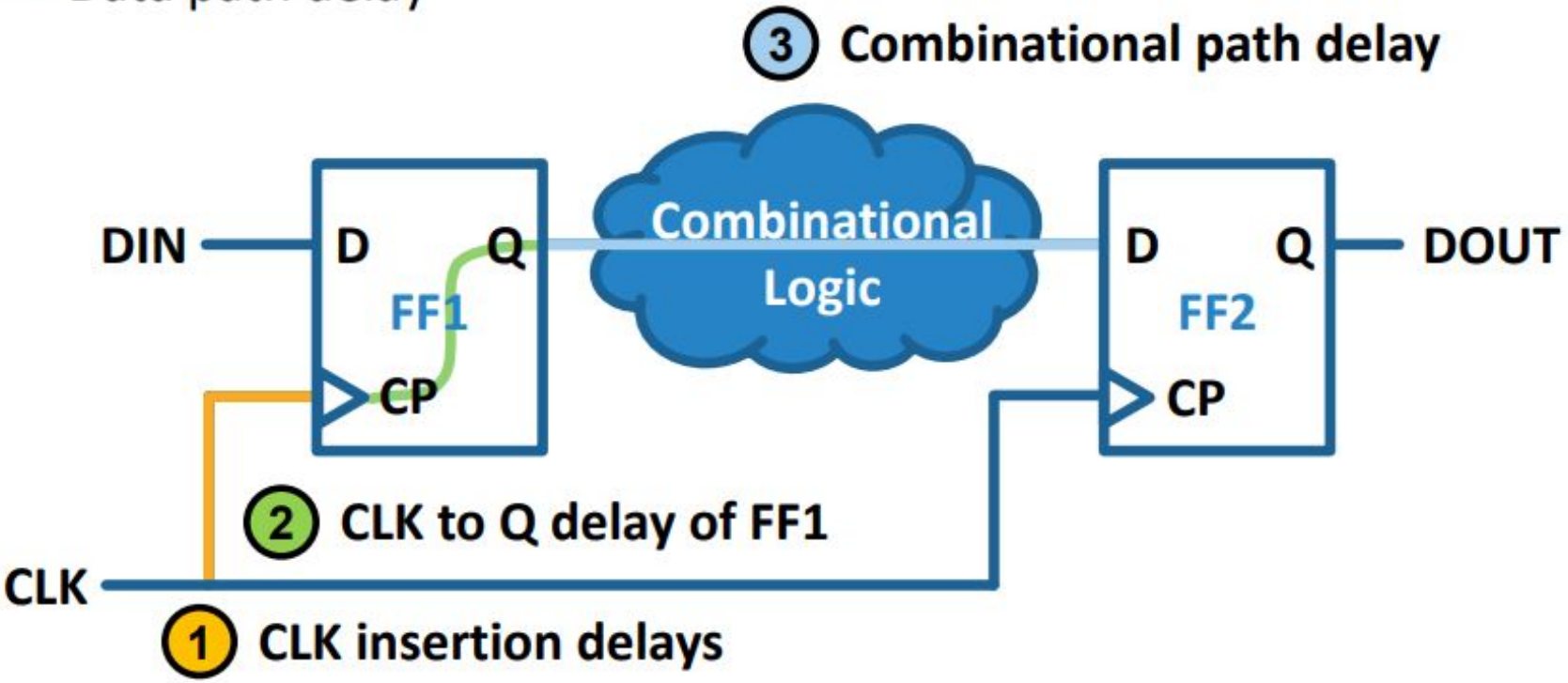
```
set_fix_hold [get_ports clk]
```

- 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

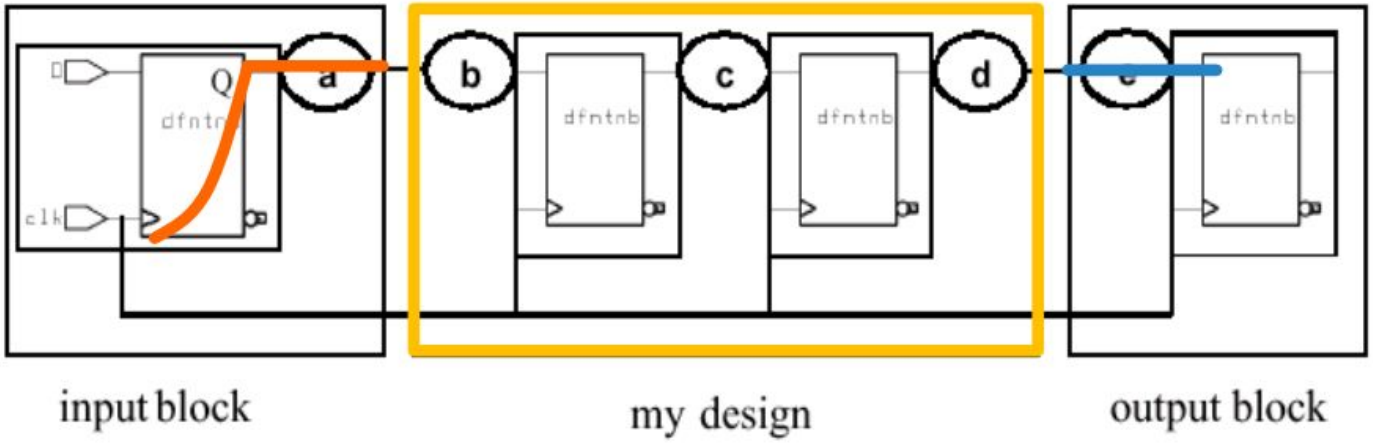


Actual data path delay (Data Arrival Time): 1 + 2 + 3

Input / Output Delay

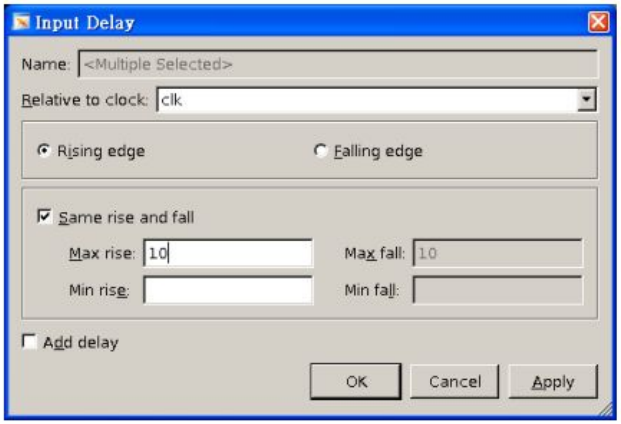
- Clock cycle >= $DFF_{clk\text{-}Qdelay}$ + a + b + DFF_{setup}
- **Input delay = $DFF_{clk\text{-}Qdelay}$ + a**

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

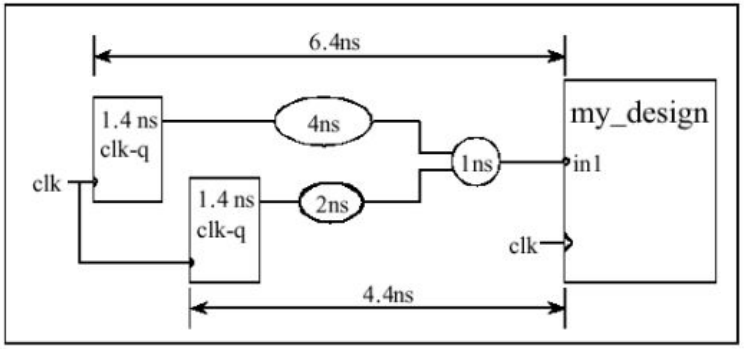


Setting Input Delay

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



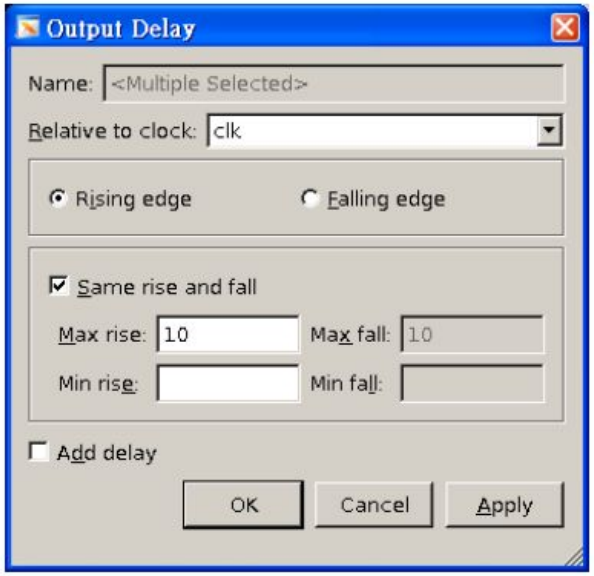
◆ Example



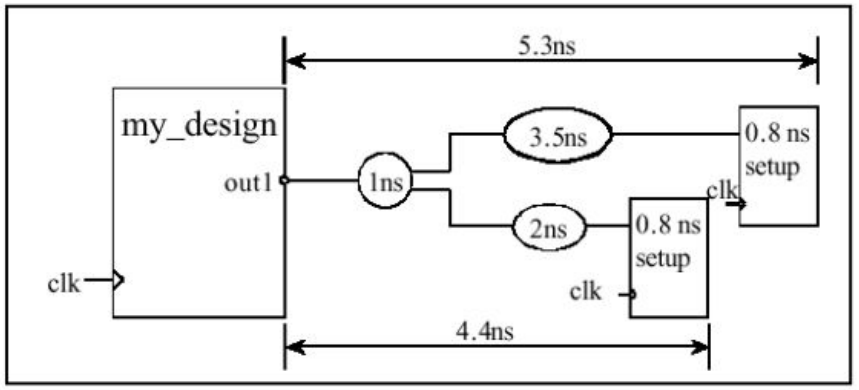

```
set_input_delay -clock clk -max 6.4 [get_ports in1]
set_input_delay -clock clk -min 4.4 [get_ports in1]
```

Setting Output Delay

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



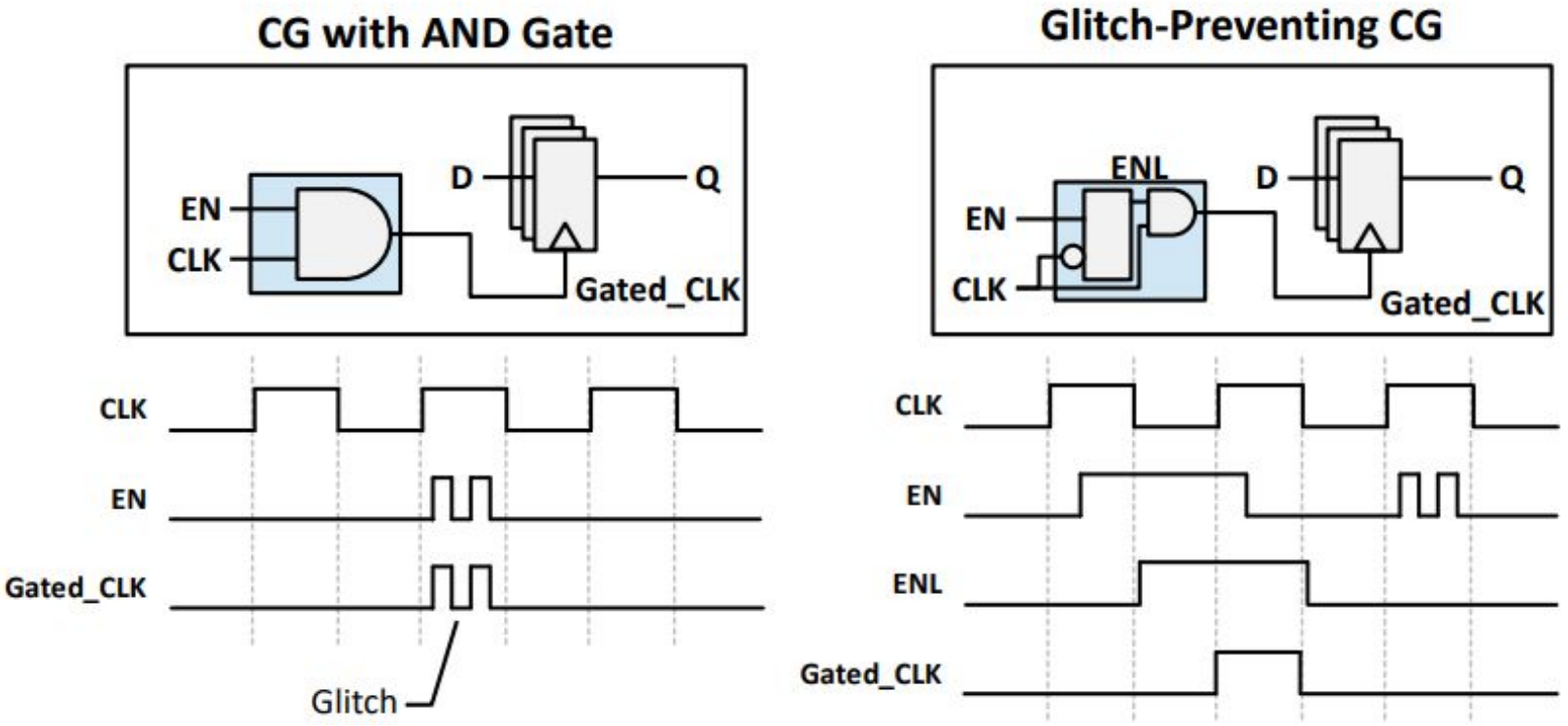
◆ Example




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

Special Circuits: Clock Gating

- Reduce dynamic power dissipation



Clock Gating (Auto CG)

- If Statement

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

- Conditional Assignment

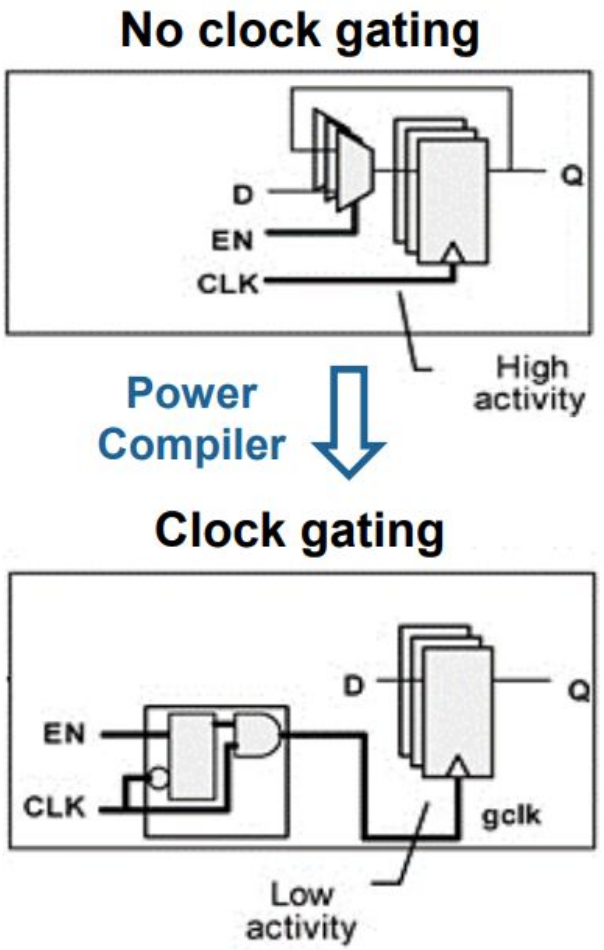
```
always@(posedge clk) begin
    Q <= (enable)? D_in:Q;
end
```

- Clock gating script style 1

```
insert_clock_gating
compile
```

- Clock gating script style 2

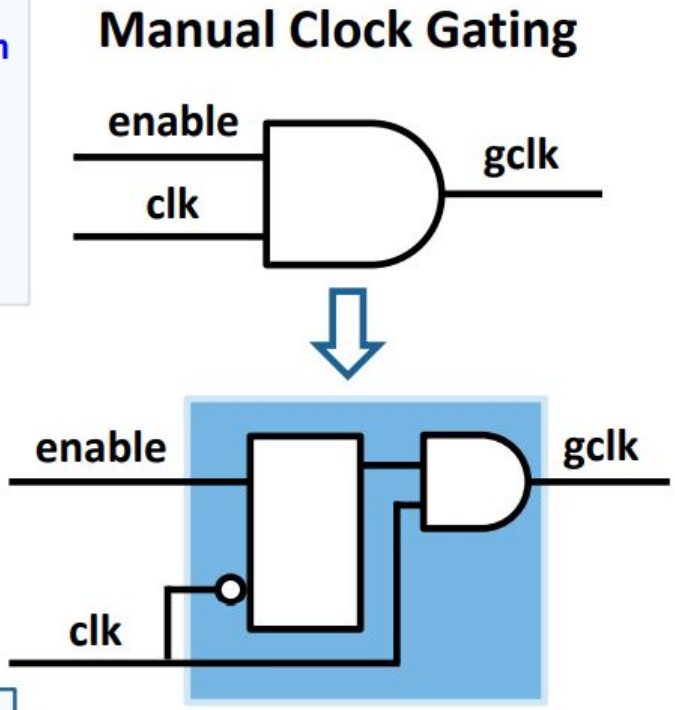
```
compile -gate_clock
```



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*



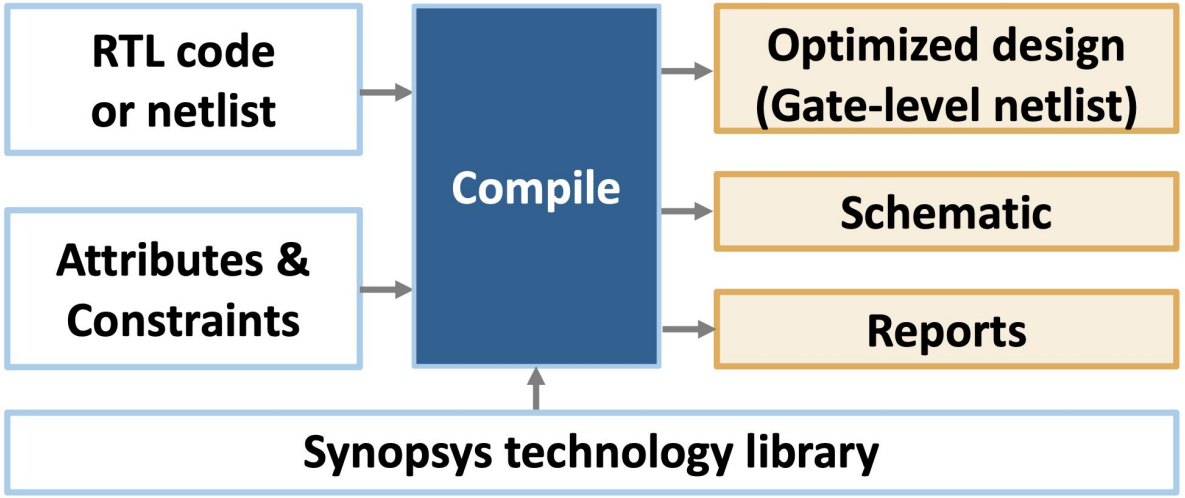
- Clock gating script

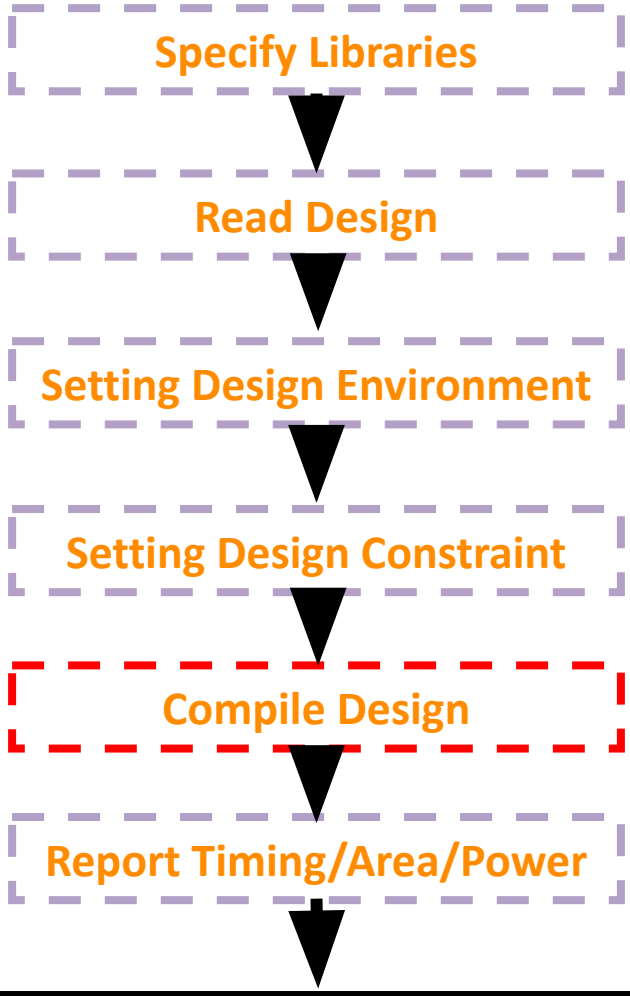
```
replace_clock_gates
compile
```

- Report clock gating

```
report_clock_gating -gating_elements
```

Synthesis Design Flow - Compile Design





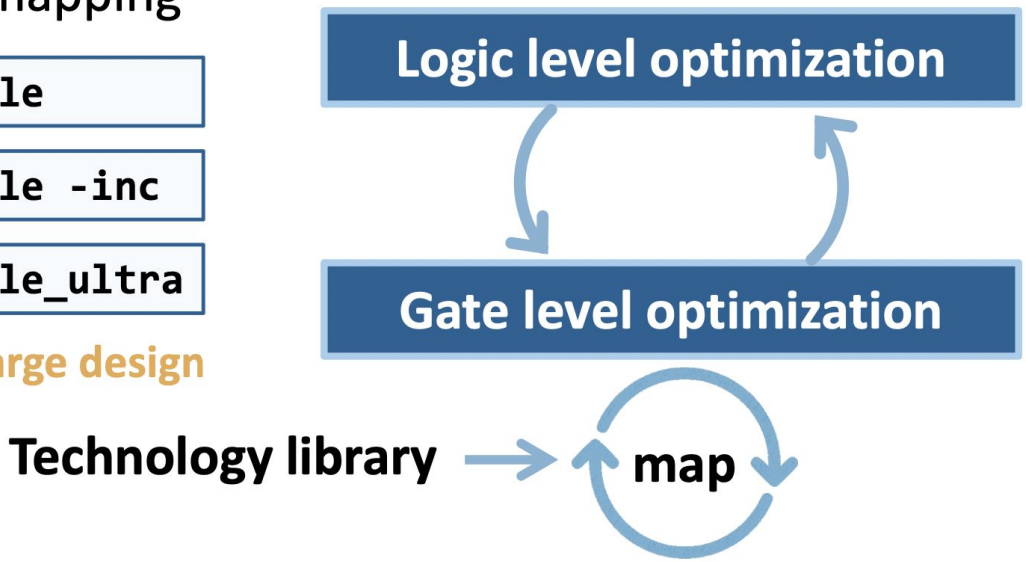
Compile Design - Perform optimization

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
```

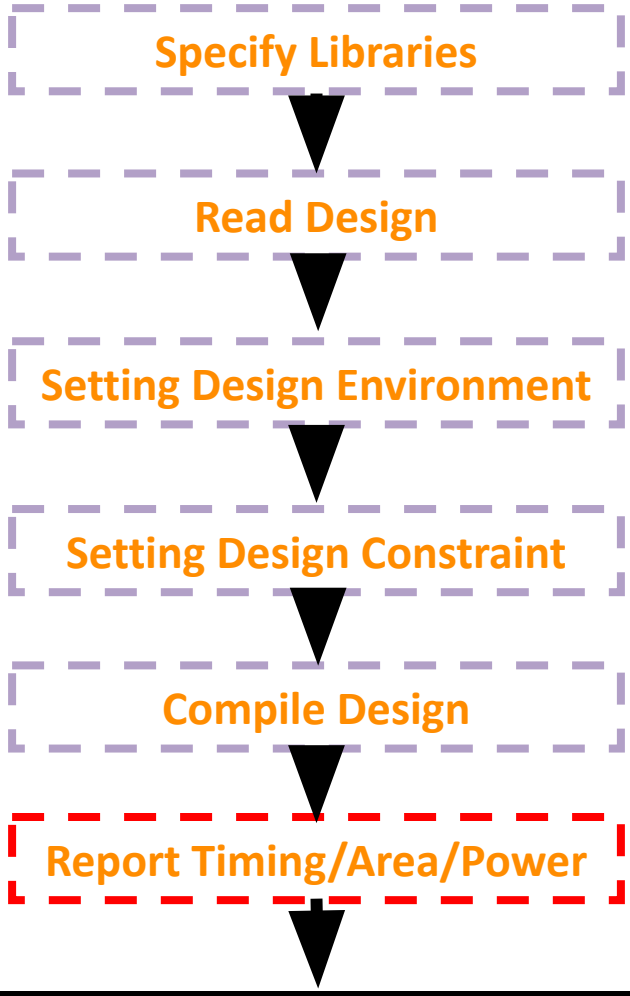
*Careful use for large design



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

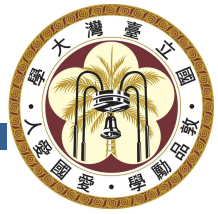
```
25   # Report Synthesis Results
26   report_timing > "./Report/${DESIGN}_syn.timing"
27   report_area > "./Report/${DESIGN}_syn.area"
28   report_power > "Report/$DESIGN_syn.power"
```



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)
  - Record the constraints and synthesis results

```
#================================================================================
# 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**

```
53 `ifdef SDF
54         initial $sdf_annotate(`SDFFILE, u_CONV);
55 `endif
56
```

Reference

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