Computer-Aided VLSI System Design

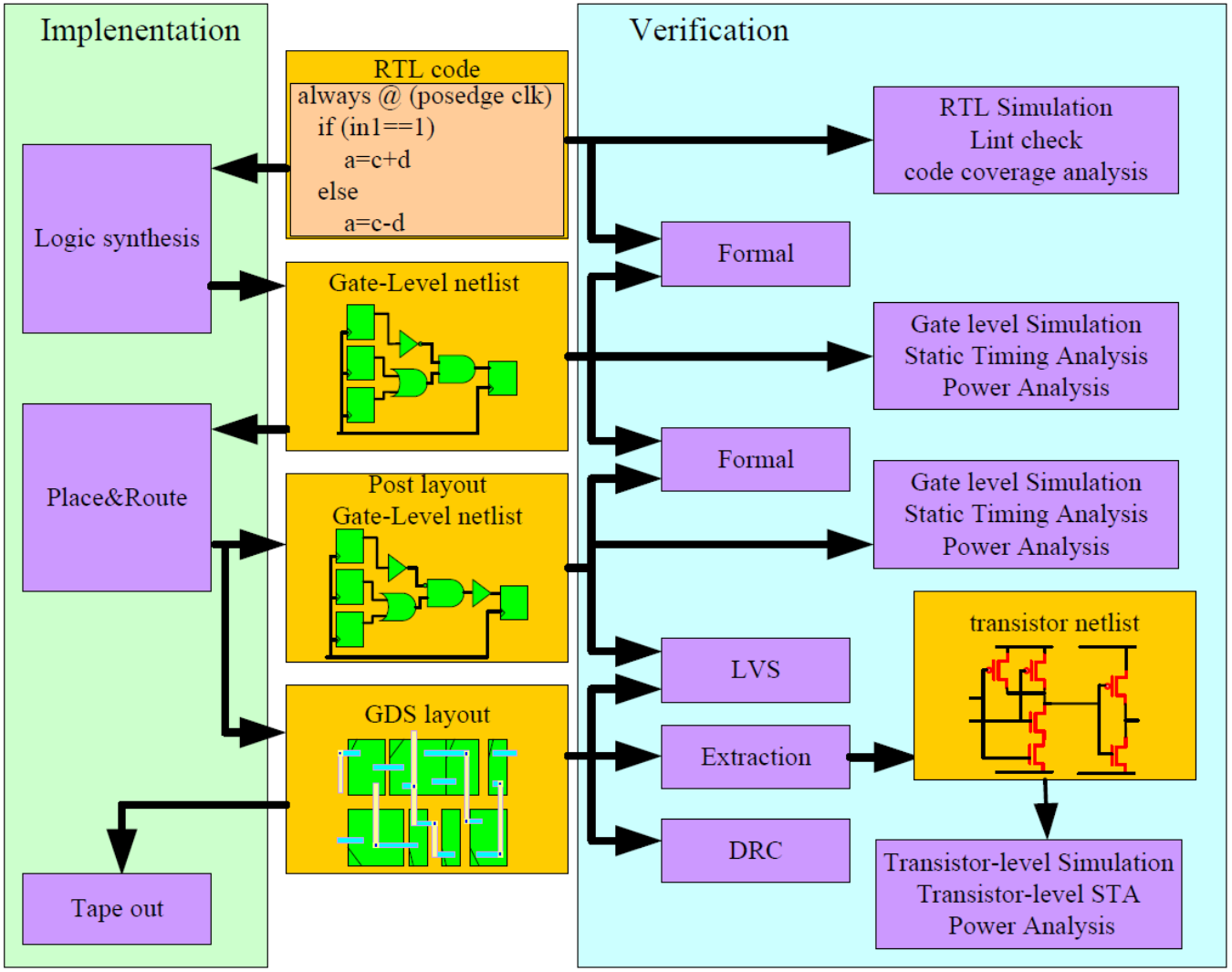
Chapter 5. Synopsys Synthesis Design Compiler & Post-sim Part2

Lecturer: Cheng-En Chiang

Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE

Recap: Cell-based IC Design Flow



Recap: Logic Synthesis

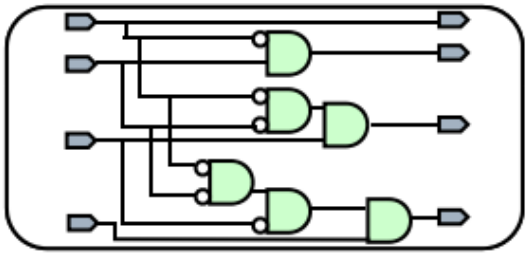
Synthesis = translation + optimization + mapping

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

HDL Source
(RTL)

Translate (HDL Compiler)

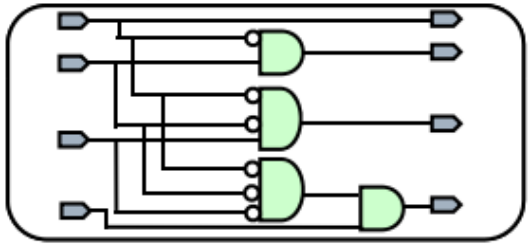
No Timing Info.



Generic Boolean
(GTECH)

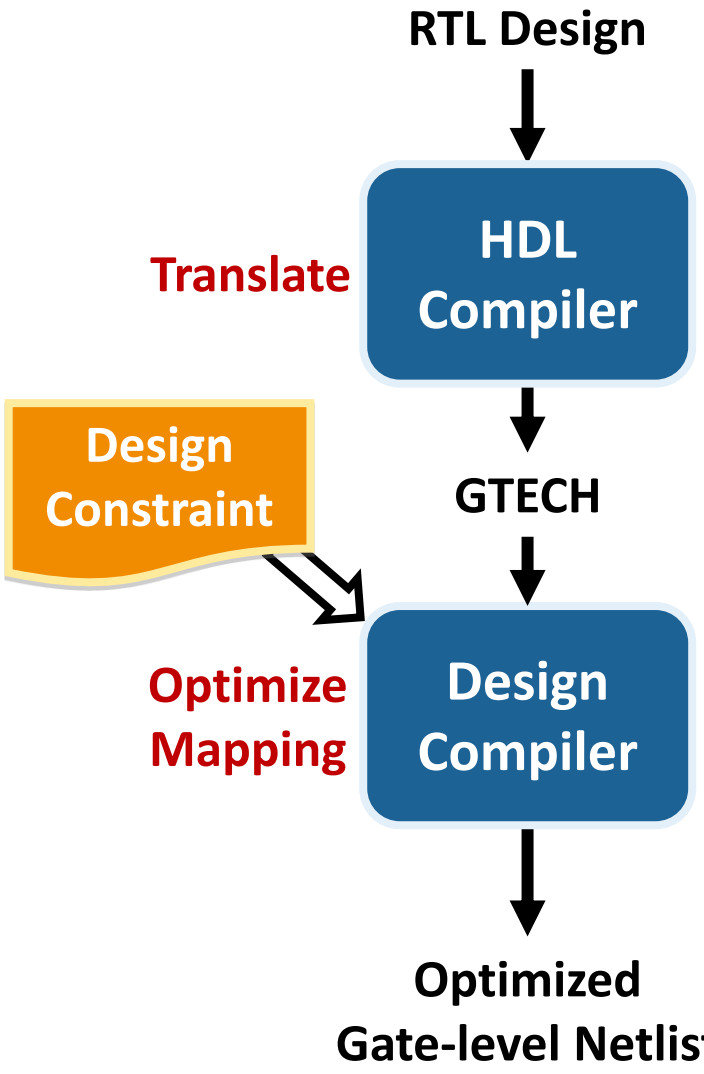
Optimize + Mapping
(Design Compiler)

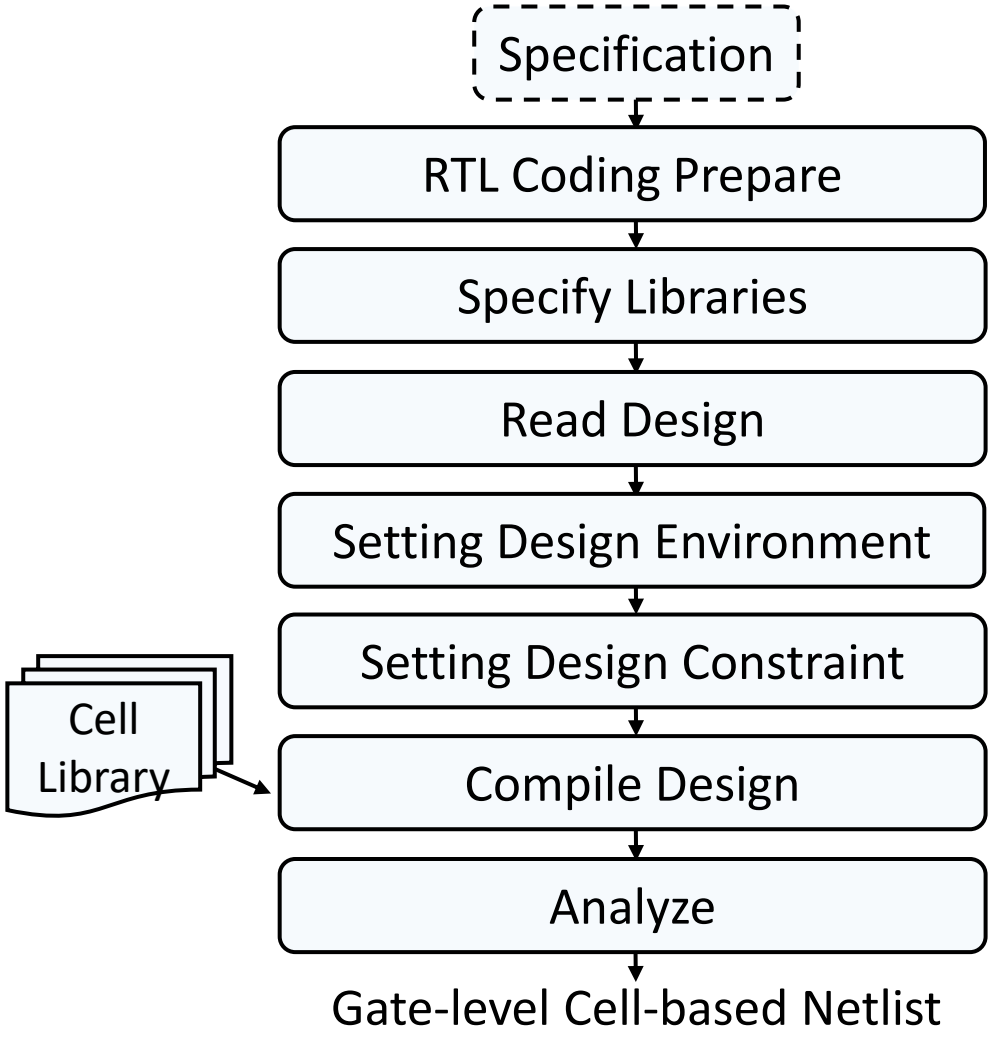
Timing Info.
Constraints

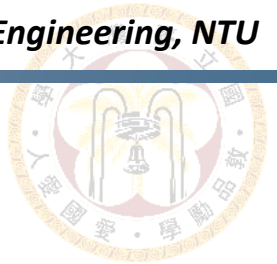


Target Technology

Recap: Synthesis Design Flow

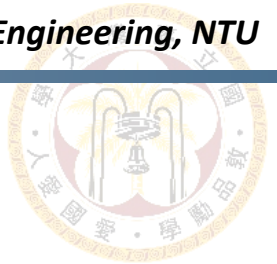






Outline

- Introduction
- RTL Coding Related to Synthesis
- Synopsys Environment
- Synthesis Design Flow
- Gate-Level Simulation
- Files Demo



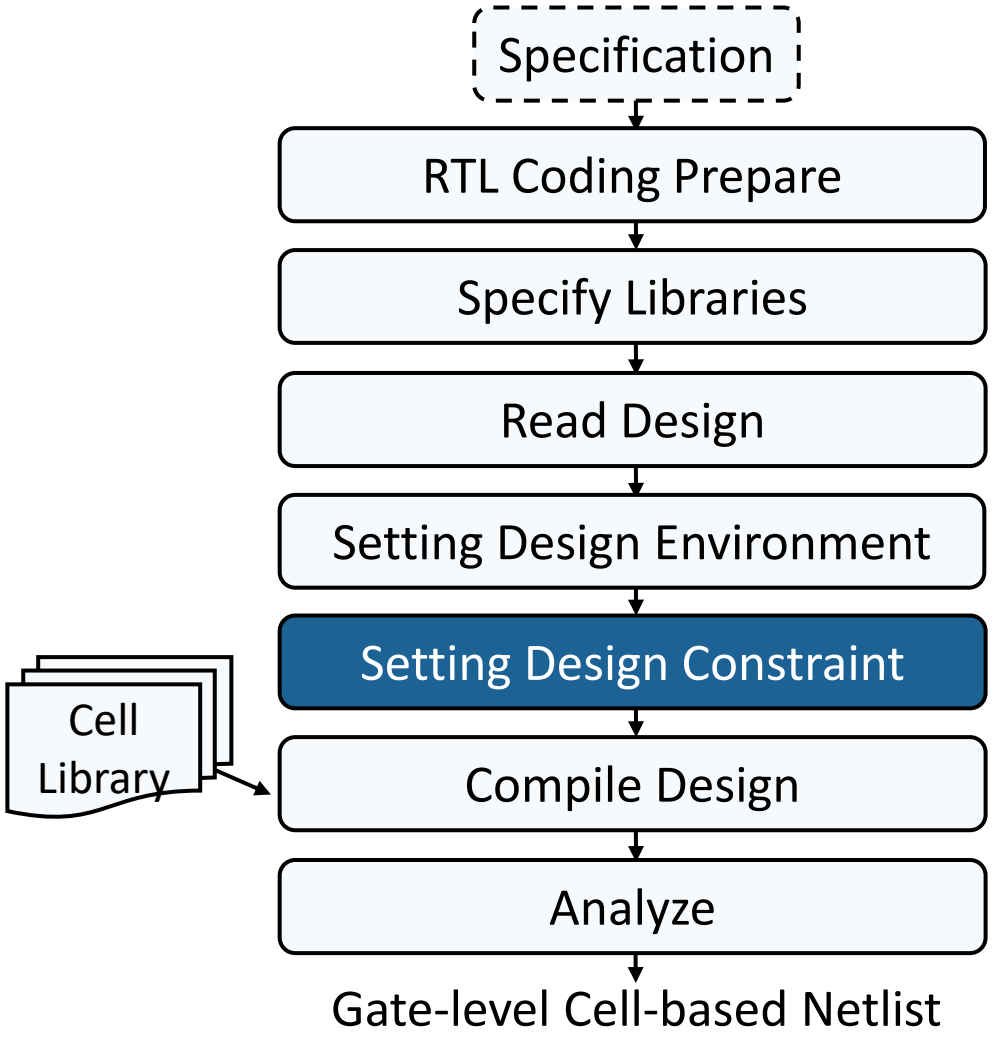
Outline

- Introduction
- RTL Coding Related to Synthesis
- Synopsys Environment
- **Synthesis Design Flow**
- Gate-Level Simulation
- Files Demo

Synthesis Design Flow

- Set constraints for synthesis
  - Design rule constraints
  - Clock constraints
  - Special circuits
  - Optimization objective

```
# Design Rule Constraints
set_max_transition
set_max_fanout
set_max_capacitance
```



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 constraint are set:
  - Maximum transition through a net, the longest time to drive a pin and change its value

```
set_max_transition 0.1
```

  - Maximum capacitive load of a net, is total capacitive load(net) that an output pin can drive

```
set_max_capacitance 0.1
```

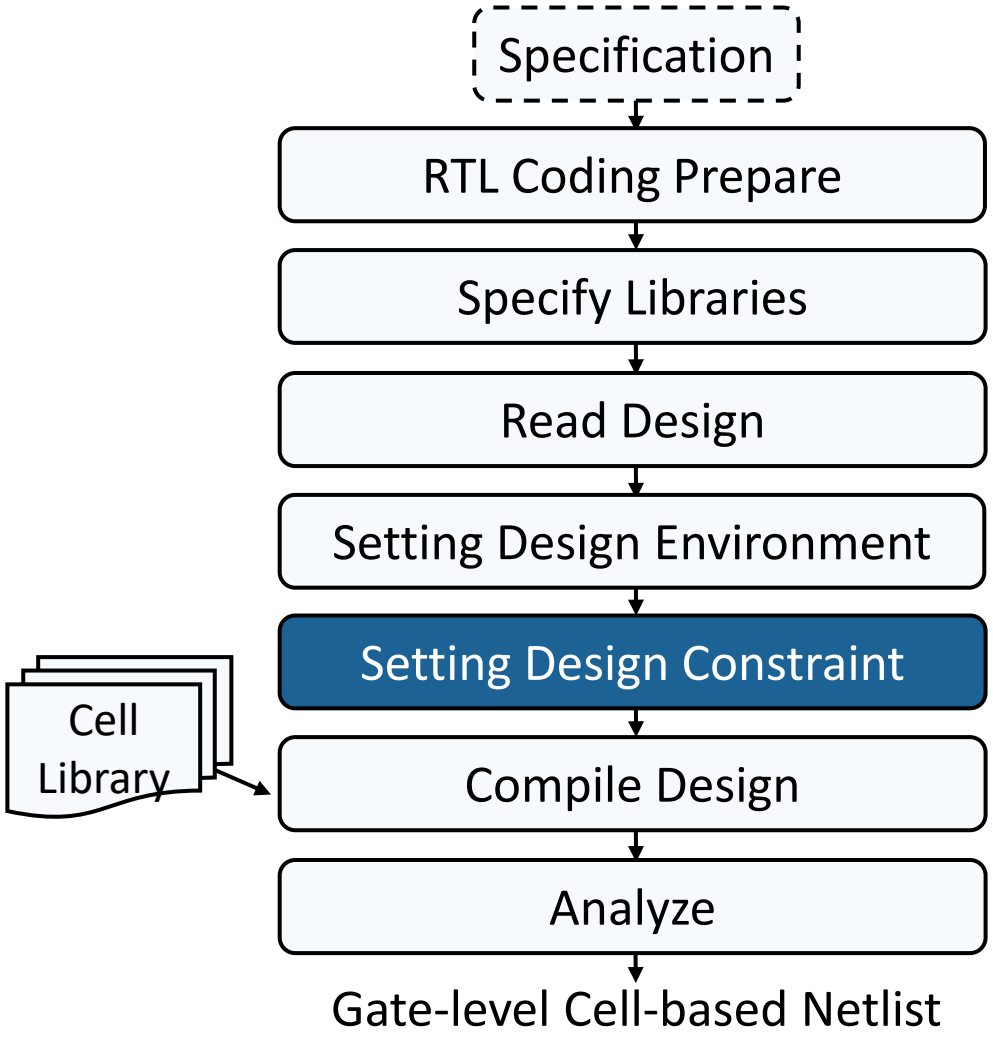
  - Maximum fanout of each gate, Σfanout_load <= max_fanout

```
set_max_fanout 20
```

Synthesis Design Flow

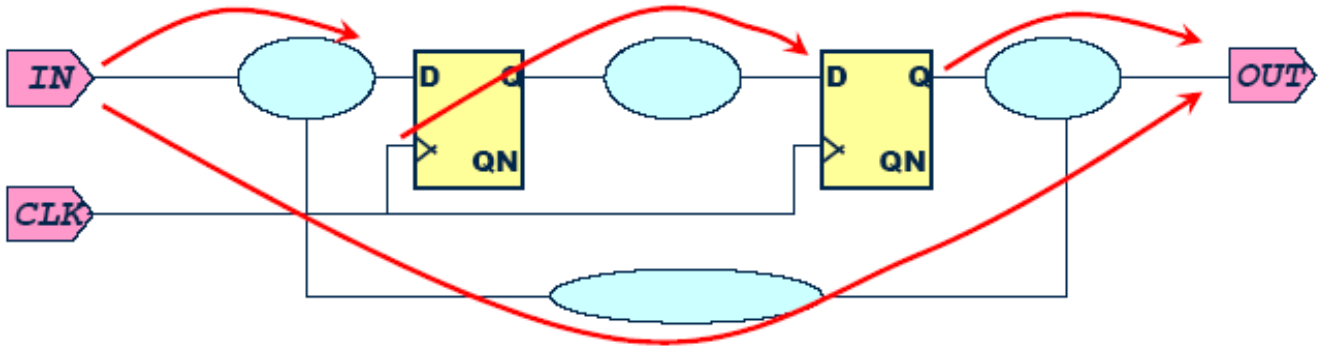
- Set constraints for synthesis
  - Design rule constraints
  - Clock constraints
  - Special circuits
  - Optimization objective

```
# Clock Constraints
1 create_clock
  set_fix_hold
2 set_dont_touch_network
  set_ideal_network
3 set_clock_uncertainty
4 set_clock_latency
5 set_input_transition
6 set_clock_transition
```



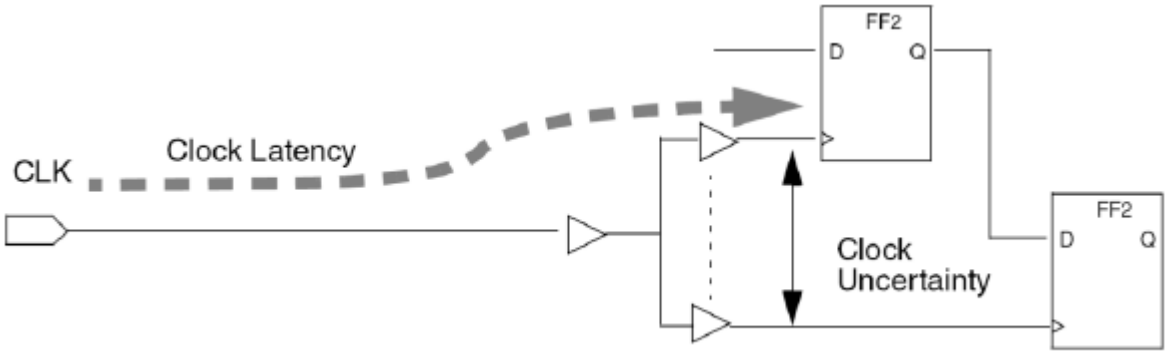
Recall: Timing Path

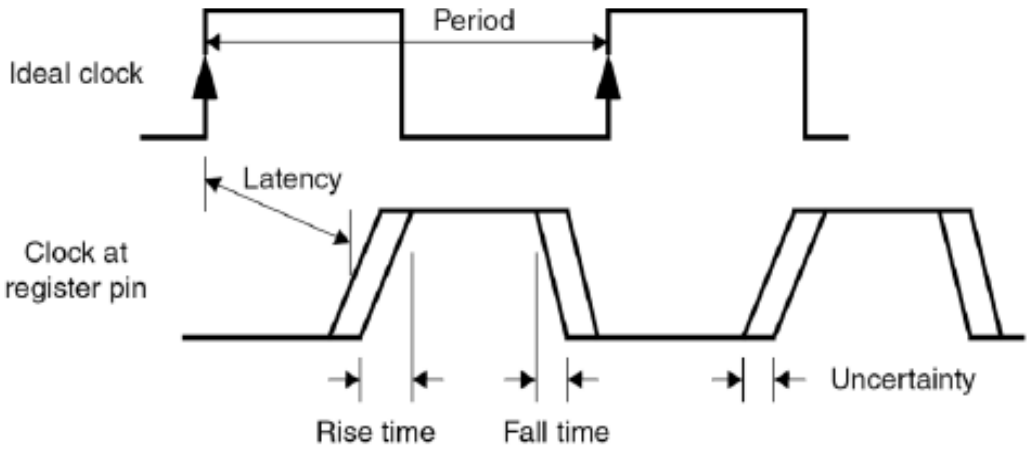
- Generally each timing path is analyzed for timing constraints when compiling
- 4 type of timing paths
  - Input to register
  - Register to register
  - Register to output
  - Input to output



Basic Clock Constraints

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

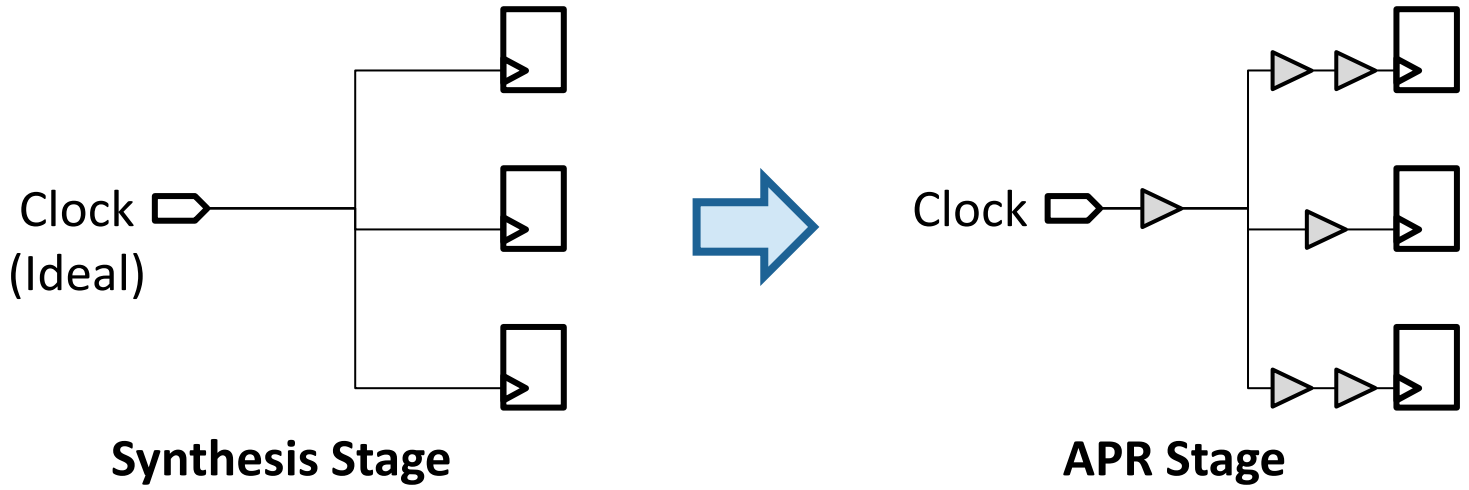




Clock Modeling in Synthesis & APR

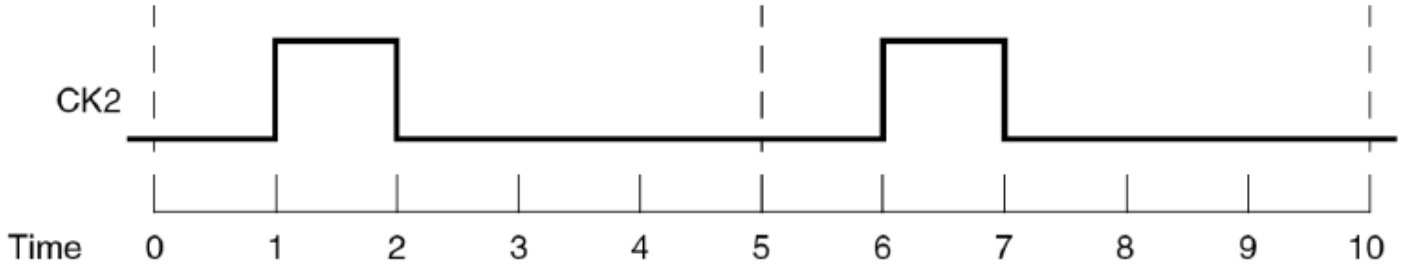
- Clock modeling in automatic place and route (APR) stage
  - Clock tree synthesis is performed
  - A buffer/inverter tree is built for the clock signal

- Clock modeling in synthesis stage
  - Ideal network + clock uncertainty + clock latency



1. Create Clock

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

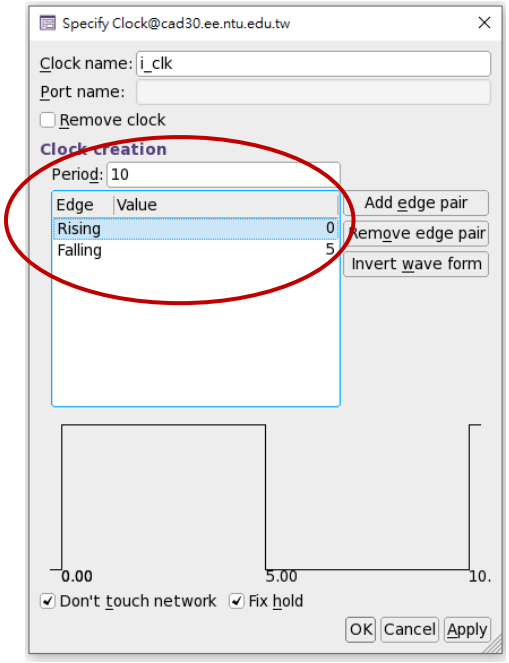


1. Create Clock

- Arguments
  - Name: name for the clock
  - Period: period of the clock, value >= 0
  - Waveform: alternating rise, fall times for 1 period
    - Must have even number of values
    - Default: {0 period/2}

- ***Attributes/Specify Clock***

```
create_clock -name clk -period 10 \
    -waveform {0 5} [get_ports clk]
```



1. Create Clock

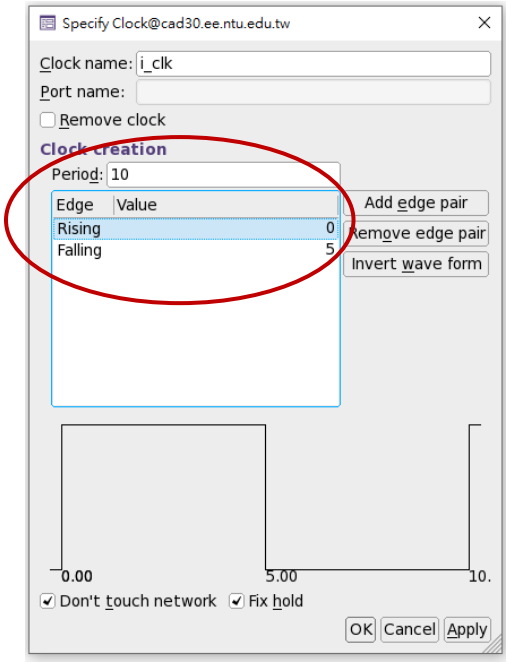
- Arguments
  - Name: name for the clock
  - Period: period of the clock, value >= 0
  - Waveform: alternating rise, fall times for 1 period
    - Must have even number of values
    - Default: {0 period/2}

- ***Attributes/Specify Clock***

```
create_clock -name clk_1 -period 10 \
    -waveform {0 5} [get_ports clk]
```

Port name to apply clock & clock name can be different
(not suggested)

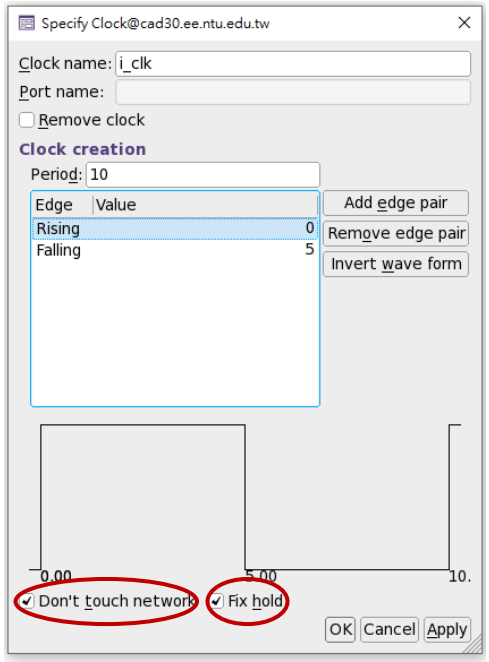
```
get_clocks clk_1
get_ports clk
```



2. Specify Clock Constraints

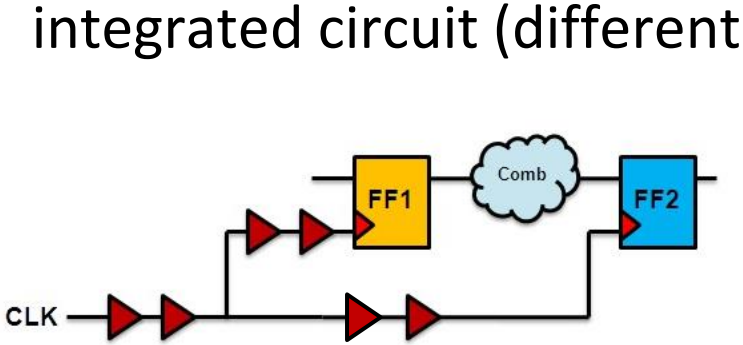
- set_fix_hold
  - Respect hold time requirement of all clocked flip-flops
- set_dont_touch_network
  - Do not re-buffer clk network
- set_ideal_network
  - No delay on specified net
  - Can be applied on clk, asyn reset, high fanout nets

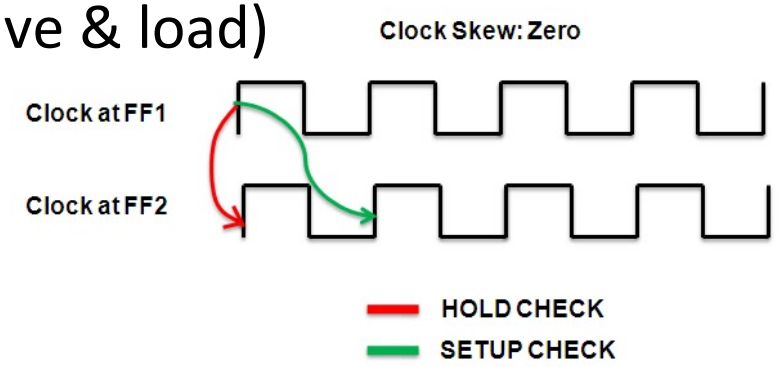
```
set_fix_hold [get_clocks clk]
set_dont_touch_network [get_clocks clk]
set_ideal_network [get_ports clk]
```

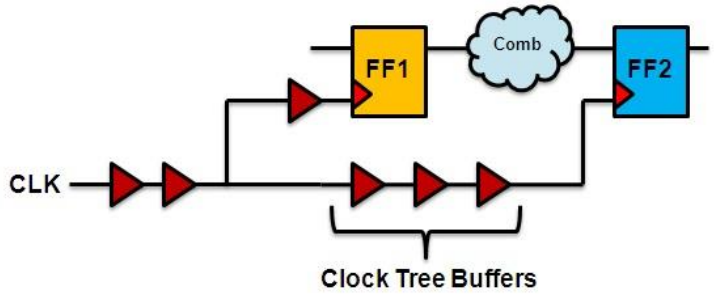


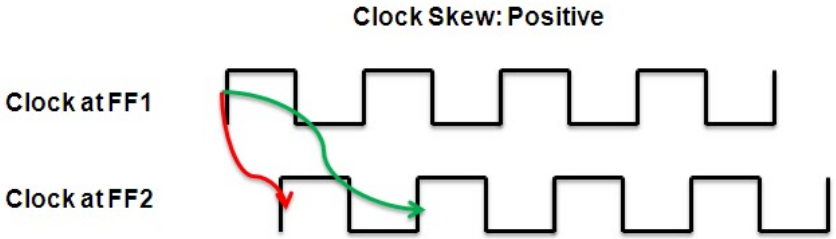
Clock Uncertainty: Skew

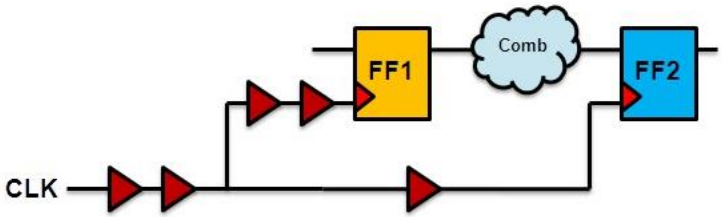
- The **spatial variation in arrival time** of a clock transition on an integrated circuit (different drive & load)

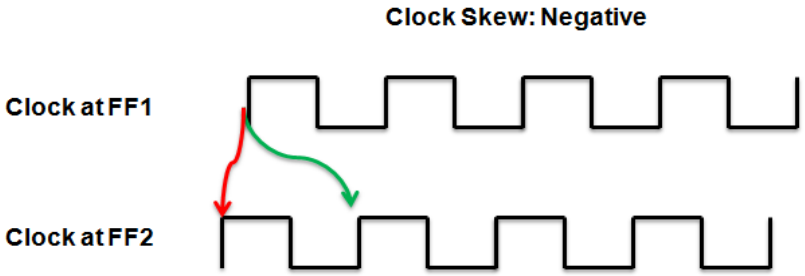






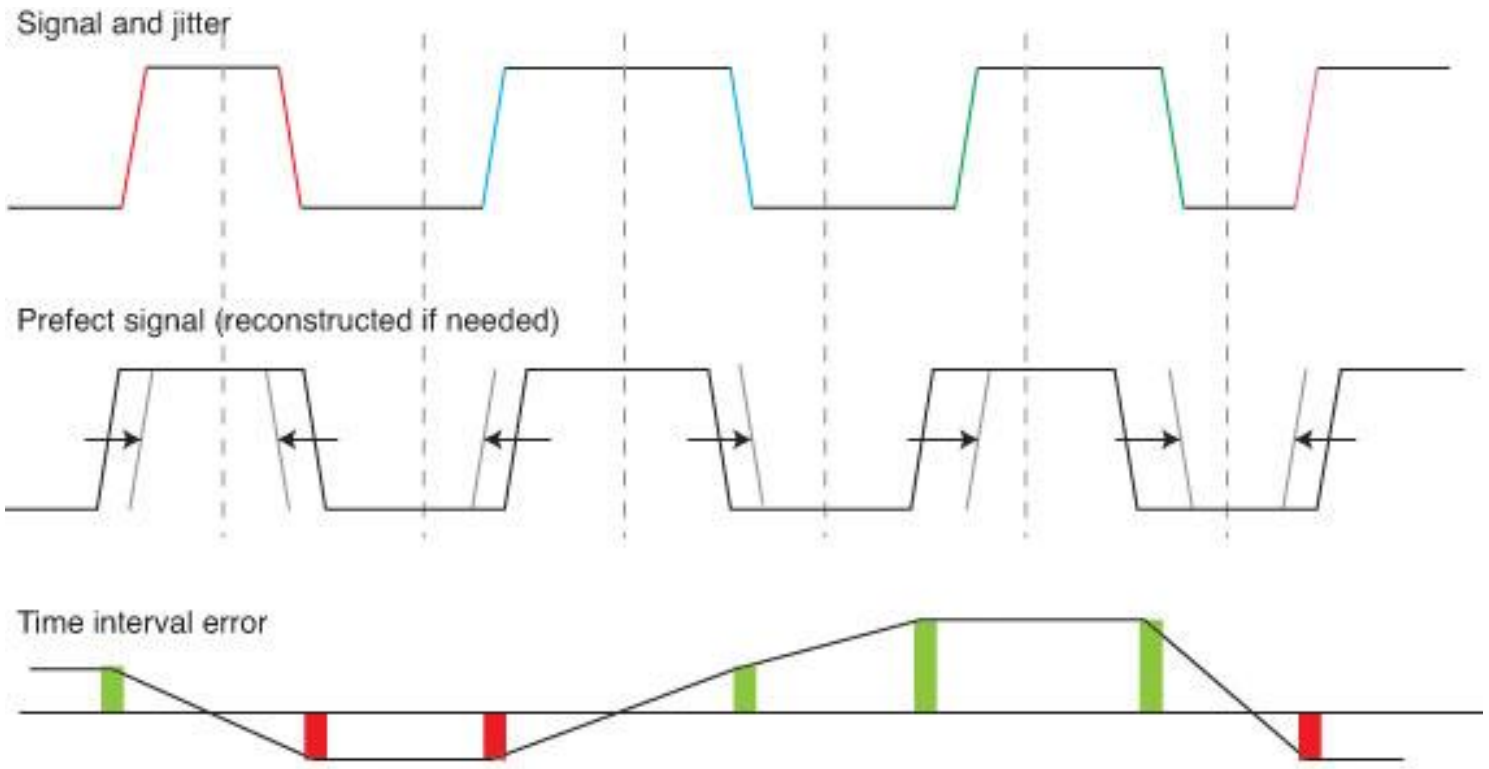






Clock Uncertainty: Jitter

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



3. Setting Clock Uncertainty

- Variation in clock arrival time
  - Models clock skew + jitter effects on the clock
  - Experience
    - Small circuits: 0.1ns
    - Large circuits: 0.3ns

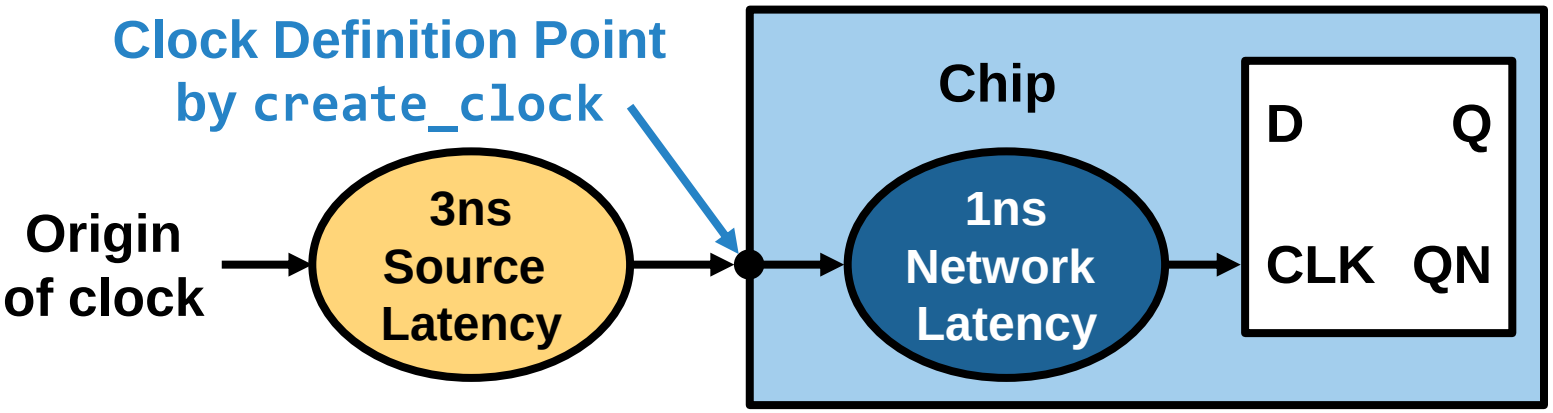
```
set_clock_uncertainty 0.1 [get_clocks clk]
```

```
memory_8k_64_2r_2c/aru/U488/Y (MXI4X1)                    0.40      10.69 r
memory_8k_64_2r_2c/aru/U668/Y (NOR2X1)                    0.09      10.77 f
memory_8k_64_2r_2c/aru/data_in_sc[0] (aru)                0.00      10.77 f
memory_8k_64_2r_2c/sc_memory/D[0] (sc_memory)             0.00      10.77 f
data arrival time                                                   10.77

clock clk (rise edge)                                    10.00      10.00
clock network delay (ideal)                               1.00      11.00
clock uncertainty                                        -0.10      10.90
memory_8k_64_2r_2c/sc_memory/CLK (sc_memory)              0.00      10.90 r
library setup time                                       -0.12      10.78
data required time                                                  10.78
-----------------------------------------------------------------------------
data required time                                                  10.78
data arrival time                                                  -10.77
-----------------------------------------------------------------------------
slack (MET)                                                         0.00
```

Clock Network Latency

- The time it takes a for clock signal to propagate **from the clock definition point to a register clock pin**
  - Clock definition by `create_clock` command

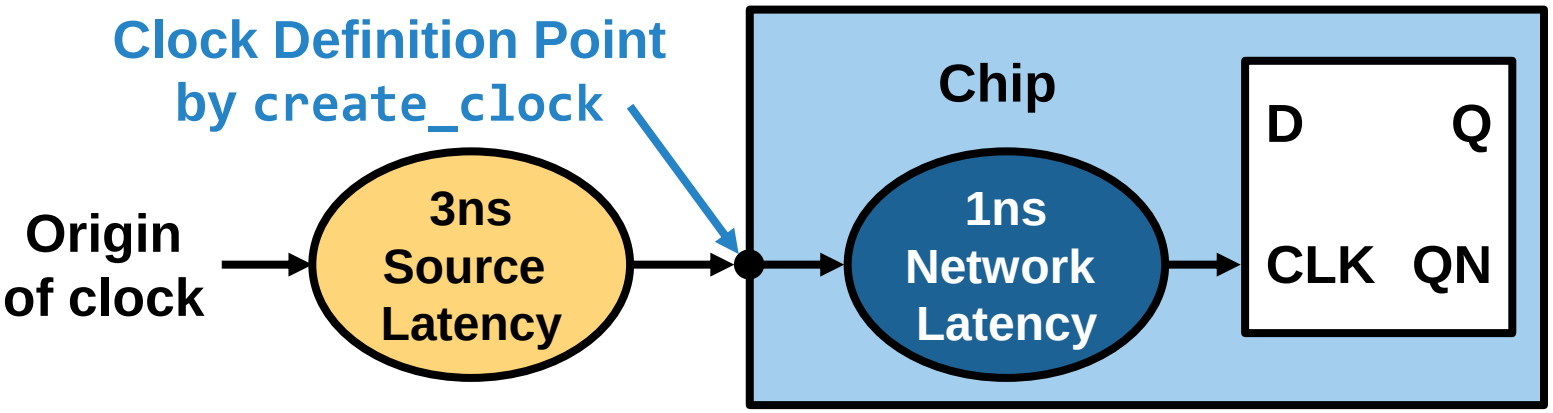
- Affects the slack of in-to-reg & reg-to-out paths



Clock Source Latency (Optional)

- The time it takes for a clock signal to propagate from its origin point to the clock definition point

- May help in multi-clock design



4. Setting Clock Latency

- Network latency

```
set_clock_latency 0.5       [get_clocks clk]
```

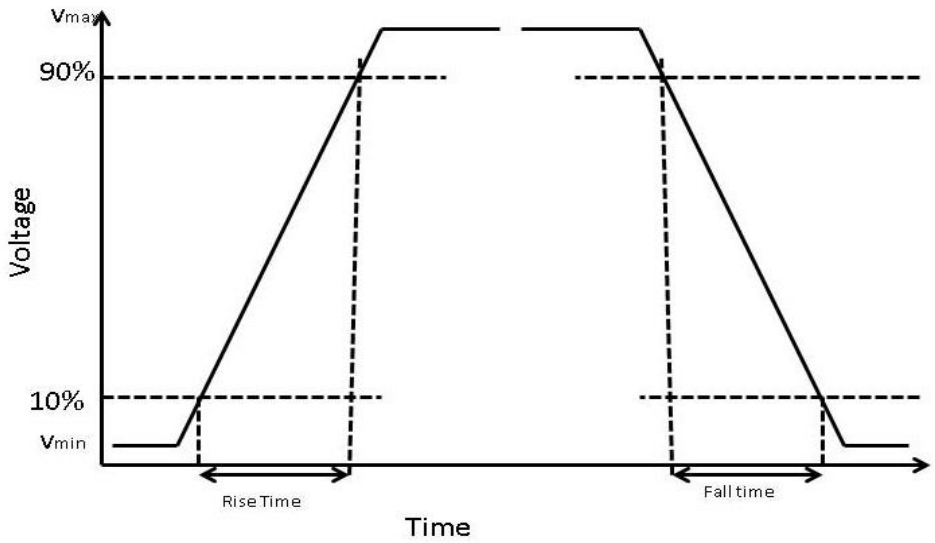
```
set_clock_latency -fall 0.5 [get_clocks clk]
set_clock_latency -rise 0.5 [get_clocks clk]
```

- 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]
```

5 & 6. Setting Clock/Input Transition



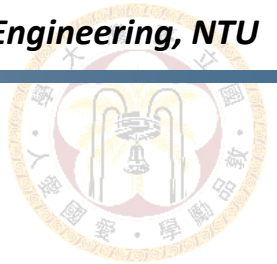
- Clock transition

```
set_clock_transition 0.1          [get_clocks clk]
```

```
set_clock_transition 0.1   -fall [get_clocks clk]
set_clock_transition 0.2   -rise [get_clocks clk]
```

- Input Transition

```
set_input_transition 0.5          [all_inputs]
```



Constraints for APR

- Synthesis uses approximated models for clock & wires
  - Clock: ideal network + clock uncertainty + clock latency
  - Wire: wire load model (estimates R, C, A based on fanout)

- In automatic placement & routing (APR) stage
  - Clock tree synthesis (CTS) for a more accurate clock model
  - Routing generates physical layout of wires

Constraints for APR

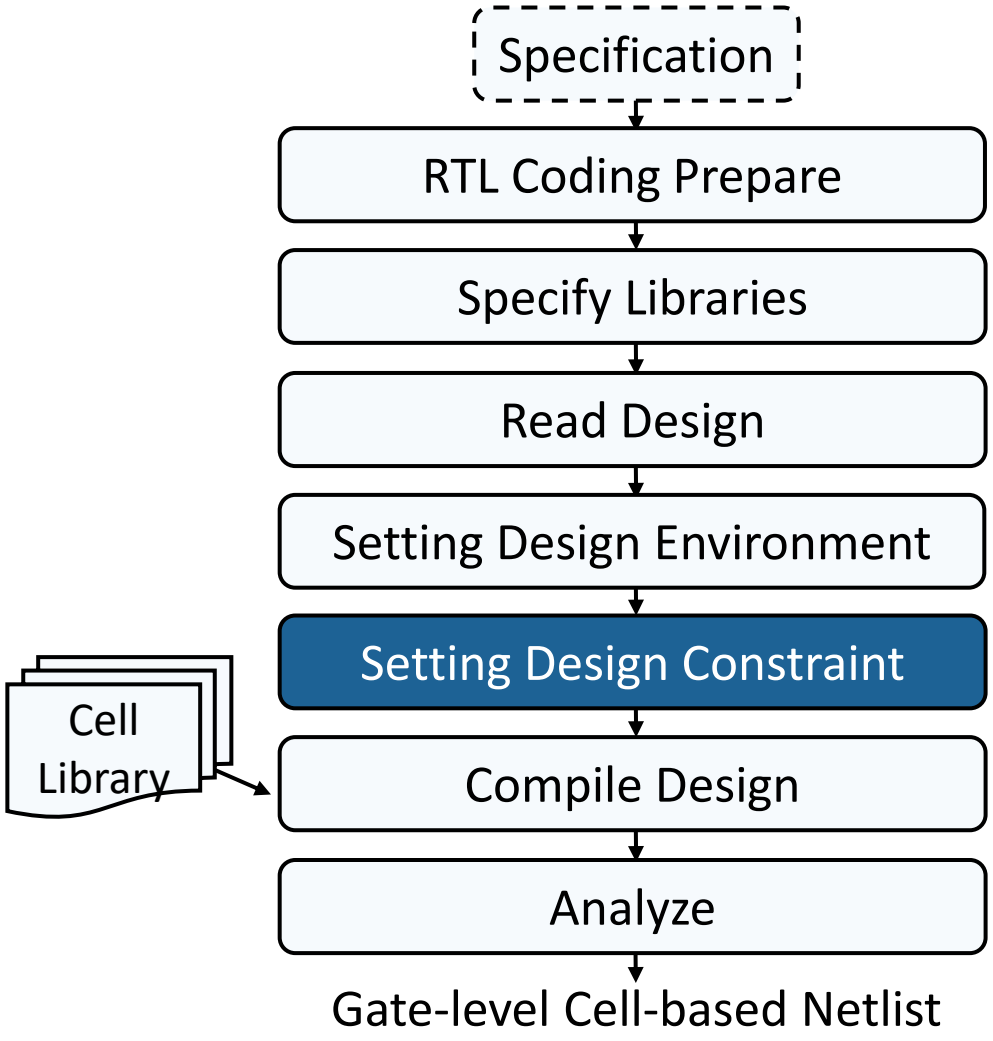
- Some constraints should be removed before APR

```
# set_wire_load_model -name tsmc18_w10 -library slow
# set_wire_load_mode top

create_clock -period 10     [get_ports clk]
set_fix_hold                [get_clocks clk]
# set_dont_touch_network      [get_clocks clk]
# set_ideal_network           [get_ports clk]
# set_clock_uncertainty 0.1   [get_ports clk]
# set_clock_latency 0.5       [get_clocks clk]
set_clock_transition 0.1    [get_clocks clk]
set_input_transition 0.5    [all_inputs]
```

Synthesis Design Flow

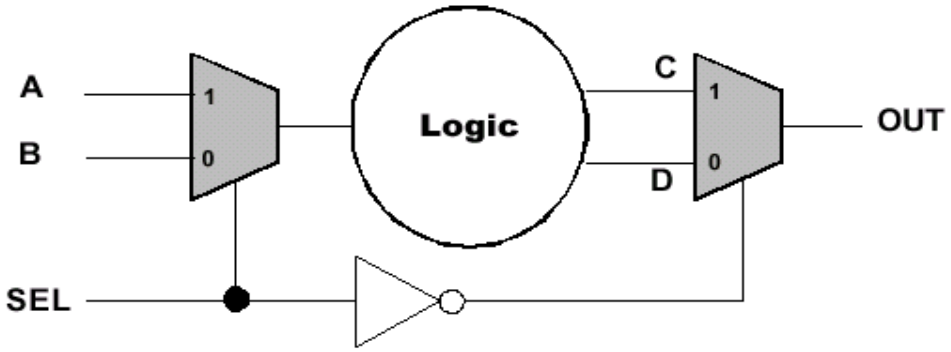
- Set constraints for synthesis
  - Design rule constraints
  - Clock constraints
  - Special circuits
  - Optimization objective

```
# Special Circuits
set_false_path
replace_clock_gates

# Optimization Objective
set_max_area
```



False Path

- Timing path that we wish to **ignore timing constraints**
- Command `set_false_path` disables timing-based synthesis for specified path
- It is useful for:
  - Constraining asynchronous paths
  - Constraining logically false paths

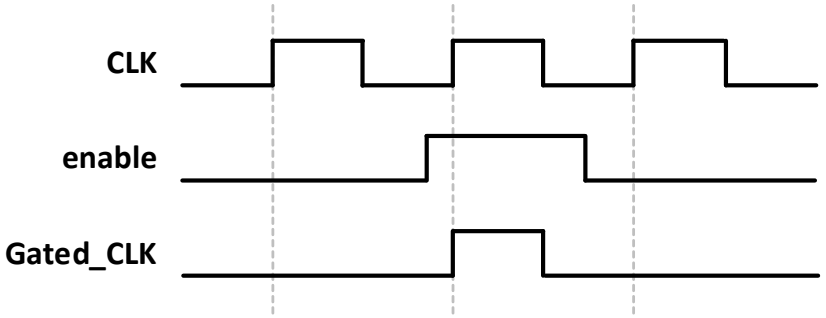


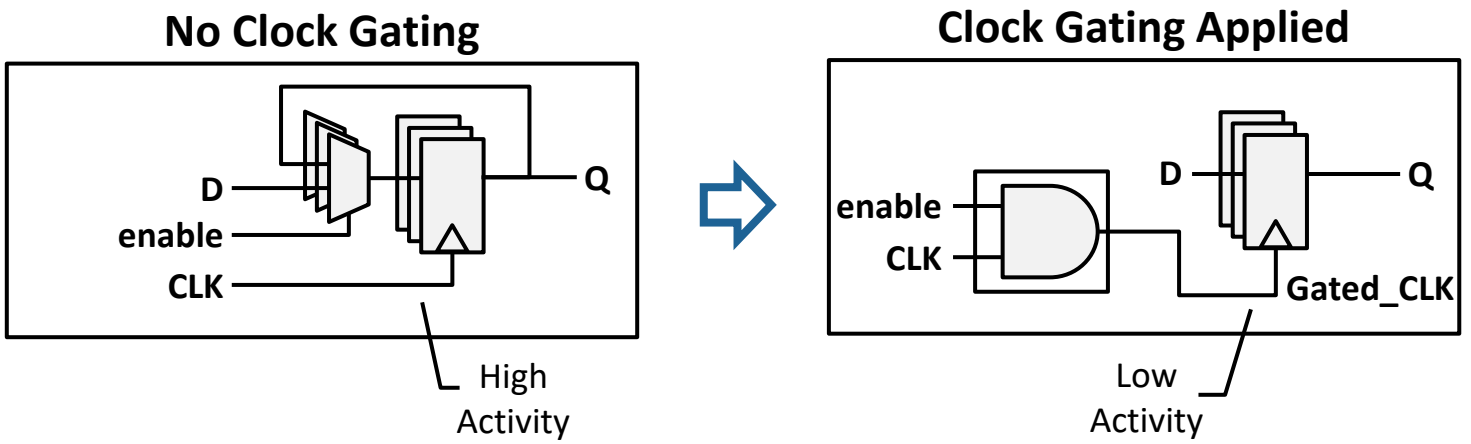

```
set_false_path -from {A} -through {C} -to {OUT}
set_false_path -from {B} -through {D} -to {OUT}
```

Clock Gating (CG) Concept (1/2)

- Concept: clock signal arrives only when data is to be switched
  - Reduce dynamic power dissipation

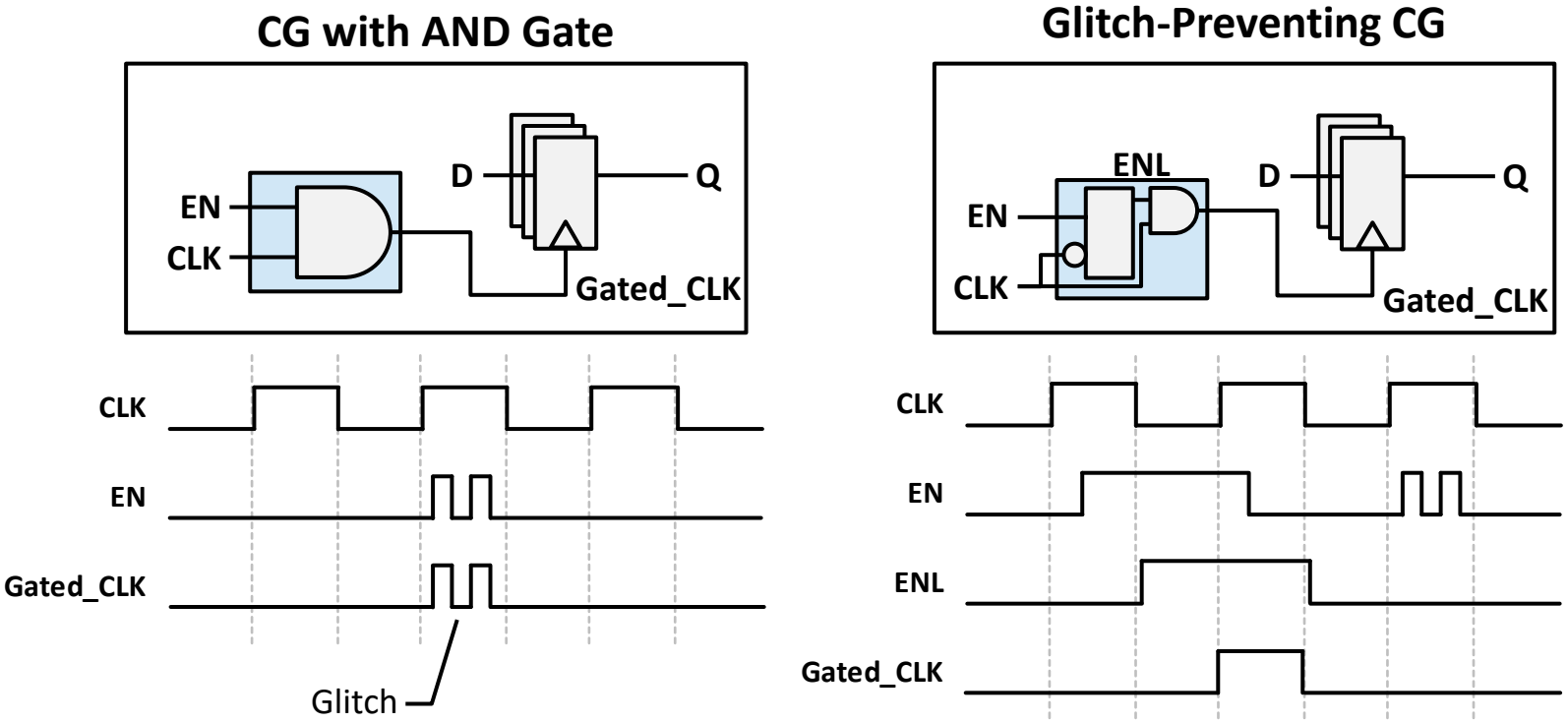
```
always@(posedge clk) begin
    if (enable) Q <= D;
    else        Q <= Q;
end
```





Clock Gating (CG) Concept (2/2)

- Unstable `enable` signal causes glitch on clock gating output
- Glitch prevention: enable generated by **latch with negative clk**



Clock Gating Commands (Auto CG)

- If Statement

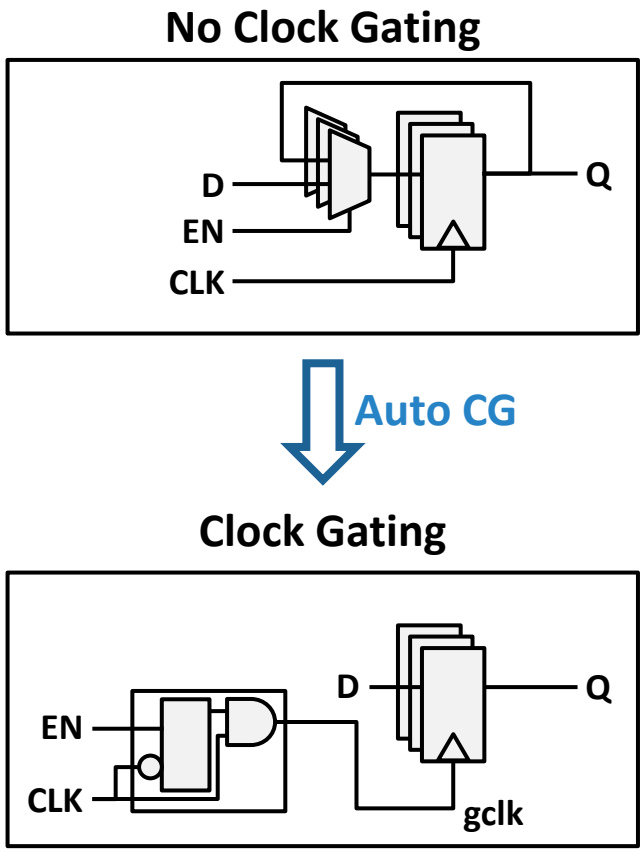
```
always@(posedge clk) begin
    if (enable) Q <= D;
    else        Q <= Q;
end
```

- Conditional Assignment

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

- Clock gating example script

```
set_clock_gating_style \
    -max_fanout 4 \
    -pos integrated \
    -control_point before \
    -control_signal scan_enable
compile -gate_clock
```



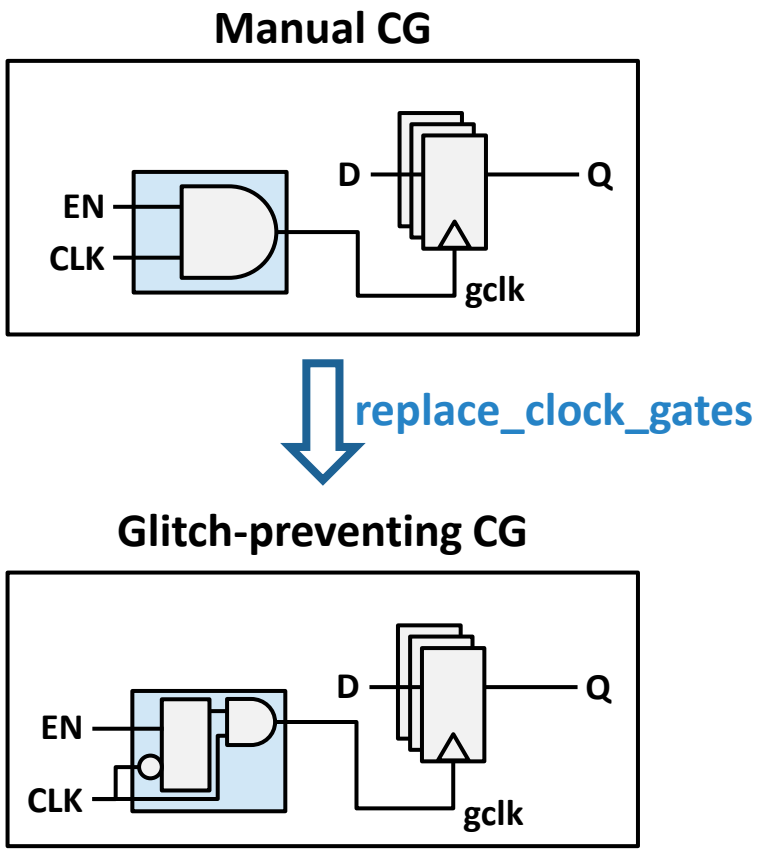
Clock Gating Commands (Manual)

- User manually defined gated clock signal

```
// Note: `define SYN in synthesis
assign gclk = clk & enable;
`ifdef SYN
    always@(posedge gclk) begin
        Q <= D;
    end
`else
    always@(posedge clk) begin
        if (enable) Q <= D;
    end
`endif
```

- Clock gating example script

```
set_clock_gating_style ...
replace_clock_gates
compile
```



Report Clock Gating

report_clock_gating -gating_elements

Clock Gating Report by Origin

	Actual (%) Count
Number of tool-inserted clock gating elements	54 (100.00%)
Number of pre-existing clock gating elements	0 (0.00%)
Number of gated registers	214 (99.53%)
Number of tool-inserted gated registers	214 (99.53%)
Number of pre-existing gated registers	0 (0.00%)
Number of ungated registers	1 (0.47%)
Number of registers	215

report_clock_gating -ungated

Ungated Register Report

Ungated Register	Reason	What Next?
reg_enable_reg	Always enabled register	-

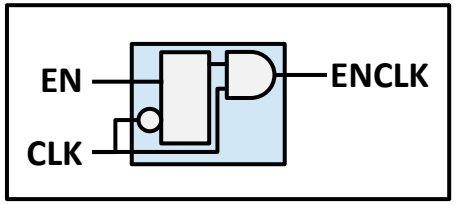
Tool-inserted ungated reasons

Reason for not gating	Regs	Regs %	Bits	Bits %
Always enabled register	1	100.00	1	100.00

Using Integrated CG (ICG) Cell

- Identify CG cell to use in `set_clock_gating_style -pos`
- Pos: specify CG cell for posedge triggered FFs
  - And (default): DC selects gates to form CG cell
  - Integrated: foundry-provided ICG cell

```
set_clock_gating_style \
    -pos {and | integrated}
compile -gate_clock
```



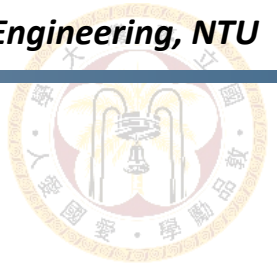
Gate-level Netlist: Gates Selected by DC

```
module SNPS_CLOCK_GATE_HIGH_top_2 ( CLK, EN, ENCLK );
  input CLK, EN;
  output ENCLK;
  wire   net350;

  TLATNX1 latch ( .D(EN), .GN(CLK), .Q(net350) );
  AND2X1 main_gate ( .A(net350), .B(CLK), .Y(ENCLK) );
endmodule
```

Gate-level Netlist: Integrated CG Cell from TSMC

```
module SNPS_CLOCK_GATE_HIGH_top_2 ( CLK, EN, ENCLK );
  input CLK, EN;
  output ENCLK;


  TLATNCAX2 latch ( .E(EN), .CK(CLK), .ECK(ENCLK) );
endmodule
```

Better Power/Performance/Area



Other Special Circuits

- Multi-cycle path
- Positive & negative edge clock trigger
- Multiple clocks design

Setting Power Constraint

- Make DC to minimize power for the design
- These commands are deprecated, DC minimizes power by default

```
set_max_dynamic_power 0 uw
```

```
set_max_leakage_power 0 uw
```

Setting Area Constraint

- ***Attributes / Optimization Constraints / Design Constraints***
- Area unit (find the unit definition in technology library):
  - Equivalent gate counts
    - Gate counts = Chip Area / NAND2 Area
  - um x um
  - Transistor count (number of transistors)
- If we want to minimize the chip area

```
set_max_area 0
```

Cell Size **(In TSMC CL018G process)**

Drive Strength	Height (μm)	Width (μm)
NAND2XL	5.0	2.0
NAND2X1	5.0	2.0
NAND2X2	5.0	3.3
NAND2X4	5.0	4.6

Constraints Priority

- During the optimization, there exists a constraint priority relationship
  - Design Rule Constraints
    (max_transition, max_fanout, max_capacitance)
  - Timing constraints (max_delay, min_delay)
  - Power
  - Area

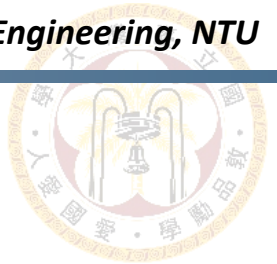
- There exists a trade-off between constraints

- Use `set_cost_priority` to modify the order (optional)

```
set_cost_priority -default
```

```
set_cost_priority -delay
```

```
set_cost_priority {max_fanout max_capacitance max_delay}
```



Check Design/Check Timing

- After you set up the deign attributes & design constraints, we recommend the next step is to check design

```
check_design > Report/check_design.txt
```

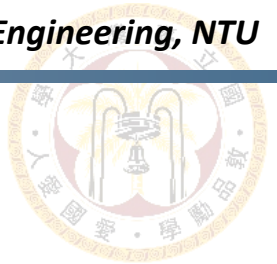
- Verify there are no remaining unconstrained paths

```
check_timing > Report/check_timing.txt
```

- ***Design/Check Design***

Example of Check Design

```
****************************************
check_design summary:
Version:     R-2020.09-SP5
Date:        Tue Dec 20 17:14:21 2022
****************************************

                  Name                                                    Total
--------------------------------------------------------------------------------
Inputs/Outputs                                                             2
    Unconnected ports (LINT-28)                                            2

Cells                                                                      7
    Cells do not drive (LINT-1)                                            7
--------------------------------------------------------------------------------

Warning: In design 's4_BUU_unfold', cell 'B_32' does not drive any nets. (LINT-1)
Warning: In design 's4_BUU_unfold', cell 'B_33' does not drive any nets. (LINT-1)
Warning: In design 's4_BUU_unfold', cell 'B_34' does not drive any nets. (LINT-1)
Warning: In design 's4_BUU_unfold', cell 'B_35' does not drive any nets. (LINT-1)
Warning: In design 's4_BUU_unfold', cell 'B_36' does not drive any nets. (LINT-1)
Warning: In design 's4_BUU_unfold', cell 'B_37' does not drive any nets. (LINT-1)
Warning: In design 's4_BUU_unfold', cell 'B_38' does not drive any nets. (LINT-1)
Warning: In design 's4_BUU_unfold', port 'i_enable' is not connected to any nets. (LINT-28)
Warning: In design 's4_BUU_unfold', port 'i_clear' is not connected to any nets. (LINT-28)
1
```



Example of Check Timing

```
Information: Checking out the license 'DesignWare'. (SEC-104)
Information: Updating design information... (UID-85)

Information: Checking generated_clocks...

Information: Checking loops...

Information: Checking no_input_delay...

Information: Checking unconstrained_endpoints...

Information: Checking pulse_clock_cell_type...

Information: Checking no_driving_cell...

Information: Checking partial_input_delay...
1
```

Synthesis Design Flow

- Optimization & mapping to gate-level netlist
  - Solving multiple instances
  - Compile Options
  - Commands to compile

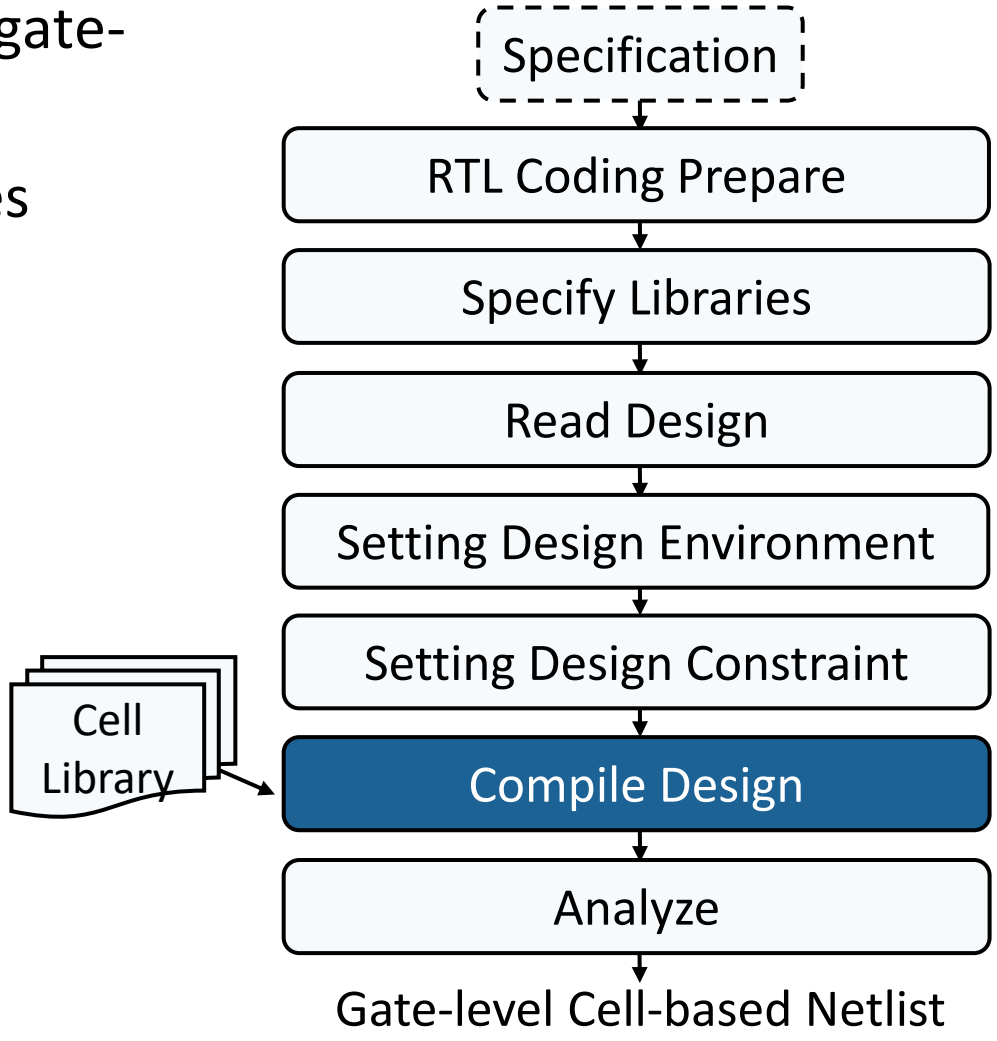
```
compile
```

or

```
compile -inc
```

or

```
compile_ultra
```

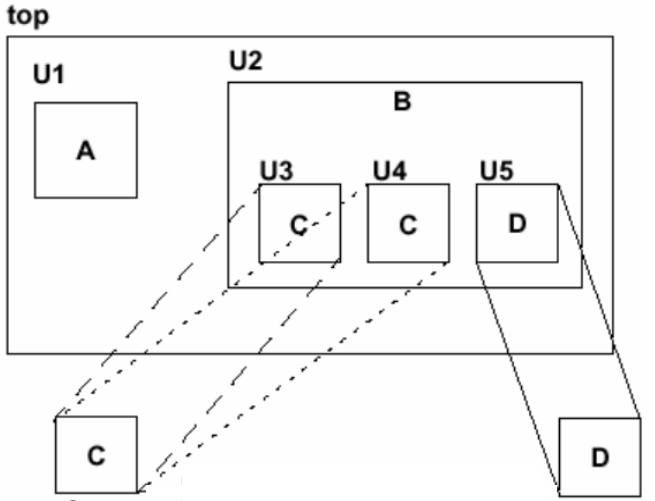


Solving Multiple Instances

- In a hierarchical design, submodules are sometimes referenced by more than one cell instance

- Methods for solving multiple instances
  - Compile-once-don't-touch
  - Uniquify
  - Ungroup

```
module B (
    ...
    C U3(.pin1(w1), .pin2(w2));
    C U4(.pin1(w3), .pin2(w4));

    D U5(.pinA(w5), .pinB(w6));
    ...
endmodule
```

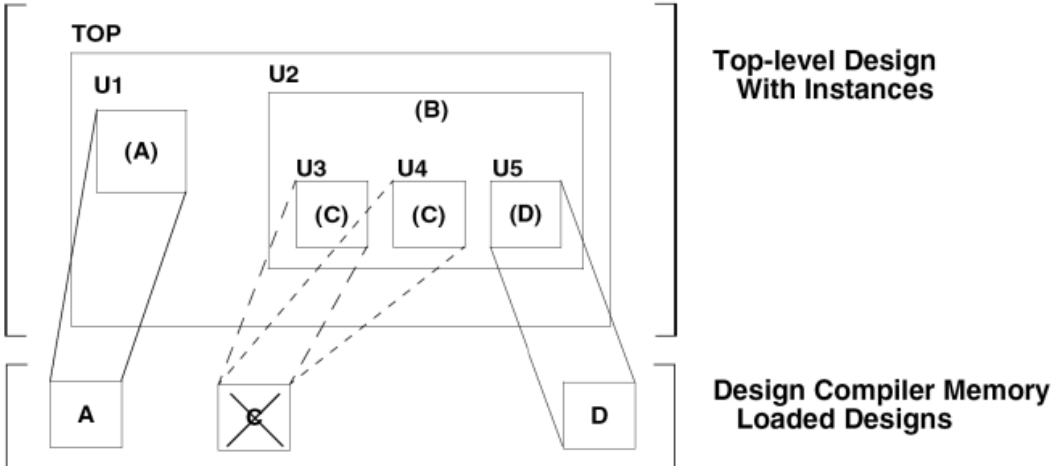


Module C Referenced Twice

1. Compile-once-don't-touch Method

- Two steps:
  - **Pre-compile the submodule**
  - Set submodules as **don't touch** for compiling entire design
- ***Attributes/Optimization Directives/Design* & set Don't Touch**

- Pros
  - Less memory
  - Shorter compile time



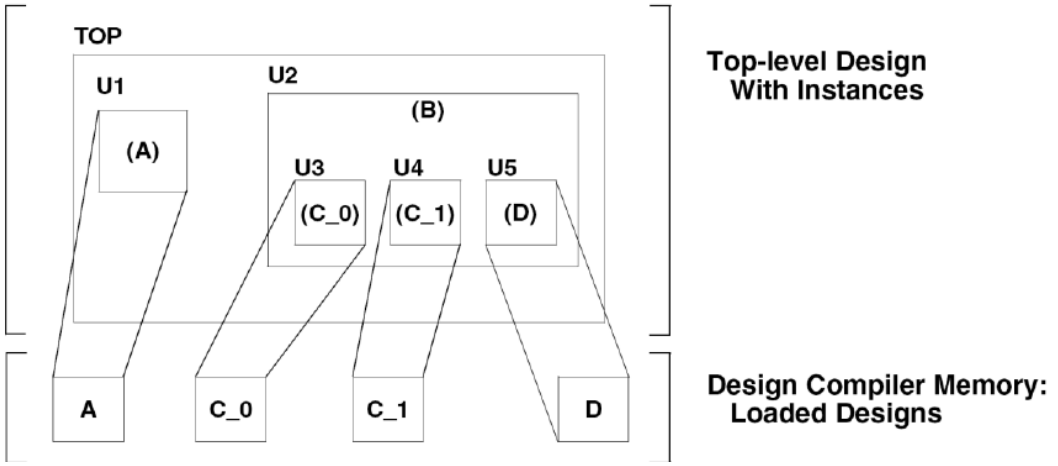

```
set_dont_touch [$Cell_list]
```

2. Uniquify Method

- Create an unique design for each instance, even if it is resolved from identical reference
- **Hierarchy of design is preserved**
- ***Hierarchy/Uniquify/Hierarchy***

- Pros and cons
  - More memory
  - Longer compile time
  - Better performance

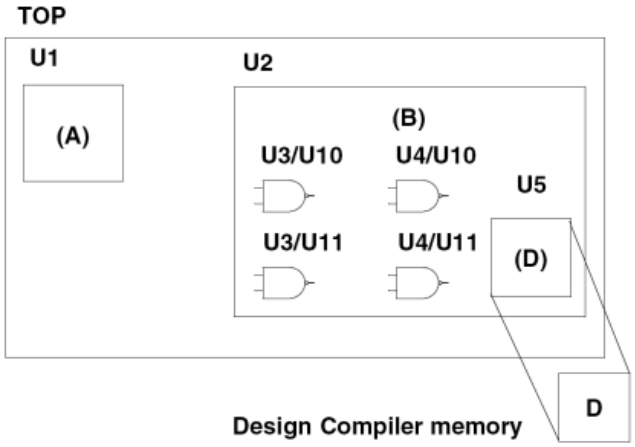
```
uniquify
```



3. Ungroup Method

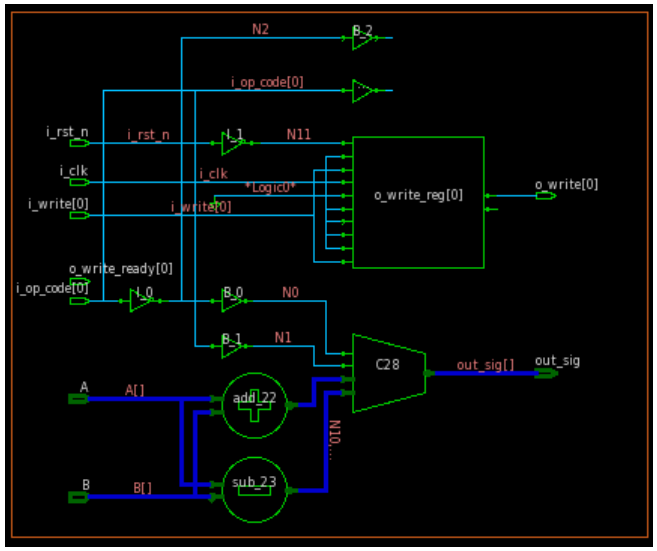
- The same effect as uniquify but it removes hierarchical levels
- **Hierarchy of design is broken**.
- ***Attributes/Optimization Directives/Design & set the Ungroup***
- Pros and cons
  - More memory
  - Longer compile time
  - Better performance
  - Removing user-defined design hierarchy

```
ungroup [$Cell_list]
```



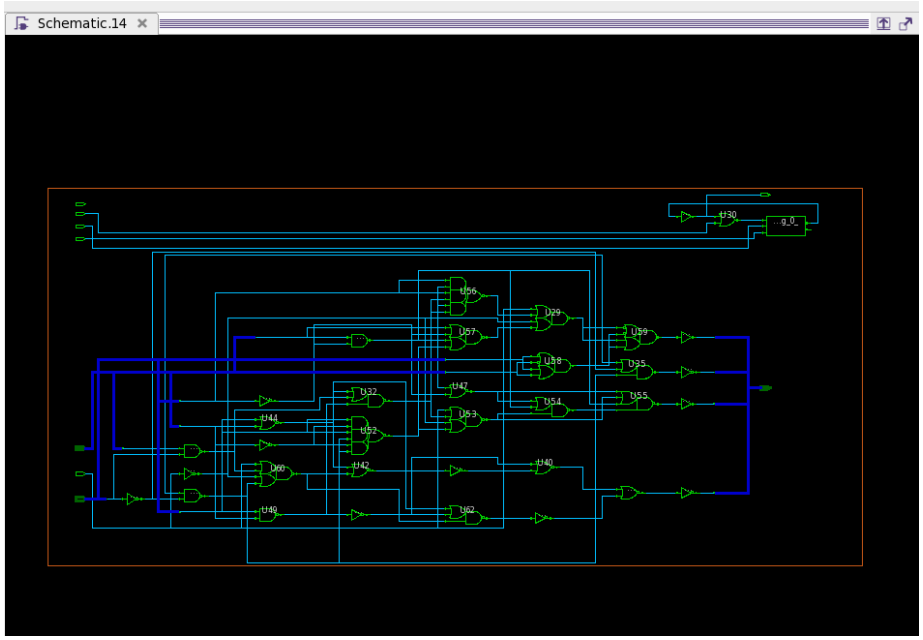
Compile Design

- Maps GTECH to **gate-level netlist**
  - Using cells from user-specified technology library (e.g., 0.18um TSMC)
  - Optimized with area/timing constraints



GTECH

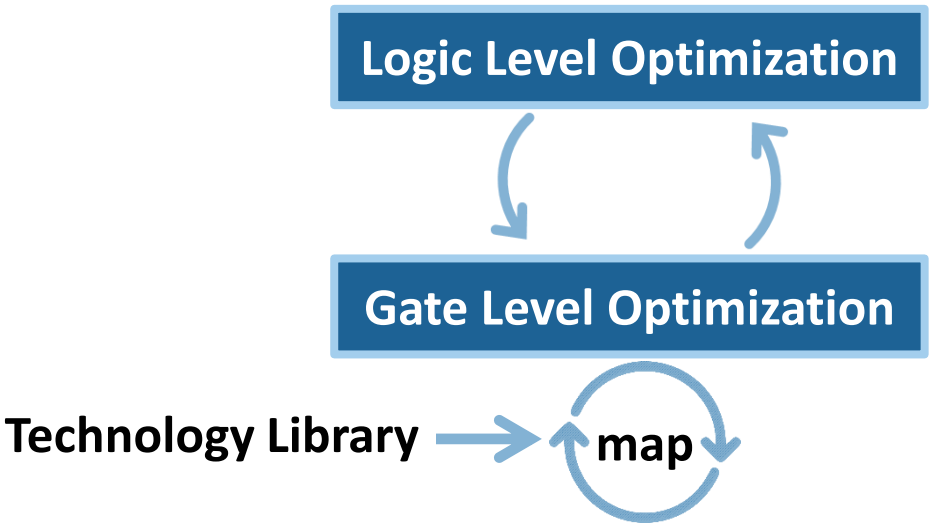
Compile



Gate-level Netlist

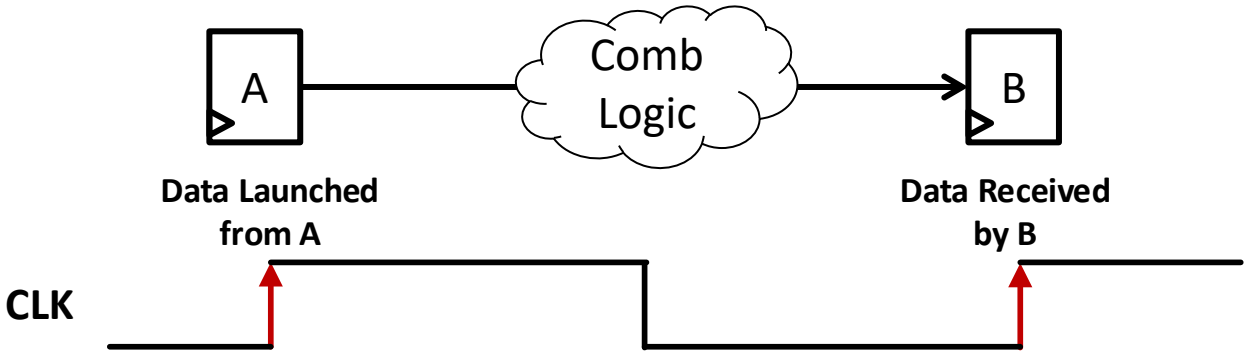
Compile Concept: Optimization

- Constraint-driven optimization, not exhaustive search
- Logic level optimization
  - Structuring: group subfunctions & intermediate variables
  - Flattening: break logics to 2-level, sum-of-products representation
- Gate level optimization (using **technology library**)
  - Generates netlist from gates in target library, check timing & area goals



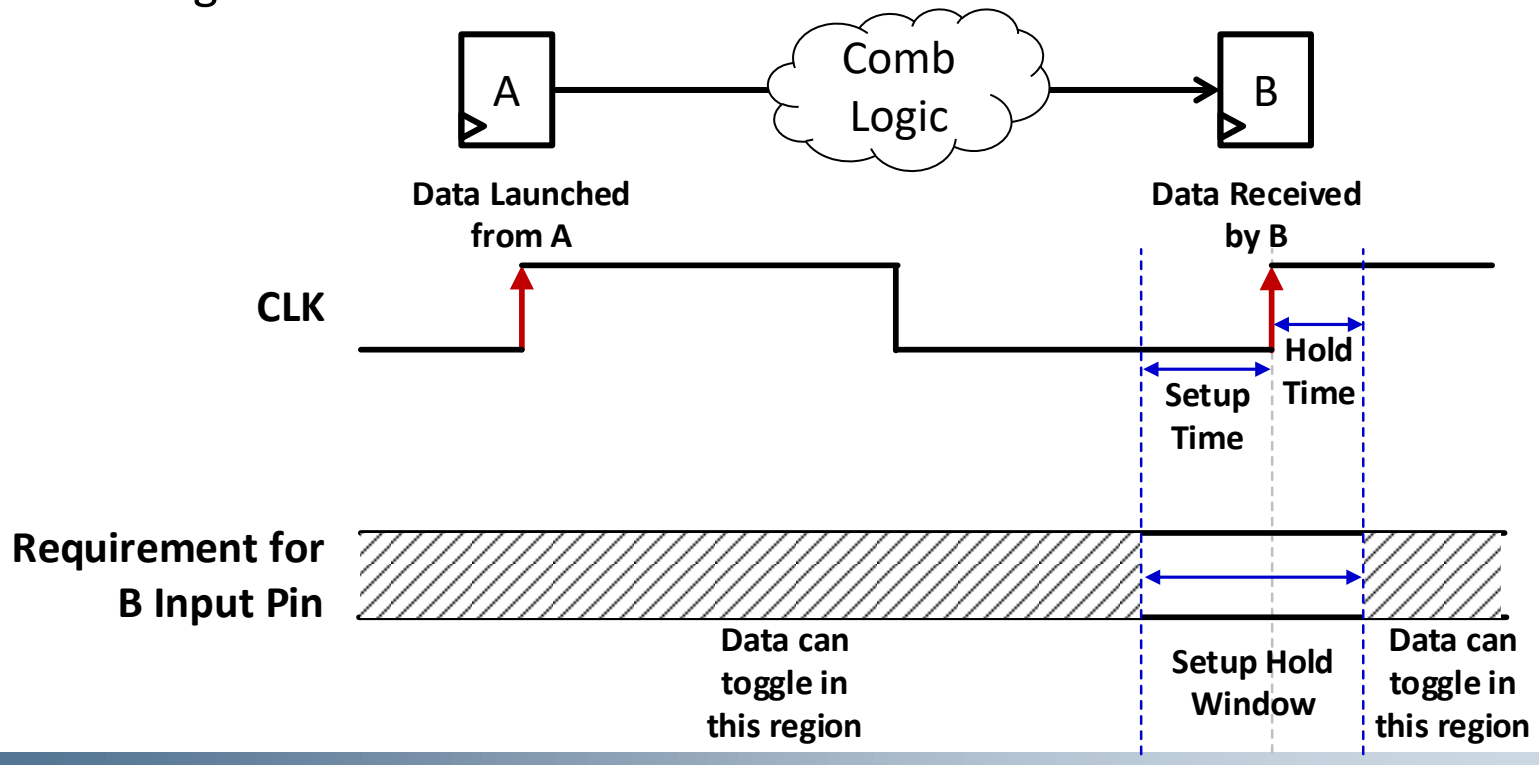
Compile Concept: Timing Requirements

- Each path in the design is analyzed for timing requirements
  - **Setup time**: the minimum amount of time before clock's active edge that the data must be stable.
  - **Hold time**: the minimum amount of time after the clock's active edge during which data must be stable.



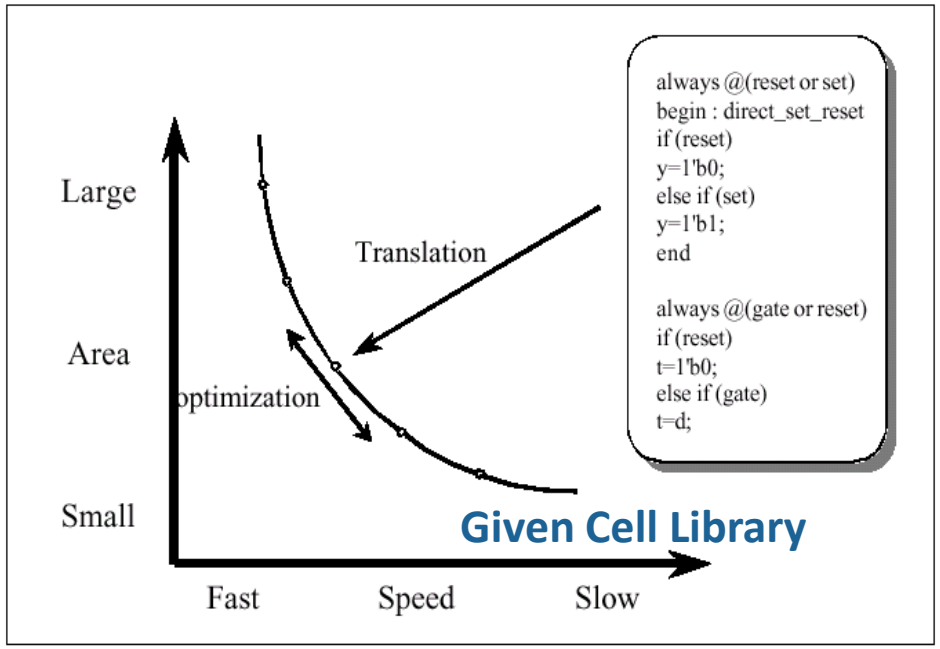
Compile Concept: Timing Requirements

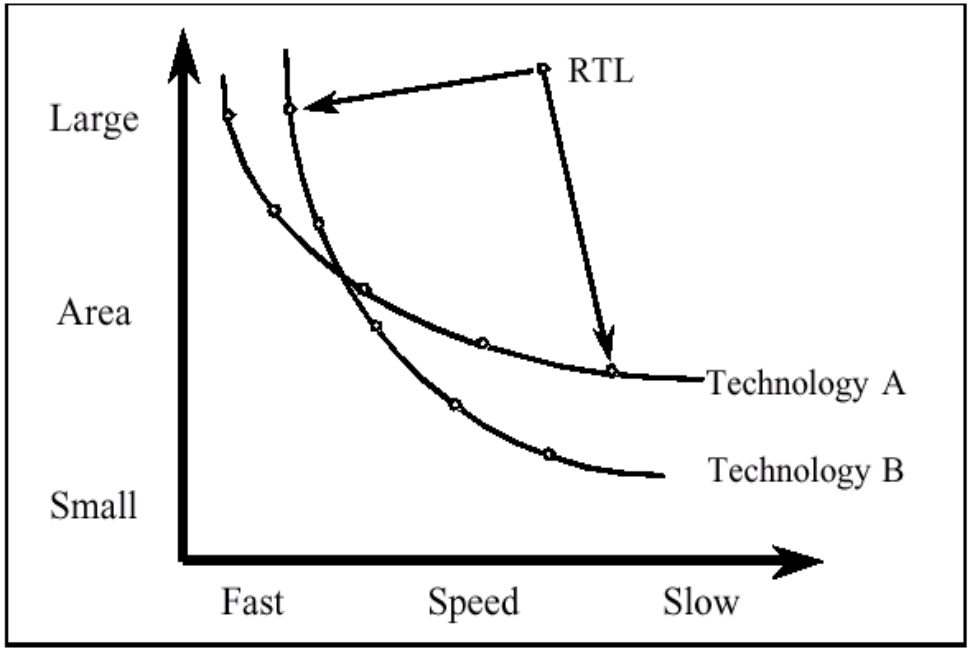
- Each path in the design is analyzed for timing requirements
  - **Setup time**: the minimum amount of time before clock's active edge that the data must be stable.
  - **Hold time**: the minimum amount of time after the clock's active edge during which data must be stable.



Recall: Trade-off between Constraints

- Given the same library and the same source code, you can
  - Sacrifice area for higher speed
  - Sacrifice speed for lower area





Option: Retiming Method

- Enable DC to **move registers through combinational logics**
- Pros and cons
  - More balanced pipeline for better area & timing
  - Opportunity to reduce number of registers
  - Longer compilation time for large design

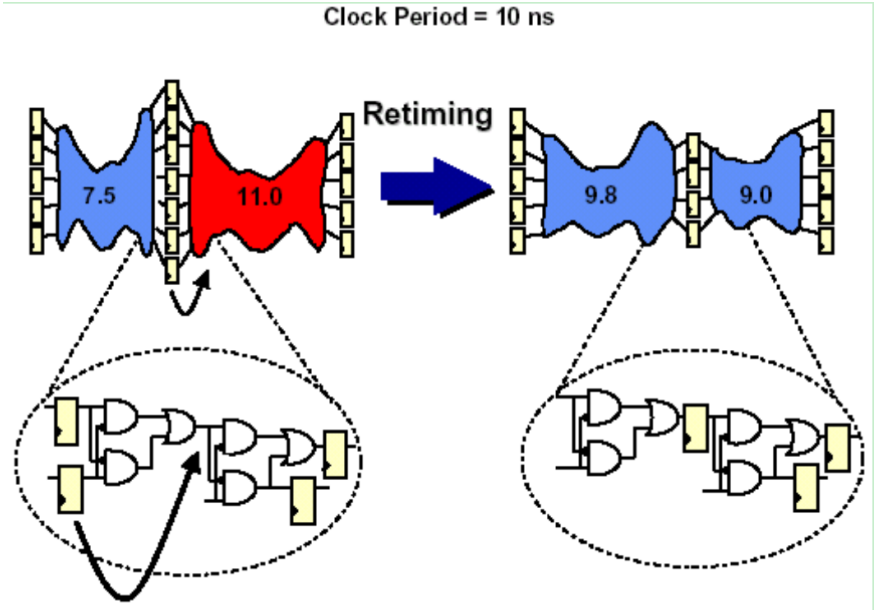


Image source: http://blog.chinaaet.com/justlxy/p/5100052114

Option: Leakage Power Optimization

- **Multi-threshold voltage (V_t)** libraries to **reduce leakage power**
  - Low V_t cells: higher speed, higher leakage
  - High V_t cells: lower speed, lower leakage
- Note: Different doping concentration for high & low V_t MOSFETs
  - High V_t: Lower source-drain current → slower, lower leakage

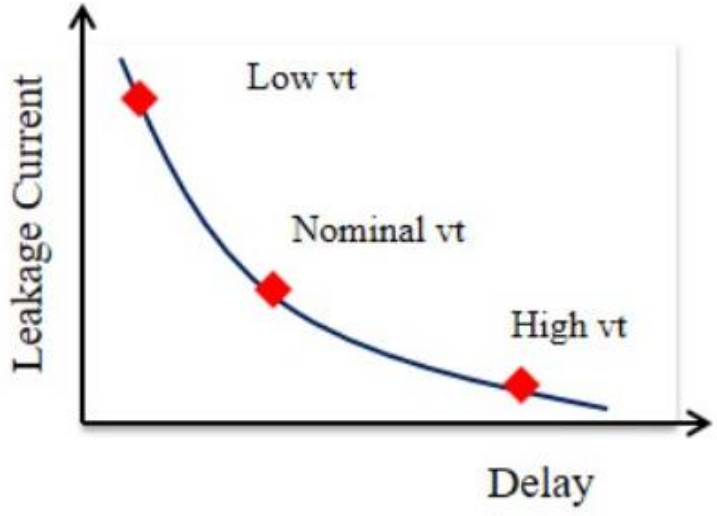
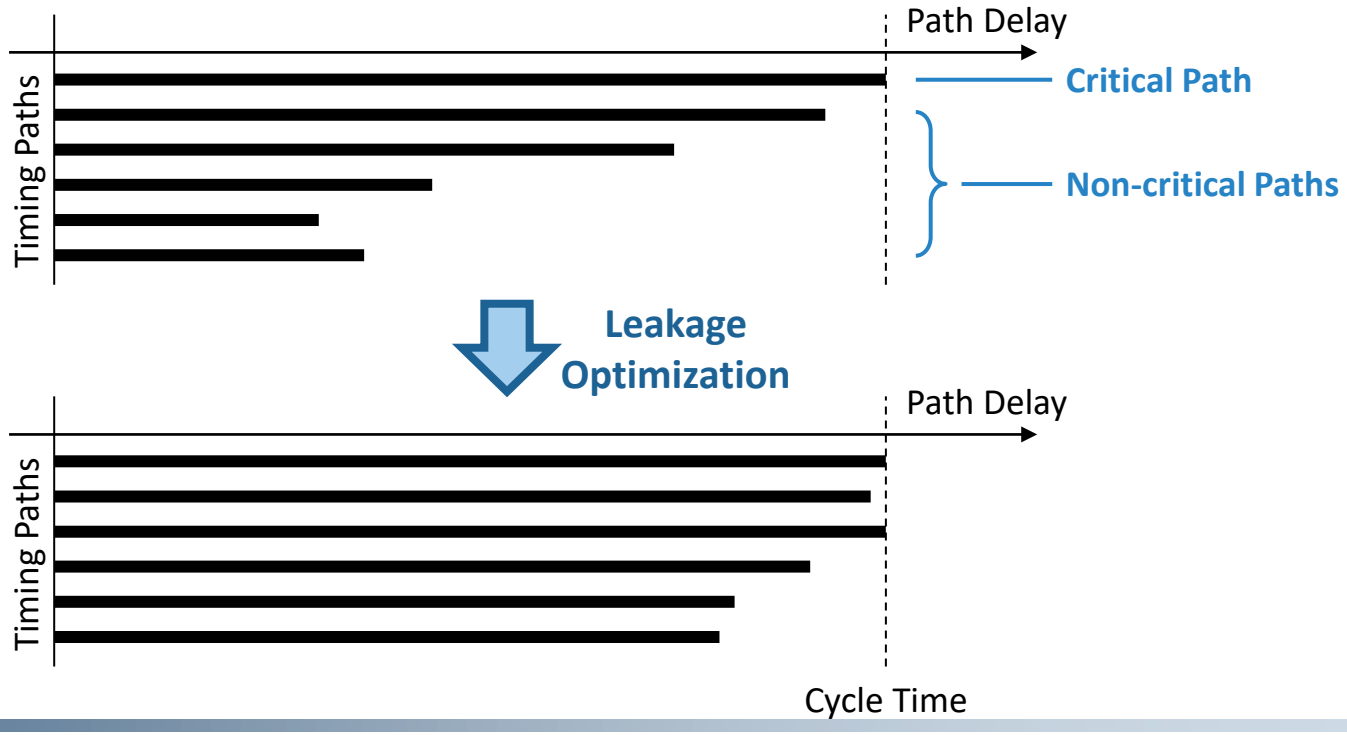
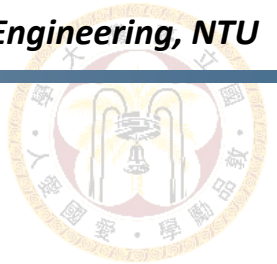


Image source: https://www.vlsiguru.com/low-power-pavan/

Option: Leakage Power Optimization

- Trade-off between timing & power
  - Use **low V_t** cells on critical paths to **improve timing**
  - Use **high V_t** cells on non-critical paths to **save power**
- Usually optimize leakage power after timing optimization





Commands to Compile (1/3)

- Compile Mapping effort levels
  - Low: quick synthesis, does not do all algorithms
  - Medium: default setting, good for many design
  - High: highest effort, sometimes hard to converge

```
compile
```

```
compile -map_effort high
```

- Compile ultra
  - Using high effort
  - Enable several optimization options
    - E.g., auto-ungroup, boundary optimization

```
compile_ultra
```

Commands to Compile (2/3)

- Incremental mapping
  - Perform gate-level optimization only
  - Can be used to optimize current gate-level netlist

```
compile -inc
```

- Retiming

```
set_optimize_registers
compile_ultra -retime
```

```
compile
optimize_registers
```

```
compile
optimize_registers -no_compile
compile_ultra -inc
```

Commands to Compile (3/3)

- Compile with leakage optimization

```
# in .synopsys_dc.setup
set target_library "slow_hvt.db fast_hvt.db \
                    slow_rvt.db fast_rvt.db"
```

```
compile
set_leakage_optimization true
compile -inc
```

- Other options can be checked by

```
compile -help
```

```
compile_ultra -help
```

Compile Process

- Recall: constraint-driven
  - We can roughly see the **trade-off between the constraints** and their priorities

```
 ELAPSED          WORST NEG   TOTAL SETUP  DESIGN RULE                    LEAKAGE POWER   MIN DELAY COST
  TIME     AREA    SLACK       COST         COST          ENDPOINT
--------- -------- --------- ------------ ------------ ------------------ --------------- --------------
  0:00:09  24208.3      0.07          7.2          0.0                      24438066.0000           0.00
  0:00:09  24208.3      0.07          7.2          0.0                      24438066.0000           0.00

Beginning Constant Register Removal
-----------------------------------

Beginning Global Optimizations
------------------------------
Numerical Synthesis (Phase 1)
Numerical Synthesis (Phase 2)
Global Optimization (Phase 1)
Global Optimization (Phase 2)
Global Optimization (Phase 3)
Global Optimization (Phase 4)

Beginning Isolate Ports
-----------------------
```

Trade-off between constraints

```
Beginning Delay Optimization
----------------------------
  0:00:10  15865.6      0.51         48.3         37.6                      15621728.0000           0.00
  0:00:12  16744.9      0.13         13.0         19.3                      16497935.0000           0.00
  0:00:12  16744.9      0.13         13.0         19.3                      16497935.0000           0.00
  0:00:12  16733.0      0.14         13.4          0.0                      16813510.0000           0.00
  0:00:12  16733.0      0.14         13.4          0.0                      16813510.0000           0.00
  0:00:12  16733.0      0.14         13.4          0.0                      16813510.0000           0.00

Beginning WLM Backend Optimization
----------------------------------
  0:00:13  16673.6      0.14         13.9          0.0                      16743739.0000           0.00
  0:00:13  16673.6      0.14         13.9          0.0                      16743739.0000           0.00
  0:00:13  16673.6      0.14         13.9          0.0                      16743739.0000           0.00
  0:00:13  16783.9      0.14         13.6          0.0                      16824478.0000           0.00
  0:00:13  16783.9      0.14         13.6          0.0                      16824478.0000           0.00
  0:00:13  17700.5      0.12         10.9          0.0                      17591000.0000           0.00
  0:00:14  17700.5      0.12         10.9          0.0                      17591000.0000           0.00
  0:00:14  17748.0      0.12         10.5          0.0                      17656992.0000           0.00
  0:00:14  17748.0      0.12         10.5          0.0                      17656992.0000           0.00
  0:00:14  17748.0      0.12         10.5          0.0                      17656992.0000           0.00
  0:00:14  17748.0      0.12         10.5          0.0                      17656992.0000           0.00
  0:00:16  18335.3      0.08          6.1          6.6                      18312380.0000           0.00
  0:00:16  18335.3      0.08          6.1          6.6                      18312380.0000           0.00
```

Solving Setup/Hold Time Violation

- Solving setup time violation
  - Increase compile effort/use compile_ultra
  - Apply additional incremental compile (gate-level opt. only)

```
compile -inc -map_effort high
```

- Solving hold time violation
  - Perform only hold time fixing, ignoring other rules
  - Note: `set_fix_hold` command must be specified

```
set_fix_hold [all_clocks]
compile -inc -only_hold_time
```

How to Fix Assignment Error

```
assign ABSVAL[19] = A[19];
assign ABSVAL[18] = A[18];
assign ABSVAL[17] = A[17];
assign ABSVAL[16] = A[16];
assign ABSVAL[15] = A[15];
```

```
BUFX1 X37X( .I(A[19]), .Z(ABSVAL[19]) );
BUFX1 X38X( .I(A[18]), .Z(ABSVAL[18]) );
BUFX1 X39X( .I(A[17]), .Z(ABSVAL[17]) );
BUFX1 X40X( .I(A[16]), .Z(ABSVAL[16]) );
BUFX1 X41X( .I(A[15]), .Z(ABSVAL[15]) );
```

- Sometimes there are still some assign syntax in gate level RTL
- The syntax of “assign” may cause problems in the layout versus schematic verification (LVS)

```
set_fix_multiple_port_nets -all -buffer_constants [get_designs *]
```

Change Naming Rule

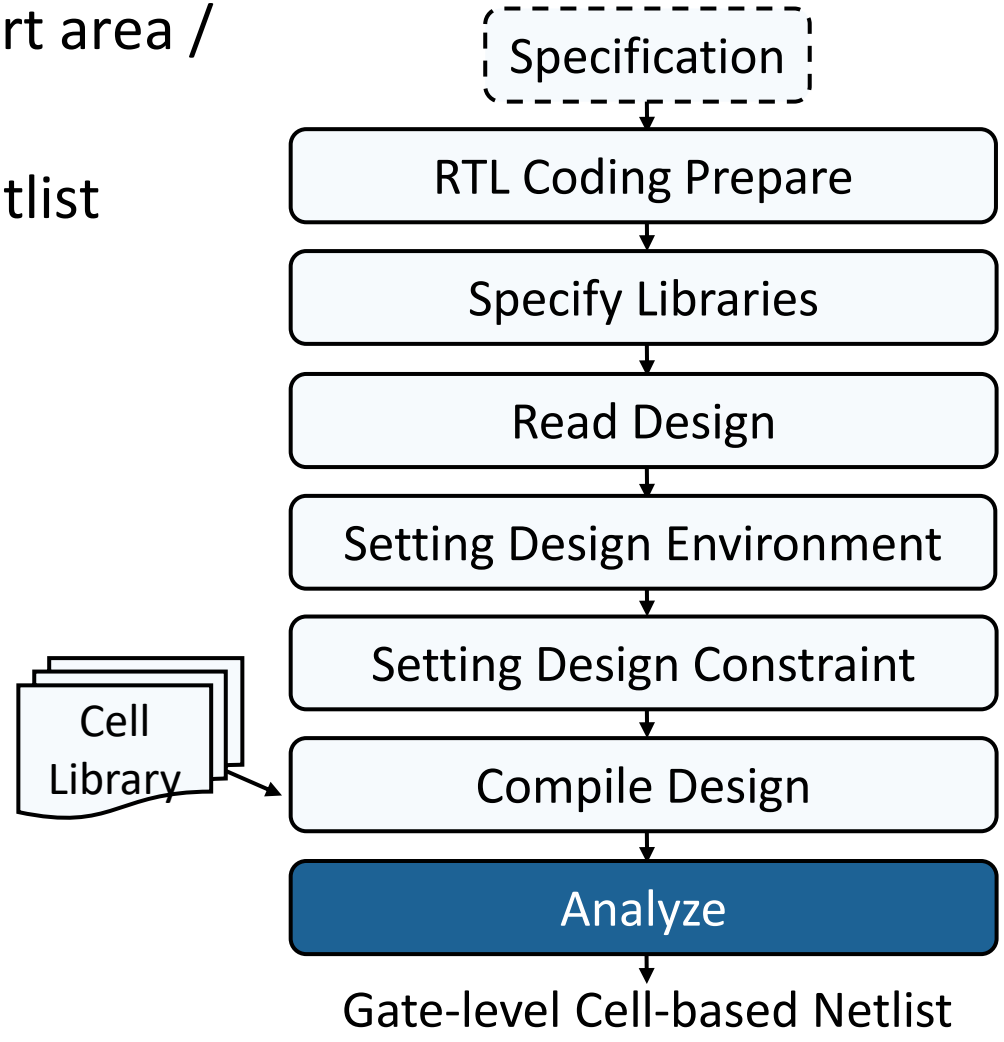
- Let the naming-rule definitions in the gate-level netlist are the same as in the timing file
- Wrong naming rules may cause problems in the layout versus schematic verification (LVS)

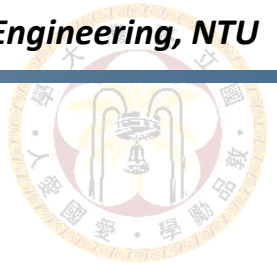
```
set bus_inference_style "%s\[%d\]"
set bus_naming_style "%s\[%d\]"
set hdlout_internal_busses true
change_names -hierarchy -rule verilog
define_name_rules name_rule -allowed "a-z A-Z 0-9 _" -max_length 255 -type cell
define_name_rules name_rule -allowed "a-z A-Z 0-9 _[]" -max_length 255 -type net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

Synthesis Design Flow

- Check design and then report area / constraints / timing issue
- Output files of gate-level netlist

```
check_design
report_area
report_constraint
report_timing

write_sdc
write_sdf
write -format verilog
```





Report Your Synthesis Results

- Timing
  - Shows maximum or minimum delay path of design
  - Specify number of paths to show in `-max_paths`

```
report_timing -delay max -max_paths 5 > Design.timing
```

- Area
  - Shows the area in um^2 of the design

```
report_area -nosplit > Design.area
```

```
report_area -hierarchy > Design.area_hier
```

- Power
  - Report both dynamic and static power
  - PrimeTime is more accurate

```
report_power > Design.power
```

Timing Report

- Reports the timing info of the longest/shortest path in design
- We check if there's negative slack (violated timing path)

```
1
2  ****************************************
3  Report : timing
4           -path full
5           -delay max
6           -max_paths 5
7  Design : alu
8  Version: N-2017.09-SP2
9  Date   : Fri Oct 16 16:09:07 2020
10 ****************************************
11
12 Operating Conditions: slow     Library: slow
13 Wire Load Model Mode: top
14
15   Startpoint: i_inst[0] (input port clocked by i_clk)
16   Endpoint: o_data_reg[1]
17             (rising edge-triggered flip-flop clocked by
18   Path Group: i_clk
19   Path Type: max
20
21   Des/Clust/Port     Wire Load Model       Library
22   ------------------------------------------------
23   alu                tsmc13_wl10           slow
24
```

```
21  Point                                   Incr       Path
22  ------------------------------------------------------------
23  clock i_clk (rise edge)                 0.00       0.00
24  clock network delay (ideal)             0.50       0.50
25  input external delay                    5.00       5.50 f
26  i_inst[0] (in)                          0.00       5.50 f
27  U188/Y (CLKBUFX3)                       0.11       5.61 f
28  U186/Y (NAND2X1)                        0.08       5.69 r
29  U152/Y (CLKINVX1)                       0.05       5.74 f
30  U150/Y (NAND2X1)                        0.05       5.79 r
31  U148/Y (OA21XL)                         0.11       5.91 r
32  U141/Y (OAI21XL)                        0.05       5.96 f
33  U140/Y (NAND2X1)                        0.11       6.07 r
34  U175/Y (OAI21XL)                        0.06       6.13 f
35  U216/Y (AOI221XL)                       0.20       6.32 r
36  U214/Y (NAND2X1)                        0.04       6.36 f
37  o_data_reg[2]/D (DFFRX1)                0.00       6.36 f
38  data arrival time                                  6.36
39
40  clock i_clk (rise edge)                10.00      10.00
41  clock network delay (ideal)             0.50      10.50
42  clock uncertainty                      -0.10      10.40
43  o_data_reg[2]/CK (DFFRX1)               0.00      10.40 r
44  library setup time                     -0.13      10.27
45  data required time                                10.27
46  ------------------------------------------------------------
47  data required time                                10.27
48  data arrival time                                 -6.36
49  ------------------------------------------------------------
50  slack (MET)                                        3.91
```

Requirement: Slack ≥ 0

Area Report

- Reports area of combinational/sequential blocks
  - **Macro/black box** refers to hard macros
  - To fix undefined net interconnect area, **set wire load model**
- We check area of each part & total cell area
  - Net interconnect area by WLM only provides a rough estimation
  - More accurate after place and route

```
Library(s) Used:

    slow (File: /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db/slow.db)

Number of ports:                          44
Number of nets:                          282
Number of cells:                         243
Number of combinational cells:           170
Number of sequential cells:               72
Number of macros/black boxes:              0
Number of buf/inv:                        23
Number of references:                      2

Combinational area:                1834.889360
Buf/Inv area:                       308.926804
Noncombinational area:             2378.057323
Macro/Black Box area:                 0.000000
Net Interconnect area:            36633.547365

Total cell area:                   4212.946684
Total area:                       40846.494048
```

Hard macros, e.g., SRAM

Don't care net area!

Area Report

- Hierarchical area distribution
  - Reports area breakdown for all hierarchies in design
  - Note: auto ungroup (default in `compile_ultra`) breaks the hierarchy in logics
  - We can disable auto ungroup with `compile_ultra -no_autoungroup`

```
Hierarchical area distribution
------------------------------

                                 Global cell area          Local cell area
                                 ------------------  ----------------------------
Hierarchical cell                Absolute   Percent  Combi-     Noncombi-  Black-
                                 Total      Total    national   national   boxes   Design
-------------------------------- ---------  -------  ---------  ---------  ------  ---------
top                              4212.9467    100.0   203.6880     0.0000  0.0000  top
u_fifo_0                         4009.2587     95.2  1631.2014  2378.0573  0.0000  fifo
-------------------------------- ---------  -------  ---------  ---------  ------  ---------
Total                                                1834.8894  2378.0573  0.0000

1
```

`report_area -hierarchy`

Power Report

- Reports a rough power dissipation (fixed toggle rate)
- For a more accurate power estimation, we use
  - Gate-level simulation to generate waveform based on input pattern
  - PrimeTime to estimate power dissipation **with waveform**

```
Design          Wire Load Model            Library
------------------------------------------------
alu                tsmc13_wl10             slow


Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =  96.1401 uW   (53%)
  Net Switching Power  =  86.3698 uW   (47%)
                         ---------
Total Dynamic Power    = 182.5099 uW  (100%)

Cell Leakage Power     =   1.1657 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register       7.7090e-02       8.0832e-03        2.4367e+05         8.5416e-02  (  46.50%)
sequential         0.0000           0.0000            0.0000             0.0000  (   0.00%)
combinational  1.9051e-02       7.8287e-02        9.2205e+05         9.8259e-02  (  53.50%)
------------------------------------------------------------------------------------------------
Total          9.6140e-02 mW     8.6370e-02 mW     1.1657e+06 pW      0.1837 mW
1
```

Leakage power will become more dominant in the advanced process

Save Design

- Synopsys Design Constraints (.sdc)
- Standard Delay Format (.sdf)
- Gate-Level Netlist (.v)

Required in gate-level simulation

- Test protocol file for ATPG tools (.spf)
- Synopsys encrypted binary file (*.ddc)
  - Record the constraints and synthesis results

Synopsys Design Constraints (.sdc)

- A format used to **specify the design intent**, including the timing, power and area constraints for a design

- SDC is TCL based

```
set DESIGN "alu"
write_sdc  Netlist/$DESIGN\_SYN.sdc -version 1.8
```

.sdc Example

```
1  ##################################################################
2
3  # Created by write_sdc on Sun Oct 25 11:11:28 2020
4
5  ##################################################################
6  set sdc_version 2.0
7
8  set_units -time ns -resistance kOhm -capacitance pF -voltage V -current mA
9  set_operating_conditions typical -library typical
10 set_wire_load_mode segmented
11 set_wire_load_model -name ForQA -library typical
12 set_max_transition 1 [current_design]
13 set_max_fanout 8 [current_design]
14 set_max_area 0
15 create_clock [get_ports clk]  -period 10  -waveform {0 5}
16 set_input_delay -clock clk  2.5  [get_ports {inputA[7]}]
17 set_input_delay -clock clk  2.5  [get_ports {inputA[6]}]
18 set_input_delay -clock clk  2.5  [get_ports {inputA[5]}]
19 set_input_delay -clock clk  2.5  [get_ports {inputA[4]}]
20 set_input_delay -clock clk  2.5  [get_ports {inputA[3]}]
21 set_input_delay -clock clk  2.5  [get_ports {inputA[2]}]
22 set_input_delay -clock clk  2.5  [get_ports {inputA[1]}]
23 set_input_delay -clock clk  2.5  [get_ports {inputA[0]}]
24 set_input_delay -clock clk  3.8  [get_ports {inputB[7]}]
25 set_input_delay -clock clk  3.8  [get_ports {inputB[6]}]
26 set_input_delay -clock clk  3.8  [get_ports {inputB[5]}]
27 set_input_delay -clock clk  3.8  [get_ports {inputB[4]}]
28 set_input_delay -clock clk  3.8  [get_ports {inputB[3]}]
29 set_input_delay -clock clk  3.8  [get_ports {inputB[2]}]
30 set_input_delay -clock clk  3.8  [get_ports {inputB[1]}]
31 set_input_delay -clock clk  3.8  [get_ports {inputB[0]}]
32 set_input_delay -clock clk  4.5  [get_ports {instruction[3]}]
33 set_input_delay -clock clk  4.5  [get_ports {instruction[2]}]
34 set_input_delay -clock clk  4.5  [get_ports {instruction[1]}]
35 set_input_delay -clock clk  4.5  [get_ports {instruction[0]}]
36 set_input_delay -clock clk  5.2  [get_ports reset]
37 set_output_delay -clock clk  8  [get_ports {alu_out[7]}]
38 set_output_delay -clock clk  8  [get_ports {alu_out[6]}]
39 set_output_delay -clock clk  8  [get_ports {alu_out[5]}]
40 set_output_delay -clock clk  8  [get_ports {alu_out[4]}]
41 set_output_delay -clock clk  8  [get_ports {alu_out[3]}]
42 set_output_delay -clock clk  8  [get_ports {alu_out[2]}]
43 set_output_delay -clock clk  8  [get_ports {alu_out[1]}]
44 set_output_delay -clock clk  8  [get_ports {alu_out[0]}]
```

```
set_operating_conditions "typical library "typical"
set_wire_load_mode "segmented“
set_wire_load_model -name "ForQA" library "typical"
set_max_transition 1 ALU
set_max_fanout 8 ALU
set_max_area 0
```

```
create_clock -name "clk -period 10 -waveform {"0" "5"} {"clk"}
set_dont_touch_network [find clock clk]
set_fix_hold clk
```

```
set_input_delay -clock clk 2.5 inputA[*]
set_input_delay -clock clk 3.8 inputB[*]
set_input_delay -clock clk 4.5 instruction[*]
set_input_delay -clock clk 5.2 reset
set_output_delay -clock clk 8 alu_out[*]
```

Standard Delay Format (.sdf)

- SDF format
  - IEEE standard for representation & interpretation of timing data for electronic design process

- SDF file generated in synthesis
  - Detailed **delay information of each cell** in gate-level netlist
  - Timing check of each cell (e.g., setup check, hold check)

```
write_sdf -version 2.1 -context verilog -load_delay
cell Netlist/$DESIGN\_SYN.sdf
```

SDF File Example

- Delay & timing check for each cell in design

```
(DELAYFILE
(SDFVERSION "OVI 2.1")
(DESIGN "adder")
(DATE "Tue Nov  1 22:48:09 2022")
(VENDOR "slow")
(PROGRAM "Synopsys Design Compiler cmos")
(VERSION "R-2020.09-SP5")
(DIVIDER /)
(VOLTAGE 1.08:1.08:1.08)
(PROCESS "slow")
(TEMPERATURE 125.00:125.00:125.00)
(TIMESCALE 1ns)
(CELL
  (CELLTYPE "adder")
  (INSTANCE)
  (DELAY
    (ABSOLUTE
    (INTERCONNECT U30/S U31/AN (0.000:0.000:0.000))
    (INTERCONNECT U24/Y U31/B (0.000:0.000:0.000))
    (INTERCONNECT in_a[3] U30/A (0.000:0.000:0.000))
    (INTERCONNECT in_b[3] U30/B (0.000:0.000:0.000))
    (INTERCONNECT U28/CO U30/CI (0.000:0.000:0.000))
    (INTERCONNECT U30/CO U29/AN (0.000:0.000:0.000))
    (INTERCONNECT U24/Y U29/B (0.000:0.000:0.000))
    (INTERCONNECT in_a[2] U28/A (0.000:0.000:0.000))
    (INTERCONNECT in_b[2] U28/B (0.000:0.000:0.000))
```

```
(CELL
  (CELLTYPE "BUFX12")
  (INSTANCE U13)
  (DELAY
    (ABSOLUTE
    (IOPATH A Y (0.701:0.701:0.701) (0.520:0.520:0.520))
    )
  )
)
(CELL
  (CELLTYPE "DFFQX1")
  (INSTANCE out_reg_0_)
  (DELAY
    (ABSOLUTE
    (IOPATH (posedge CK) Q (0.466:0.466:0.466) (0.346:0.346:0.346))
    )
  )
  (TIMINGCHECK
    (WIDTH (posedge CK) (0.176:0.176:0.176))
    (WIDTH (negedge CK) (0.249:0.249:0.249))
    (SETUP (posedge D) (posedge CK) (0.152:0.152:0.152))
    (SETUP (negedge D) (posedge CK) (0.265:0.283:0.283))
    (HOLD (posedge D) (posedge CK) (-0.107:-0.107:-0.107))
    (HOLD (negedge D) (posedge CK) (-0.088:-0.113:-0.113))
  )
)
```

Gate-Level Netlist (.v)

- Description of the connectivity of an electronic circuit, containing all of the logic and delays of the entire circuit
- Used in gate-level simulation
- Write out gate-level netlist

1. ***File/Save As ➔ Verilog***
2. `write -format verilog -output Netlist/$DESIGN\_SYN.v -hierarchy`

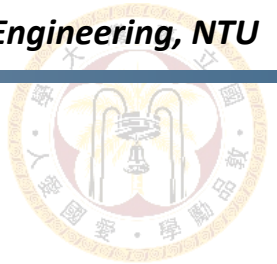
Gate-Level Netlist (.v) Example

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

RTL Design
counter.v

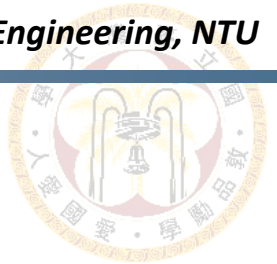
```
module counter ( i_clk, rstn, load, in, out );
  input [1:0] in;
  output [1:0] out;
  input i_clk, rstn, load;
  wire   n12, n13, N6, N7, n5, n8, n9, n10, n11;

  DFFQX1 out_reg_1_ ( .D(N7), .CK(i_clk), .Q(n12) );
  DFFQX1 out_reg_0_ ( .D(N6), .CK(i_clk), .Q(n13) );
  CLKINVX1 U9 ( .A(n13), .Y(n5) );
  INVX16 U10 ( .A(n5), .Y(out[0]) );
  BUFX16 U11 ( .A(n12), .Y(out[1]) );
  INVXL U12 ( .A(load), .Y(n9) );
  OAI21XL U13 ( .A0(in[0]), .A1(n9), .B0(rstn), .Y(n8) );
  AOI21XL U14 ( .A0(n9), .A1(out[0]), .B0(n8), .Y(N6) );
  AOI2BB2X1 U15 ( .B0(out[1]), .B1(out[0]), .A0N(out[1]),
                  .A1N(out[0]), .Y(n11) );
  OAI21XL U16 ( .A0(in[1]), .A1(n9), .B0(rstn), .Y(n10) );
  AOI2BB1X1 U17 ( .A0N(n11), .A1N(load), .B0(n10), .Y(N7) );
endmodule
```

Gate-level Netlist
counte_syn.v



Synthesis the Design Vision

- File/Read or File/Analyze + File/Elaborate
- Attributes – set up Design Environment & Goals (record in script sdc files)
- Analysis/Report – check if set up is OK
- Analysis/Check Design
- Tools/Design Optimization
- Analysis/Report
- File/Save



Outline

- Introduction
- RTL Coding Related to Synthesis
- Synopsys Environment
- Synthesis Design Flow
- **Gate-Level Simulation**
- Files Demo

Gate-Level Simulation

- Prepare the following files for gate-level simulation

File	Example Filename
Testbench	testbench.v
Gate-level netlist	chip_syn.v
SDF file	chip_syn.sdf
Cell model	tsmc13_neg.v

- Modify your testbench file to **include timing delay from SDF file**

```
$sdf_annotate ("chip_syn.sdf", top_module_instance_name);
```

- Gate level simulation with timing information:

```
ncverilog testbench.v chip_syn.v -v tsmc13_neg.v +access+r
```

```
vcs -full64 -debug_access -R +v2k +neg_tchk testbench.v
chip_syn.v tsmc13_neg.v
```

Including SDF File

- Without SDF file, simulator uses delay information from cell model (e.g., tsmc13_neg.v), which must be wrong
- The simulation can still be performed, but lots of **timing violations** may occur even if there's no negative slack in synthesis

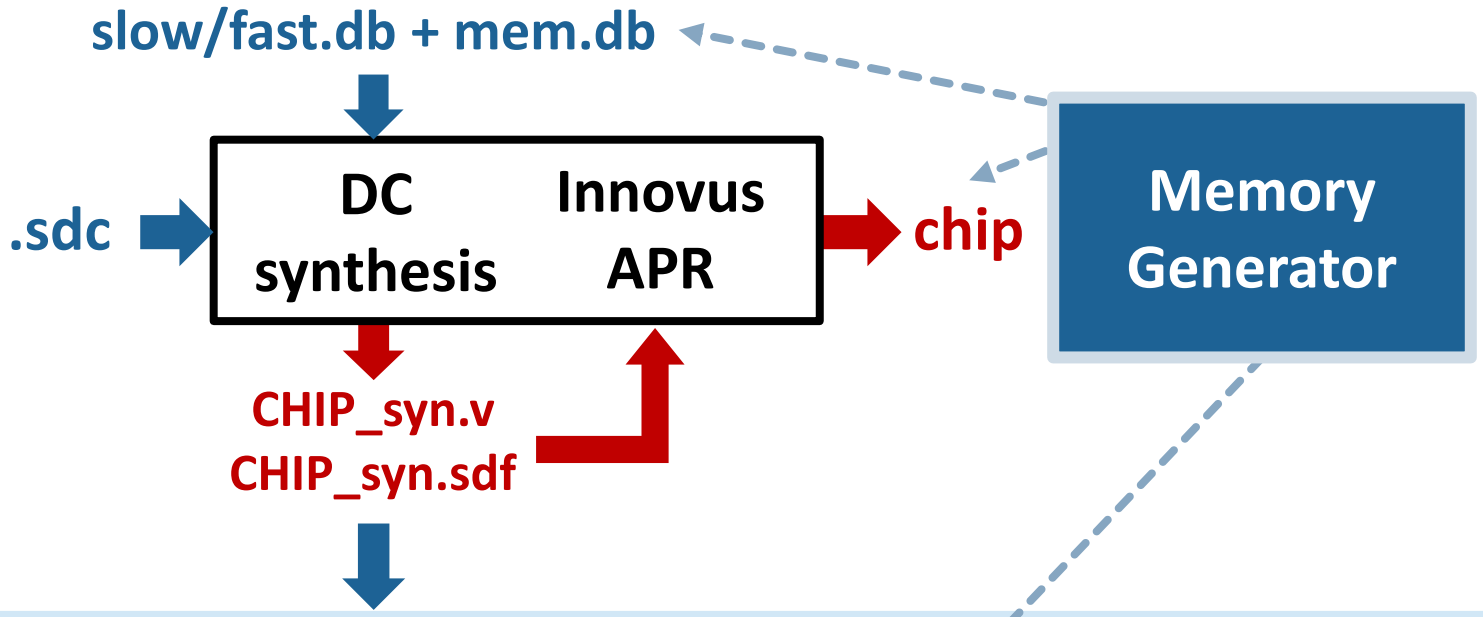
```
Warning!  Timing violation
          $setuphold<hold>( posedge CK &&& (flag == 1):6170 NS, posedge D:6170 NS,  1.000 : 1 NS,  0.500 : 500 PS );
           File: ./tsmc13_neg.v, line = 18416
          Scope: test.u_RFILE.yt_reg_4_
           Time: 6170 NS


Warning!  Timing violation
          $setuphold<hold>( posedge CK &&& (flag == 1):6170 NS, posedge D:6170 NS,  1.000 : 1 NS,  0.500 : 500 PS );
           File: ./tsmc13_neg.v, line = 18416
          Scope: test.u_RFILE.yt_reg_3_
           Time: 6170 NS


Warning!  Timing violation
          $setuphold<hold>( posedge CK &&& (flag == 1):6170 NS, posedge D:6170 NS,  1.000 : 1 NS,  0.500 : 500 PS );
           File: ./tsmc13_neg.v, line = 18416
          Scope: test.u_RFILE.yt_reg_2_
           Time: 6170 NS


Warning!  Timing violation
          $setuphold<hold>( posedge CK &&& (flag == 1):6170 NS, posedge D:6170 NS,  1.000 : 1 NS,  0.500 : 500 PS );
           File: ./tsmc13_neg.v, line = 18416
          Scope: test.u_RFILE.yt_reg_0_
           Time: 6170 NS


Warning!  Timing violation
```

Including SDF File

- With SDF file properly included, timing paths are denotated and can be checked in gate-level simulation

```
        Annotation completed with 0 Errors and 76 Warnings
        SDF statistics: No. of Pathdelays = 1629  Annotated = 100.00%  -- No. of Tchecks = 1386  Annotated = 93.94%
                                    Total      Annotated      Percentage
                 Path Delays         1629           1629          100.00
                      $width          630            630          100.00
                   $setuphold         756            672           88.89
        Building instance overlay tables: ..................... Done
        Generating native compiled code:
                worklib.fifo_DEPTH8_LENGTH8_0:v <0x044be911>
                        streams:   1, words:  1122
                worklib.fifo_DEPTH8_LENGTH8_1:v <0x2e6dc566>
                        streams:   1, words:  1122
                worklib.top:v <0x38b989e1>
                        streams:   1, words:   546
        Building instance specific data structures.
        Loading native compiled code:     ..................... Done
        Design hierarchy summary:
                                 Instances  Unique
                Modules:               517      84
                UDPs:                  238       2
                Primitives:           1549       8
                Timing outputs:        679      16
                Registers:             256      23
                Scalar wires:         1829       -
                Expanded wires:         38       3
                Always blocks:           2       2
                Initial blocks:          5       5
                Pseudo assignments:      8       8
                Timing checks:        2142     407
                Interconnect:         1329       -
                Delayed tcheck signals:  714     204
                Simulation timescale:    1ps
        Writing initial simulation snapshot: worklib.test:v
Loading snapshot worklib.test:v ..................... Done
*Verdi* Loading libsscore_ius152.so
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
```

Synthesis & Simulation w/ Memory File



Gate-level simulation

```
ncverilog tb.v CHIP_syn.v tsmc13.v mem.v +define+SDF +access+r
```

1) Behavior description
2) Default timing & timing check

Timing info. & timing check

Dump waveform authorization

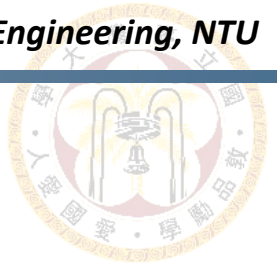
Pre/Post-layout Simulation w/ IO Pad

- For accurate simulation, I/O pads should be also considered for timing delay assumptions.
- The file ***tpz973g.v*** should be included for tsmc180 process

```
path:
/home/raid7_4/raid1_1/cic/CBDK018_TSMC_Artisan/orig_lib/fe_tpz973g_240b
/TSMCHOME/digital/Front_End/verilog/tpz973g_240b/
```

- For ncverilog simulation, you should use

```
ncverilog tb.v CHIP.v tpz973g.v tsmc18.v +define+SDF +access+r
```

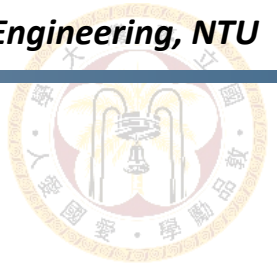


Outline

- Introduction
- RTL Coding Related to Synthesis
- Synopsys Environment
- Synthesis Design Flow
- Gate-Level Simulation
- **Files Demo**
  - Syn.tcl Example
  - SRAM File
  - Pad Example
  - Cell Model Example

Syn.tcl Example (1/7)

```
#================================================================================
#  .synopsys_dc setup
#================================================================================
#  Library Setting
#================================================================================
set search_path  {CAD_path/CBDK013_TSMC_Artisan/CIC/SynopsysDC/db \
                  /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/lib \
                  /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db \
                 $search_path }

set link_library      "typical.db slow.db fast.db dw_foundation.sldb"
set target_library    "typical.db slow.db fast.db"
set symbol_library    "generic.sdb"
set synthetic_library "dw_foundation.sldb"
```



Syn.tcl Example (2/7)

```
#=============================================================================
#  Synopsys Design Environmental Setting
#=============================================================================
# Treat text between translate statements as comments
set hdlin_translate_off_skip_text   {true}

# Enable Presto compiler for analyzing and elaborating VHDL/verilog design files,
# which allows high-level behavior structure such as extracting memory.
set hdlin_enable_presto_for_vhdl    {true}

# In order to generate compatible EDIF files
set edifout_netlist_only            {true}

# if inout used, tri net will be used
set verilogout_no_tri               {true}

set plot_command                    {lpr -Plp}
```

Syn.tcl Example (3/7)

```
#================================================================================
#  Read file and set current design and parameters
#================================================================================
set DESIGN "alu"
set period 10.0
read_verilog -rtl ../01_RTL/$DESIGN\.v
current_design $DESIGN

#================================================================================
#  Design Clock Constraints
#================================================================================
create_clock -name "i_clk" -period $period i_clk
set_fix_hold                [get_clocks i_clk]
set_dont_touch_network      [get_clocks i_clk]
set_ideal_network           [get_ports  i_clk]
set_clock_uncertainty  0.1  [get_clocks i_clk]
set_clock_latency      0.5  [get_clocks i_clk]
set_clock_transition   0.1  [get_clocks i_clk]
```

Remove these lines
Before APR

Syn.tcl Example (4/7)

```
#================================================================================
#  Synthesis Environmental Setting
#================================================================================
set t_in         [expr $period/2]
set t_out        [expr $period/2]

set_operating_conditions -max_library slow -max slow -min_library fast -min fast
set_wire_load_model -name tsmc13_wl10 -library slow
set_wire_load_mode top

set_load           0.05   [all_outputs]
set_drive          0.05   [all_inputs]
set_input_delay    $t_in  -clock i_clk [remove_from_collection [all_inputs] [get_ports i_clk]]
set_output_delay   $t_out -clock i_clk [all_outputs]

#================================================================================
#  Design Rule Constraints
#================================================================================
set_max_transition  0.1 -clock i_clk
set_max_capacitance 0.1 [get_ports *]
set_max_fanout 20       [all_inputs]
```

Remove these lines before APR

Synthesis w/ IO Pad

Syn.tcl Example (5/7)

```
#================================================================================
#  Optimization Objective (e.g. area, power...etc)
#================================================================================
set_max_area    0
#================================================================================

#================================================================================
# Check Design and Compile
#================================================================================
check_design > Report/check_design.txt
check_timing > Report/check_timing.txt

# Resolve multiple references
uniquify

# Fix Assignment Problem
set_fix_multiple_port_nets -all -buffer_constants [get_designs *]

# Fix Hold Time violation
set_fix_hold [all_clocks]

compile
```

Syn.tcl Example (6/7)

```
#====================================================================================
#  Report Synthesis Results
#====================================================================================
report_design                           >  Report/$DESIGN\.design
report_timing_requirements              >  Report/$DESIGN\.timingrequirement
report_constraint -all_violators        >  Report/$DESIGN\.constraint
report_timing -delay min -max_paths 5 >  Report/$DESIGN\.timing_min
report_timing -delay max -max_paths 5 >  Report/$DESIGN\.timing_max
report_area                             >  Report/$DESIGN\.area
report_resource -hierarchy              >  Report/$DESIGN\.resource
report_power                            >  Report/$DESIGN\.power

#====================================================================================
#  Change Naming Rule
#====================================================================================
set bus_inference_style "%s\[%d\]"
set bus_naming_style "%s\[%d\]"
set hdlout_internal_busses true
change_names -hierarchy -rule verilog
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"}}
change_names -hierarchy -rules name_rule
```

Syn.tcl Example (7/7)

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

#================================================================================
#  Finish and Quit
#================================================================================
report_timing
report_area
check_design

exit
```

SRAM File (1/5)

In sram_256x8.pdf

Process Technology:
TSMC CL013G

High-Speed Single-Port Synchronous Flex-Repair^{TM} SRAM with Redundancy

sram_256x8
256X8, Mux 8, Drive 6

Features

- Precise Optimization for TSMC's Eight-Layer Metal 0.13µm CL013G CMOS Process
- High Density (area is 0.018mm^2)
- Fast Access Time (1.20ns at fast@0C process 1.32V, 0˚C)
- Fast Cycle Time (1.31ns at fast@0C process 1.32V, 0˚C)
- One Read/Write Port
- Completely Static Operation
- Near-Zero Hold Time (Data, Address, and Control Inputs)

Memory Description

The 256X8 SRAM is a high-performance, synchronous single-port, 256-word by 8-bit memory designed to take full advantage of TSMC's eight-layer metal, 0.13µm CL013G CMOS process.

The SRAM's storage array is composed of six-transistor cells with fully static memory circuitry. The SRAM operates at a voltage of 1.2V ± 10% and a junction temperature range of -40˚C to +125˚C.

SRAM File (2/5)

In sram_256x8.pdf

Pin Description

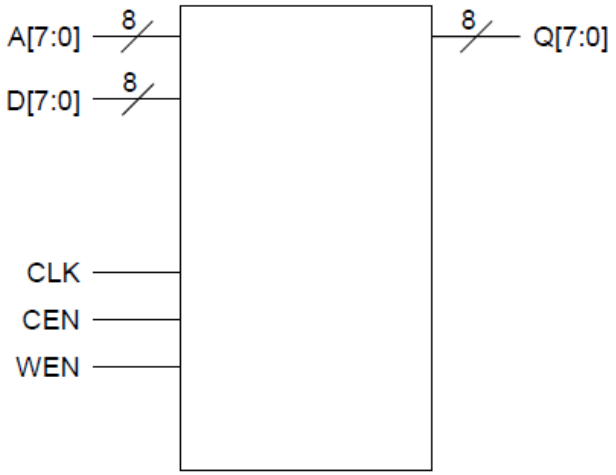
Pin	Description
A[7:0]	Addresses (A[0] = LSB)
D[7:0]	Data Inputs (D[0] = LSB)
CLK	Clock Input
CEN	Chip Enable
WEN	Write Enable
Q[7:0]	Data Outputs (Q[0] = LSB)

Area

Area Type	Width (mm)	Height (mm)	Area (mm^2)
Core	0.143	0.125	0.018
Footprint	0.153	0.136	0.021

The footprint area includes the core area and user-defined power ring and pin spacing areas.

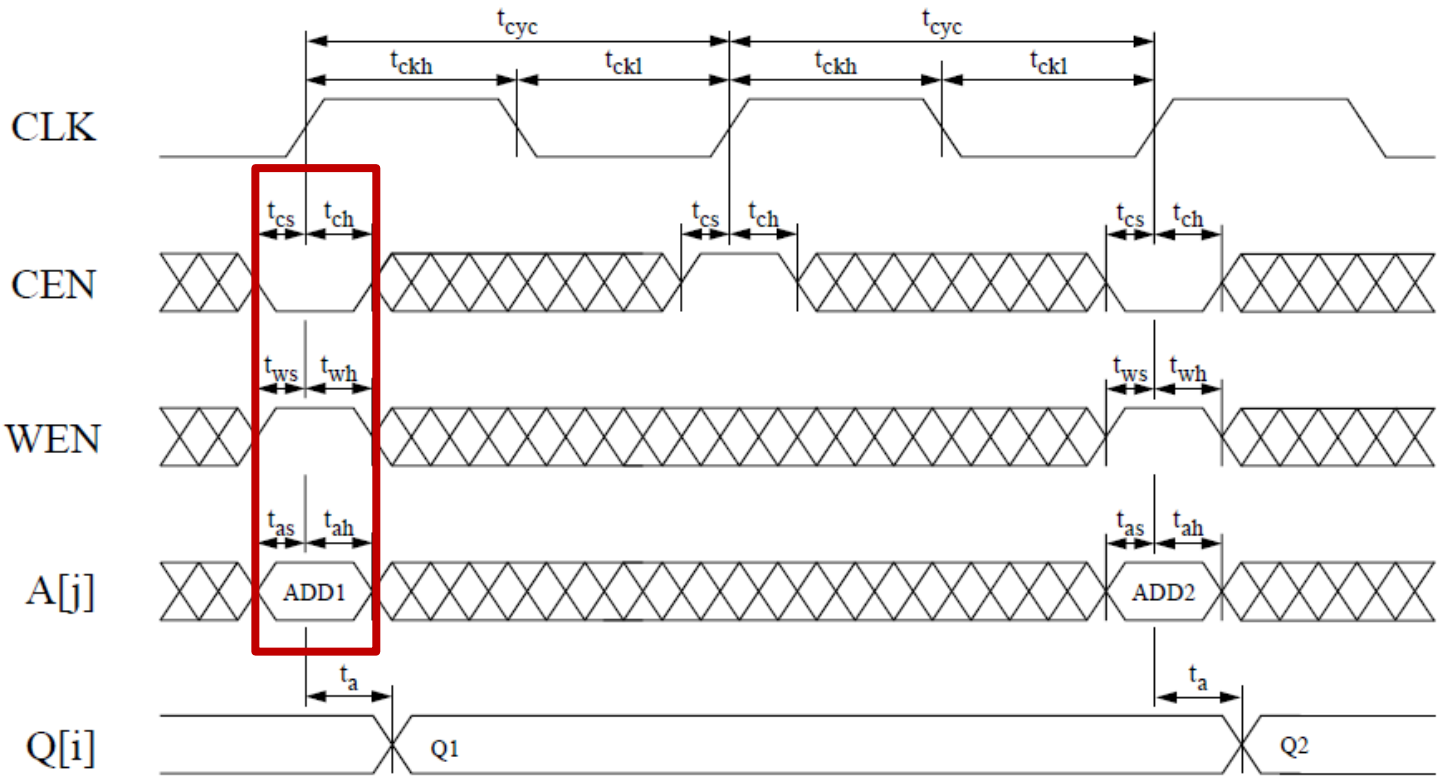
Symbol



CEN	WEN	Data Out	Mode
H	X	Last Data	Standby
L	L	Data In	Write
L	H	SRAM Data	Read

SRAM File (3/5)

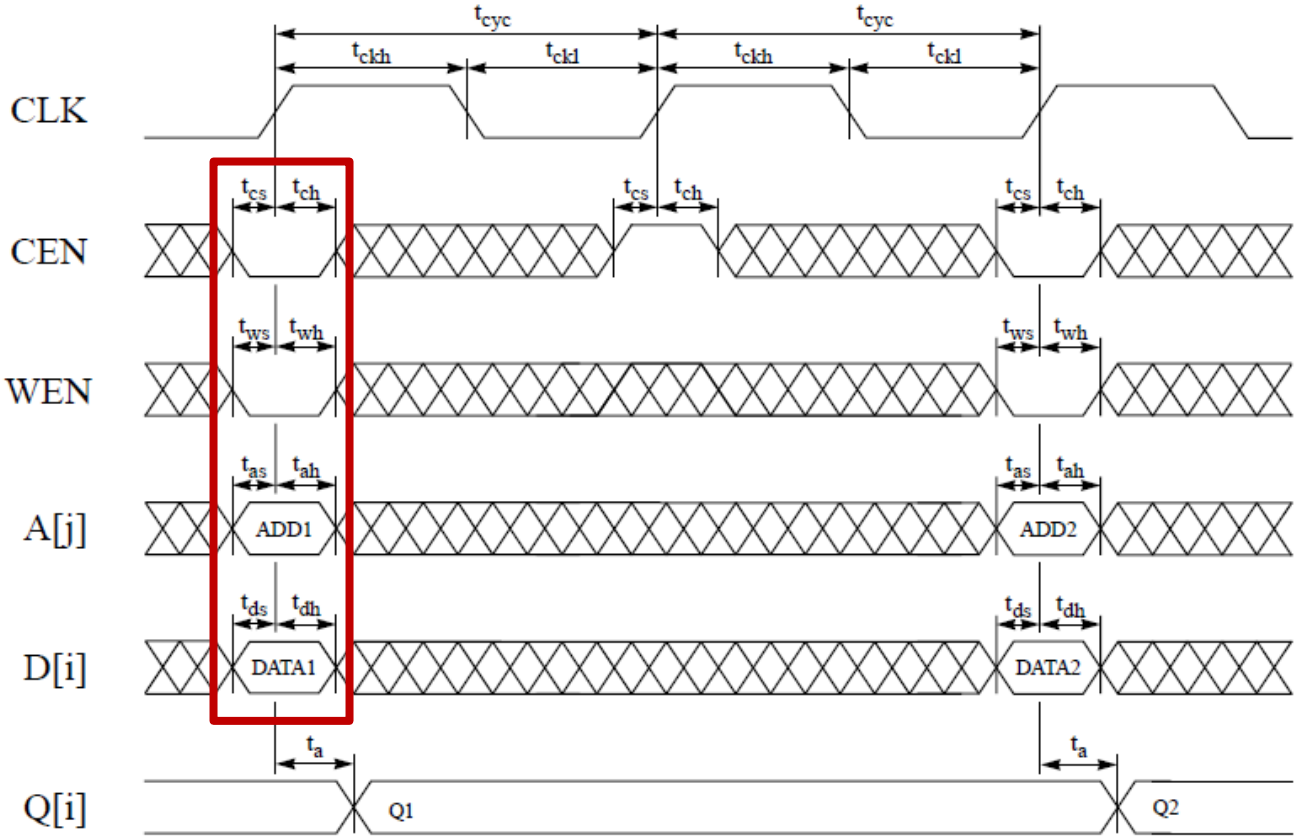
Synchronous Single-Port SRAM Read-Cycle Timing



Rising delays are measured at 50% of VDD and falling delays are measured at 50% of VDD.
Rising and falling slews are measured from 10% VDD to 90% VDD.

SRAM File (4/5)

Synchronous Single-Port SRAM Write-Cycle Timing



Rising delays are measured at 50% of VDD and falling delays are measured at 50% of VDD.
Rising and falling slews are measured from 10% VDD to 90% VDD.

SRAM File (5/5)

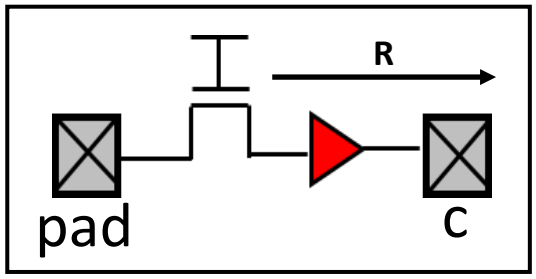
```
library(sram_256x8) {
    delay_model     : table_lookup;
    revision        : 1.1;
    date            : "2013-03-29 08:26:06Z";
    comment         : "Confidential Information of Artisan Components, Inc.  Use sub
ject to Artisan Components license. Copyright (c) 2013 Artisan Components, Inc.";
    time_unit               : "1ns";
    voltage_unit            : "1V";
    current_unit            : "1mA";
    leakage_power_unit      : "1mW";
    nom_process             : 1;
    nom_temperature         : 125.000;
    nom_voltage             : 1.080;
    capacitive_load_unit     (1,pf);
    pulling_resistance_unit  : "1kohm";

    /* additional header data */
    default_cell_leakage_power  : 0;
    default_fanout_load         : 1;
    default_inout_pin_cap       : 0.005;
    default_input_pin_cap       : 0.005;
    default_output_pin_cap      : 0.0;
    default_max_transition      : 1.500;
```

sram_256x8_slow_syn.lib

Pad Example: tpz973g.v

Input Pad

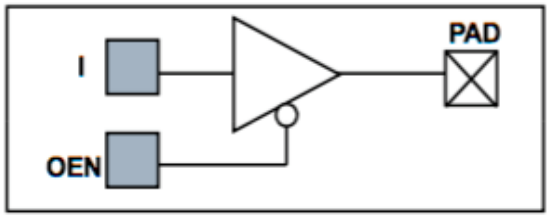
TSMC18: PDIDGZ




```
// type: PDIx
`celldefine
module PDIDGZ (PAD, C);
   input PAD;
   output C;
   buf   (C, PAD);
   specify
      (PAD => C)=(0, 0);
   endspecify
endmodule
`endcelldefine
```

(for TSMC180 process)

Output Pad



(PDO24CDG)

PDO02CDG / PDO04CDG / PDO08CDG
PDO12CDG / PDO16CDG / PDO24CDG

```
// type: PDOx
`celldefine
module PDO02CDG (I, PAD);
    input I;
    output PAD;
    buf   (PAD, I);
    specify
       (I => PAD)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// type: PDOx
`celldefine
module PDO024CDG (I, PAD);
    input I;
    output PAD;
    buf   (PAD, I);
    specify
       (I => PAD)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

Cell Model Example: tsmc13.v

```
`timescale 1ns/1ps
`celldefine
module BUFX2 (Y, A);
output Y;
input A;

  buf I0(Y, A);
  specify
    // delay parameters
    specparam
      tplh$A$Y = 1.0,
      tphl$A$Y = 1.0;

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

endmodule // BUFX2
`endcelldefine
```

```
`timescale 1ns/1ps
`celldefine
module INVX1 (Y, A);
output Y;
input A;

  not I0(Y, A);
  specify
    // delay parameters
    specparam
      tplh$A$Y = 1.0,
      tphl$A$Y = 1.0;

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

endmodule // INVX1
`endcelldefine
```

Cell Model Example: tsmc13.v

```
`timescale 1ns/1ps
`celldefine
module DFFRX1 (Q, QN, D, CK, RN);
output Q, QN;
input  D, CK, RN;
reg NOTIFIER;
supply1 xSN;
  buf     XX0 (xRN, RN);
  buf     IC (clk, CK);
  udp_dff I0 (n0, D, clk, xRN, xSN, NOTIFIER);
  and     I4 (flag, xRN, xSN);
  buf     I1 (Q, n0);
  not     I2 (QN, n0);
  specify
    specparam
    tplh$RN$Q     = 1.0,
    tphl$RN$Q     = 1.0,
    tplh$RN$QN    = 1.0,
    tphl$RN$QN    = 1.0,
    tplh$CK$Q     = 1.0,
    tphl$CK$Q     = 1.0,
    tplh$CK$QN    = 1.0,
    tphl$CK$QN    = 1.0,
    tsetup$D$CK   = 1.0,
    thold$D$CK    = 0.5,
    tsetup$RN$CK  = 1.0,
    thold$RN$CK   = 0.5,
    tminpwl$RN    = 1.0,
    tminpwl$CK    = 1.0,
    tminpwh$CK    = 1.0;
  if (flag)
       (posedge CK *> (Q    +: D)) = (tplh$CK$Q,    tphl$CK$Q);
  if (flag)
       (posedge CK *> (QN   -: D)) = (tplh$CK$QN,   tphl$CK$QN);
  $setuphold(posedge CK &&& (flag == 1), posedge D, tsetup$D$CK, thold$D$CK, NOTIFIER);
  $setuphold(posedge CK &&& (flag == 1), negedge D, tsetup$D$CK, thold$D$CK, NOTIFIER);
       (negedge RN *>  (Q    +: 1'b0)) = (tphl$RN$Q);
       (negedge RN *>  (QN -: 1'b0)) = (tplh$RN$QN);
  $setuphold(posedge CK, posedge RN, tsetup$RN$CK, thold$RN$CK, NOTIFIER);
  $width(negedge RN, tminpwl$RN, 0, NOTIFIER);
  $width(negedge CK &&& (flag == 1), tminpwl$CK, 0, NOTIFIER);
  $width(posedge CK &&& (flag == 1), tminpwh$CK, 0, NOTIFIER);
    endspecify
endmodule // DFFRX1
`endcelldefine
```

Cell Model Example: TSMC13g.pdf

Path: /home/raid7_2/course/cvsd/CBDK_IC_Contest/orig_lib/aci/sc-x/doc/tsmc13g.pdf

NAND2

Cell Description

The NAND2 cell provides the logical NAND of two inputs (A, B). The output (Y) is represented by the logic equation:

$Y = \overline{(A \bullet B)}$

Logic Symbol

Functions

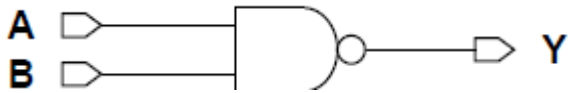
A	B	Y
0	x	1
x	0	1
1	1	0

Cell Size

Drive Strength	Height (µm)	Width (µm)
NAND2XL	3.69	1.38
NAND2X1	3.69	1.38
NAND2X2	3.69	2.30
NAND2X4	3.69	3.22
NAND2X6	3.69	4.60
NAND2X8	3.69	5.52

Gate counts = Chip Area / NAND2 Area

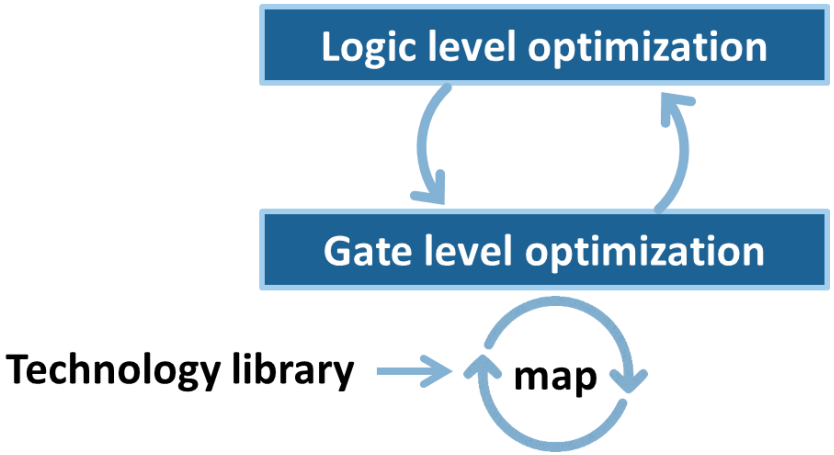
Functional Schematic



Appendix: Static Timing Analysis (STA)

- Gate level optimization
  - Generates gate-level netlist
  - Check timing & area goals

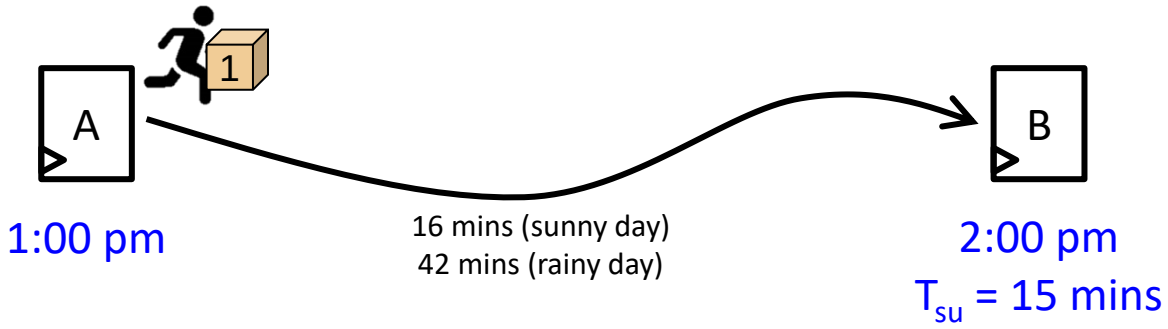
How?



	Point	Incr	Path
21			
22	---	---	---
23	clock i_clk (rise edge)	0.00	0.00
24	clock network delay (ideal)	0.50	0.50
25	input external delay	5.00	5.50 f
26	i_inst[0] (in)	0.00	5.50 f
27	U188/Y (CLKBUFX3)	0.11	5.61 f
28	U186/Y (NAND2X1)	0.08	5.69 r
29	U152/Y (CLKINVX1)	0.05	5.74 f
30	U150/Y (NAND2X1)	0.05	5.79 r
31	U148/Y (OA21XL)	0.11	5.91 r
32	U141/Y (OAI21XL)	0.05	5.96 f
33	U140/Y (NAND2X1)	0.11	6.07 r
34	U175/Y (OAI21XL)	0.06	6.13 f
35	U216/Y (AOI221XL)	0.20	6.32 r
36	U214/Y (NAND2X1)	0.04	6.36 f
37	o_data_reg[2]/D (DFFRX1)	0.00	6.36 f
38	data arrival time		6.36
39			
40	clock i_clk (rise edge)	10.00	10.00
41	clock network delay (ideal)	0.50	10.50
42	clock uncertainty	-0.10	10.40
43	o_data_reg[2]/CK (DFFRX1)	0.00	10.40 r
44	library setup time	-0.13	10.27
45	data required time		10.27
46	---	---	---
47	data required time		10.27
48	data arrival time		-6.36
49	---	---	---
50	slack (MET)		3.91

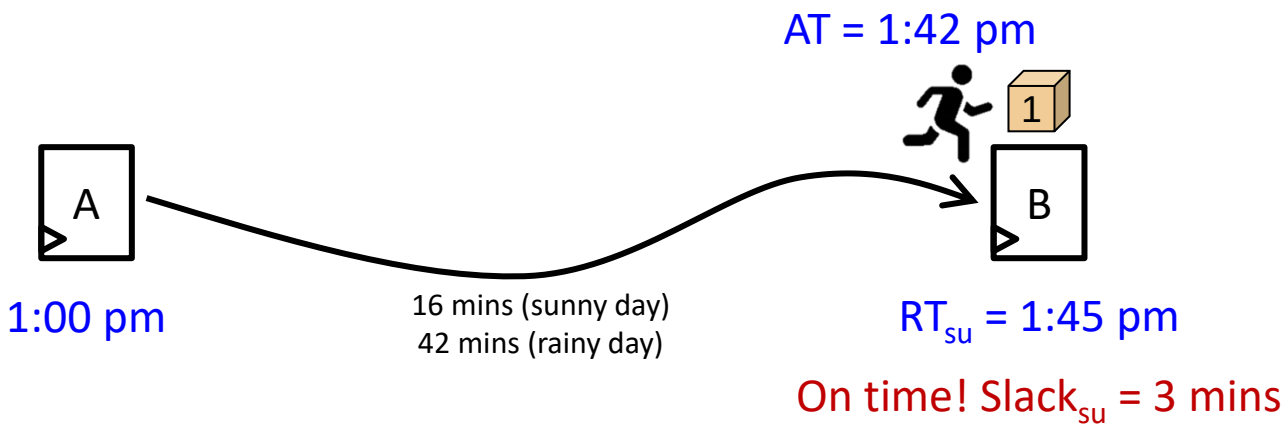
How to calculate the slack?

STA Example: Setup Requirement



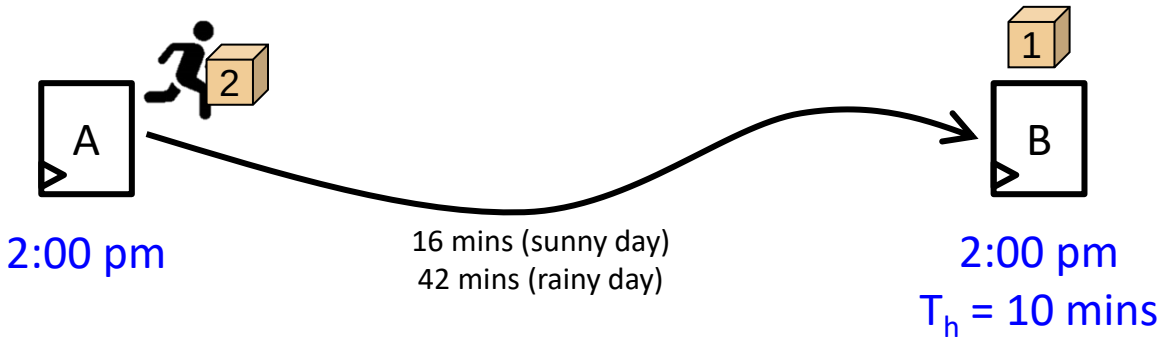
- Given conditions (assume clock is ideal)
  - Clock period D_p = 60 mins
  - Setup time T_s = 15 mins
  - Path delay (slow condition) = 42 mins
- Requirement 1: **data ready before 1:45 pm**

STA Example: Setup Requirement



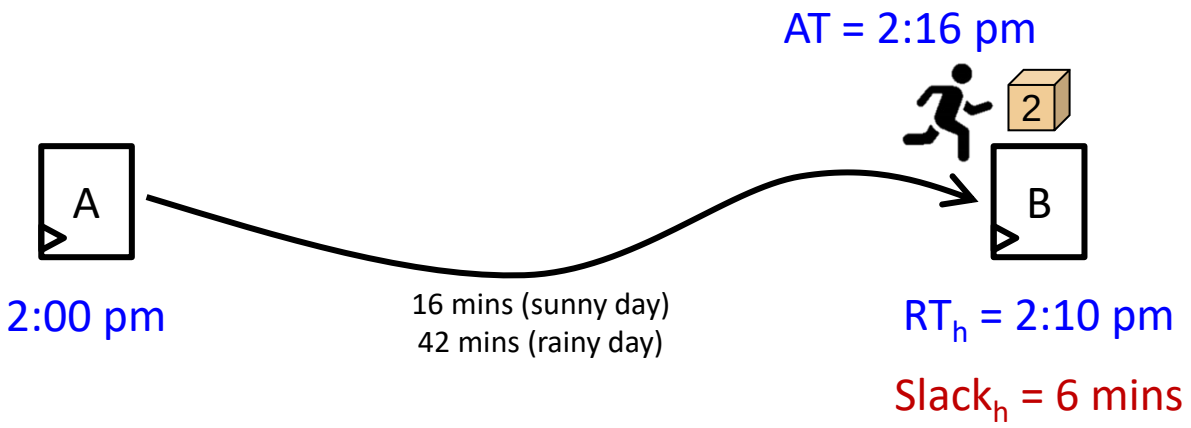
- Given conditions (assume clock is ideal)
  - Clock period D_p = 60 mins
  - Setup time T_s = 15 mins
  - Path delay (slow condition) = 42 mins
- Requirement 1: **data ready before 1:45 pm**
- We get
  - Require time$_{setup}$ RT_{su} = 1:45 pm
  - Arrival time AT = 1:42 pm
  - $Slack_{su} = RT_{su} - AT$ = 3 mins

STA Example: Hold Requirement



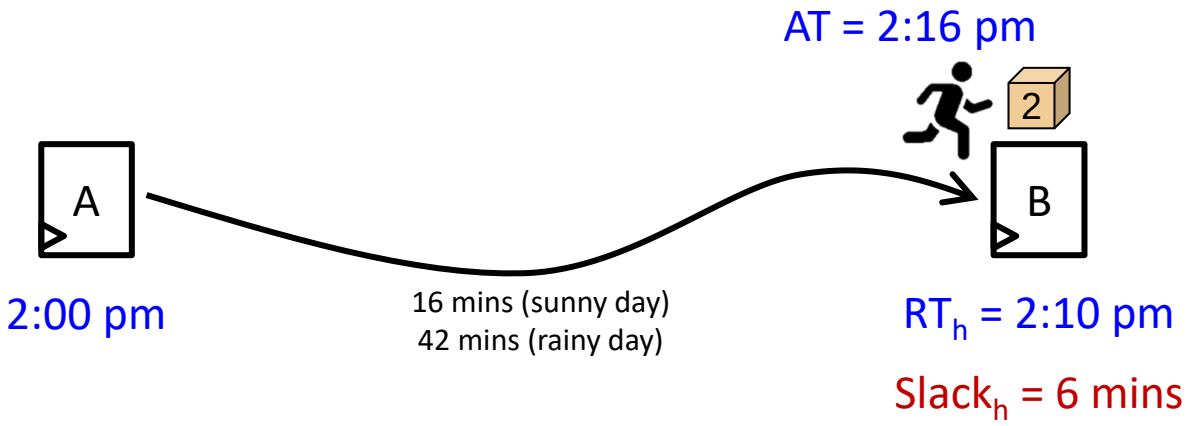
- Given conditions (assume clock is ideal)
  - Clock period D_p = 60 mins
  - Hold time T_h = 10 mins
  - Path delay (fast condition) = 16 mins
- Requirement 2: **data not overwritten until 2:10 pm**

STA Example: Hold Requirement



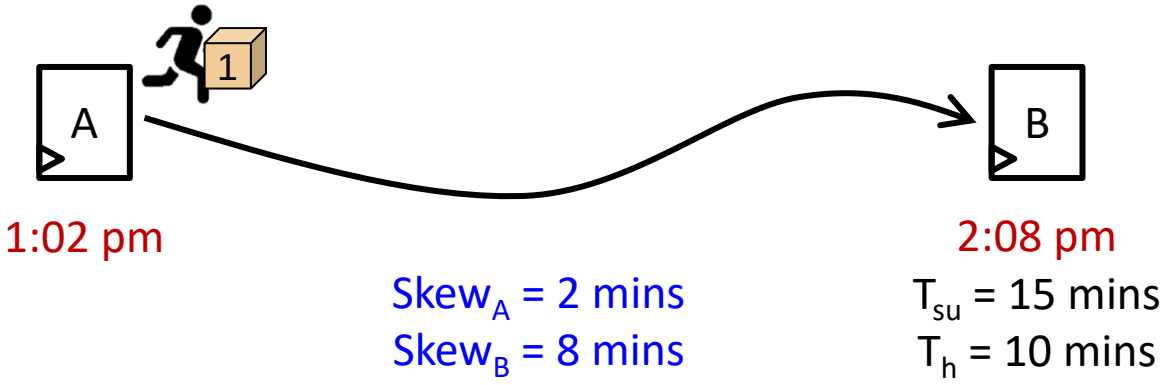
- Given conditions (assume clock is ideal)
  - Clock period D_p = 60 mins
  - Hold time T_h = 10 mins
  - Path delay (fast condition) = 16 mins
- Requirement 2: **data not overwritten until 2:10 pm**
- We get
  - Require time$_{hold}$ RT_h = 2:10 pm
  - Arrival time AT = 2:16 pm
  - $Slack_h = AT - RT_h$ = 6 mins

Simplified STA Formulas



- Ideal clock, reg-to-reg timing path
- Requirement 1: setup requirement
  - $RT_{su} = D_p - T_{su}$
  - Require $Slack_{su} = RT_{su} - AT \geq 0$
- Requirement 2: hold requirement
  - $RT_h = T_h$
  - Require $Slack_h = AT - RT_h \geq 0$

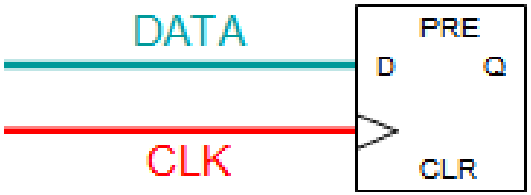
STA Example: Non-ideal Clock

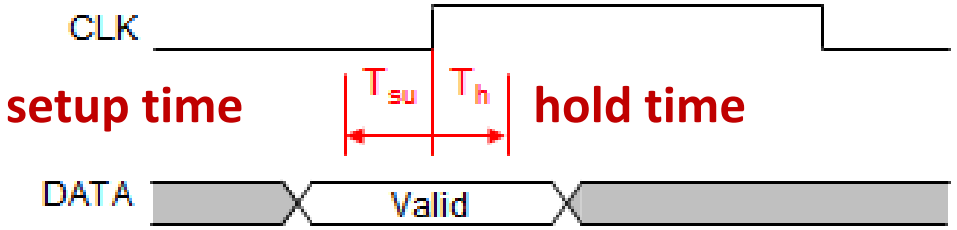


- Consider clock skew: Clock arrival time = clock edge + clock skew
- Case 1: Skew$_A$ < Skew$_B$
  - E.g., clk arrives at B 6 mins after A
  - Good for setup requirement, bad for hold requirement
- Case 2: Skew$_A$ > Skew$_B$
  - E.g., clk arrives at B 6 mins before A
  - Bad for setup requirement, good for hold requirement

- Consider clock uncertainty: bad for both setup & hold requirements

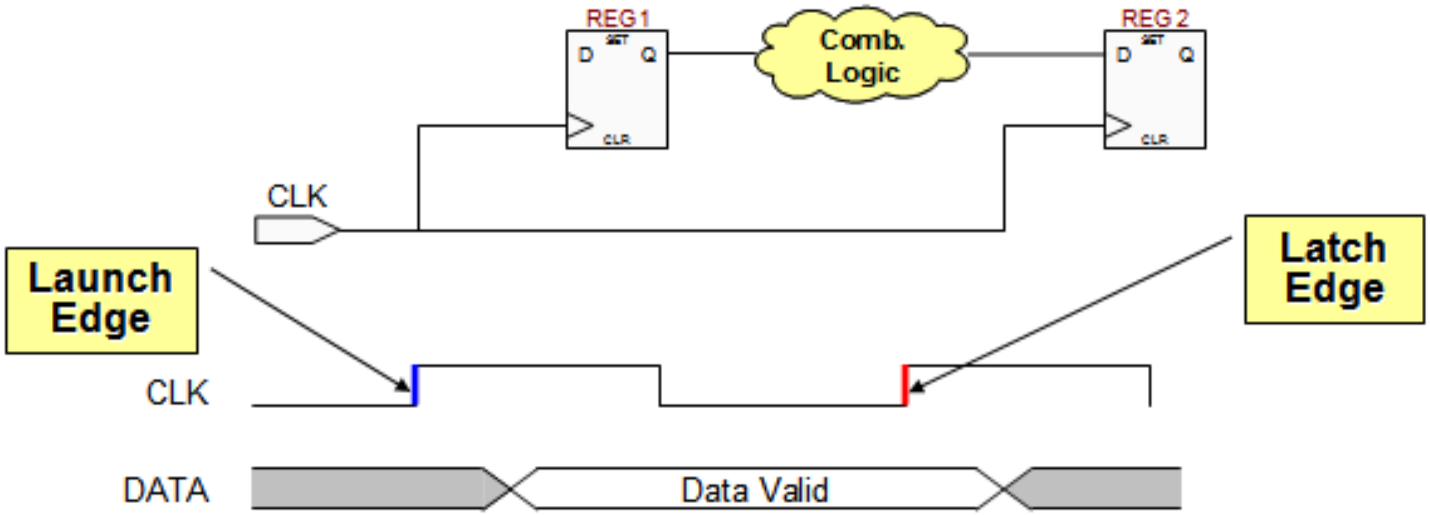
Setup Time & Hold Time





- **Setup Time**
  - The minimum amount of time before the clock's active edge that the data must be stable
  - Violation may cause incorrect data to be captured
- **Hold Time**
  - The minimum amount of time after the clock's active edge during which data must be stable
  - Violation may cause incorrect data to be latched

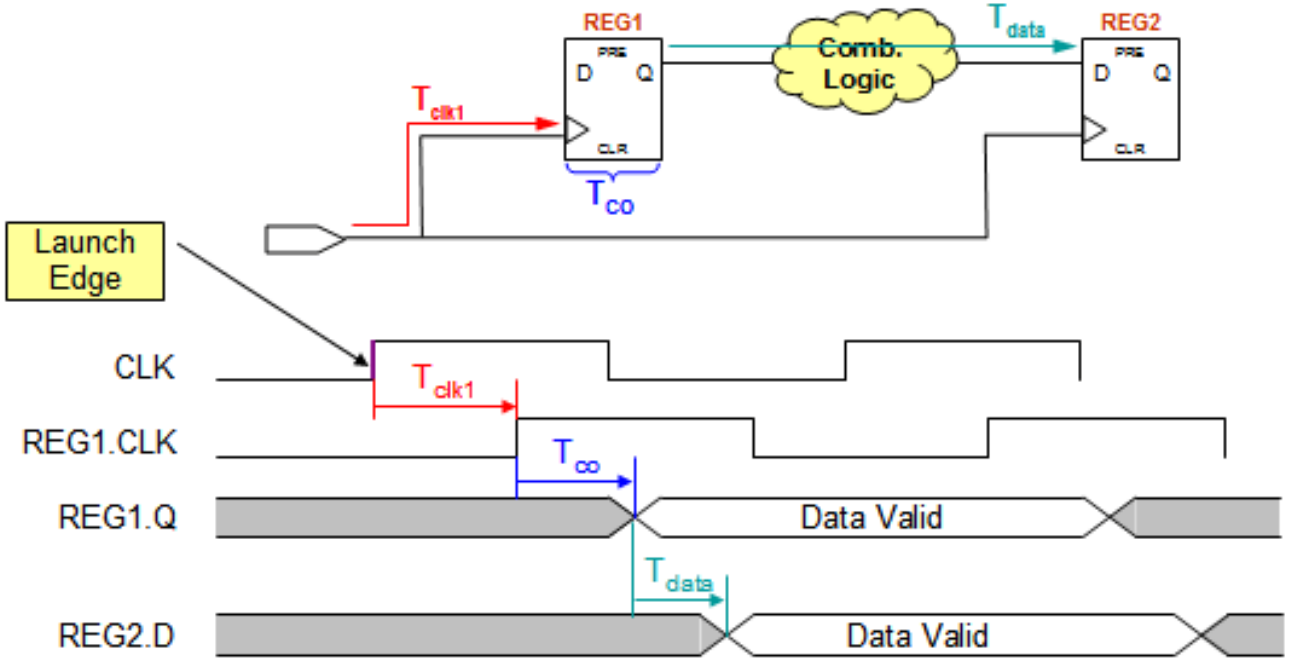
Launch Edge & Latch Edge (Capture Edge)



Latch edge = (launch edge) + D_p

- **Launch Edge**
  - CLK rising edge of register 1 for generating data
- **Latch Edge**
  - CLK rising edge of register 2 for receiving data

Data Arrival Time (Setup Time)



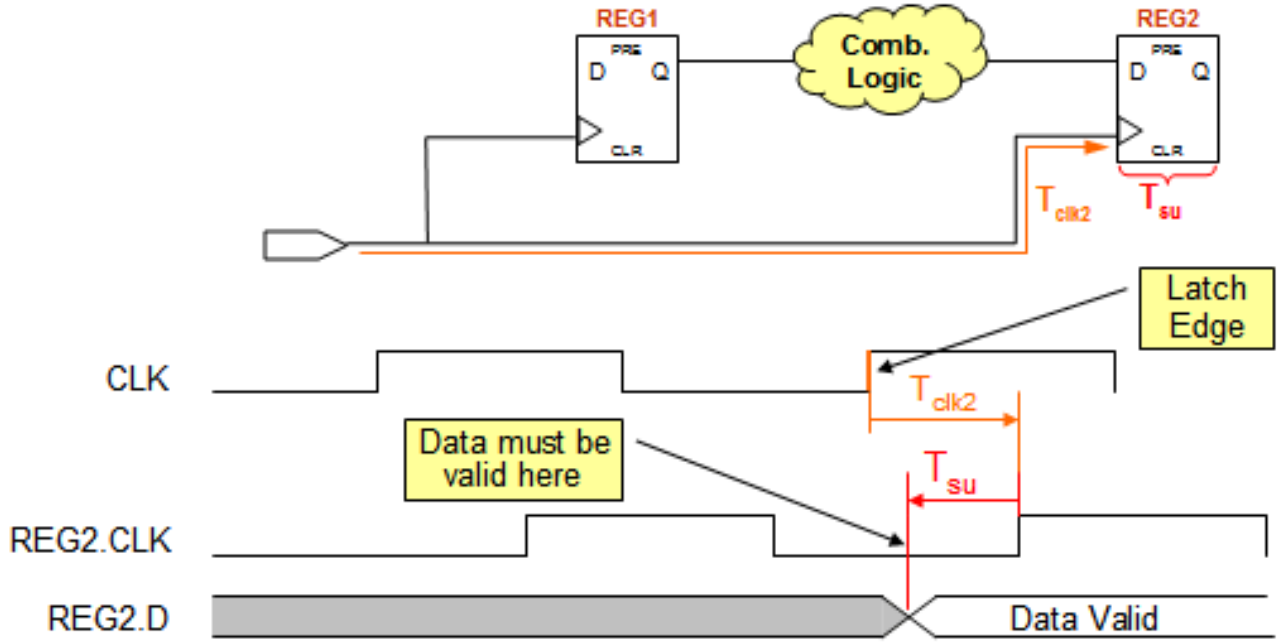
$$\text{Clock Arrival Time}_{REG1} = T_{clk1}$$

$$AT = T_{clk1} + \boxed{T_{co} + T_{data}}$$

Path Delay

- T_{clk1}: clock skew of register 1
- T_{co}: clock to output delay of register 1
- T_{data}: combinational logic delay

Data Required Time (Setup Time)

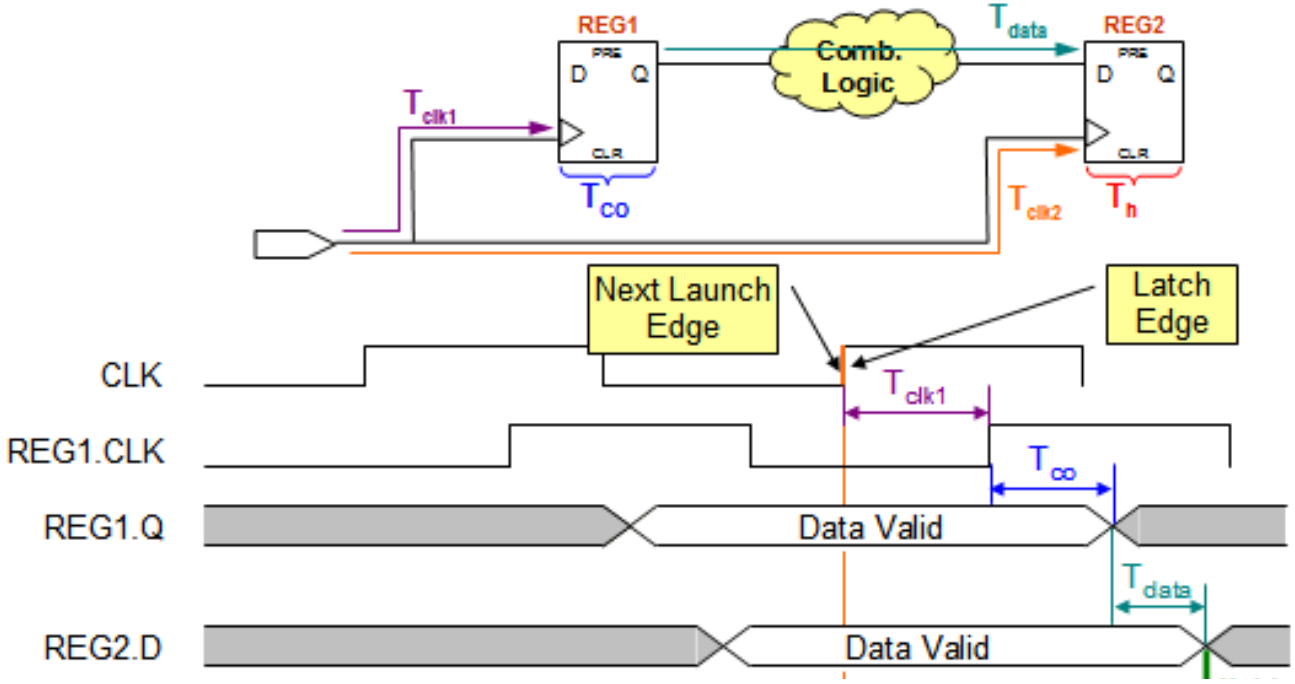


$$\text{Clock Arrival Time}_{REG2} = D_p + T_{clk2}$$

$$RT_{su} = D_p + T_{clk2} - T_{su} - \text{Uncertainty}$$

- T_{clk2}: clock skew of register 2
- T_{su}: setup time of data

Data Arrival Time (Hold Time)



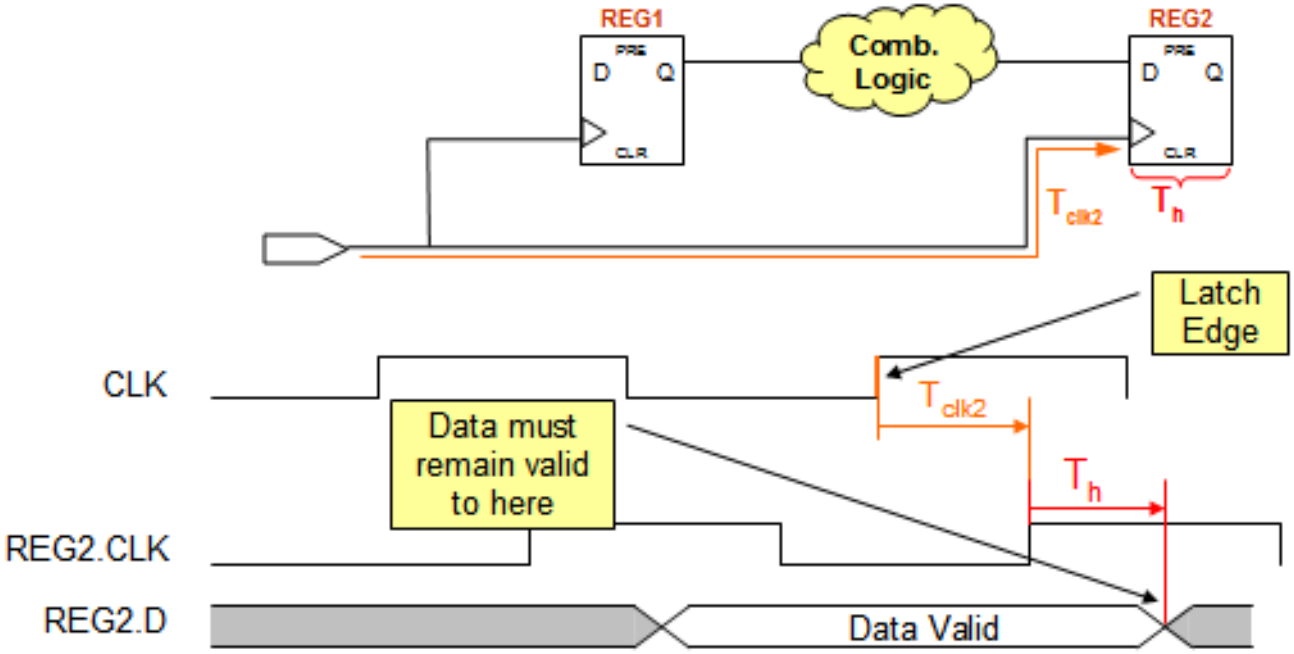
$$\text{Clock Arrival Time}_{REG1} = D_p + T_{clk1}$$

$$AT = D_p + T_{clk1} + \boxed{T_{co} + T_{data}}$$

Path Delay

- T_{clk1}: clock skew of register 1
- T_{co}: clock to output delay of register 1
- T_{data}: combinational logic delay

Data Required Time (hold time)

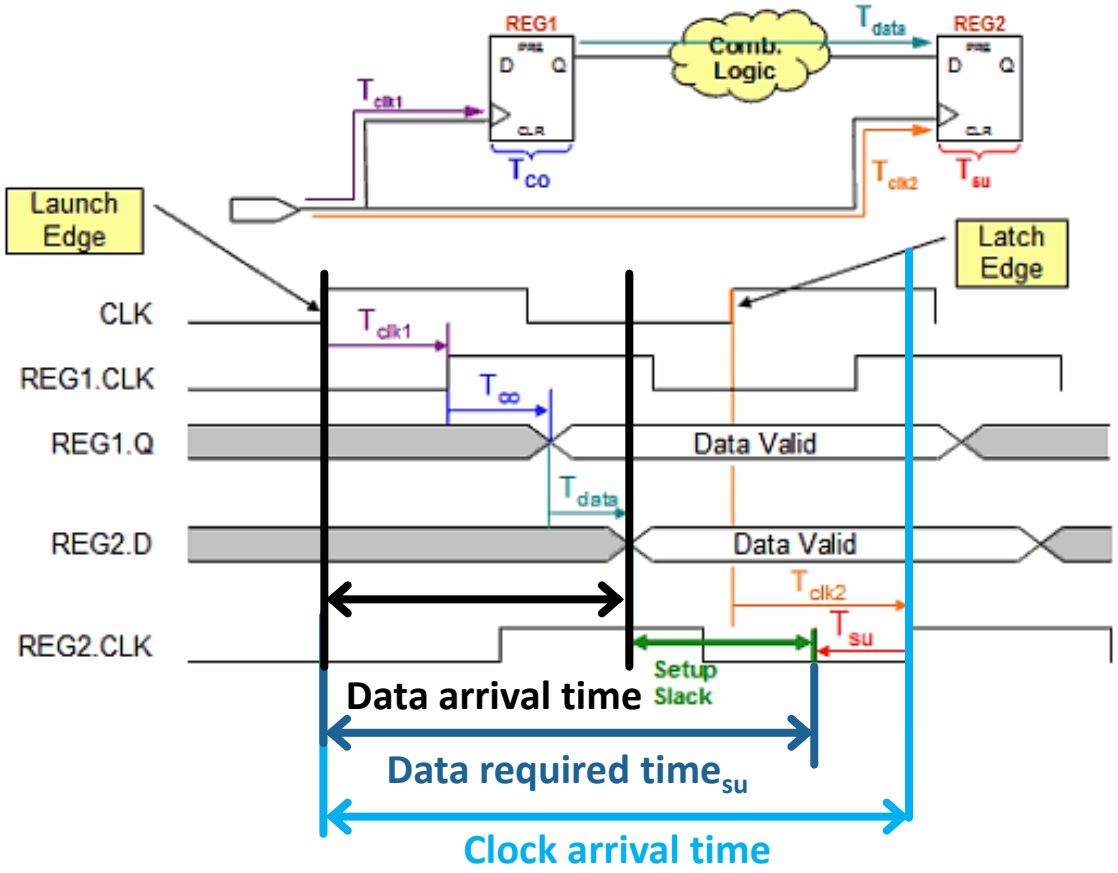


$$\text{Clock Arrival Time}_{REG2} = D_p + T_{clk2}$$

$$RT_h = D_p + T_{clk2} + T_h + \text{Uncertainty}$$

- T_{clk2}: clock skew of register 2
- T_h: hold time of data

Setup Slack

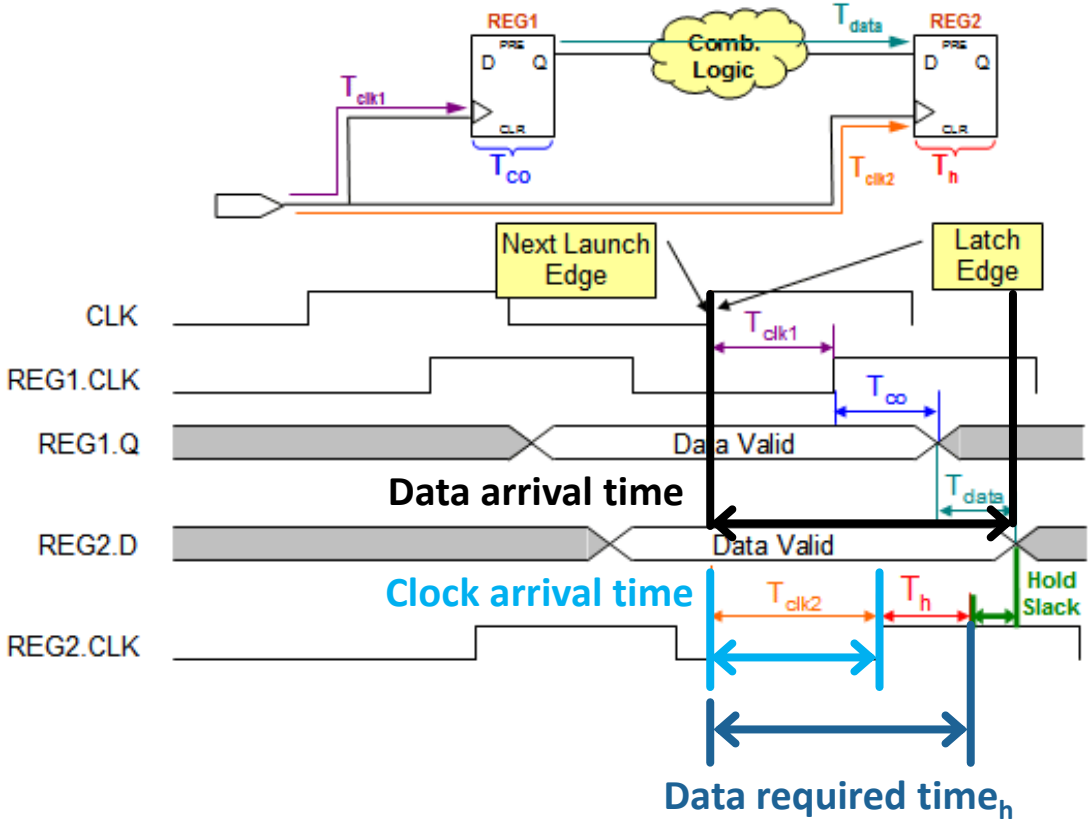


$$RT_{su} = D_p + T_{clk2} - T_{su} - \text{Uncertainty}$$

$$AT = T_{clk1} + T_{co} + T_{data}$$

$$\text{Setup Slack} = RT_{su} - AT$$

Hold Slack

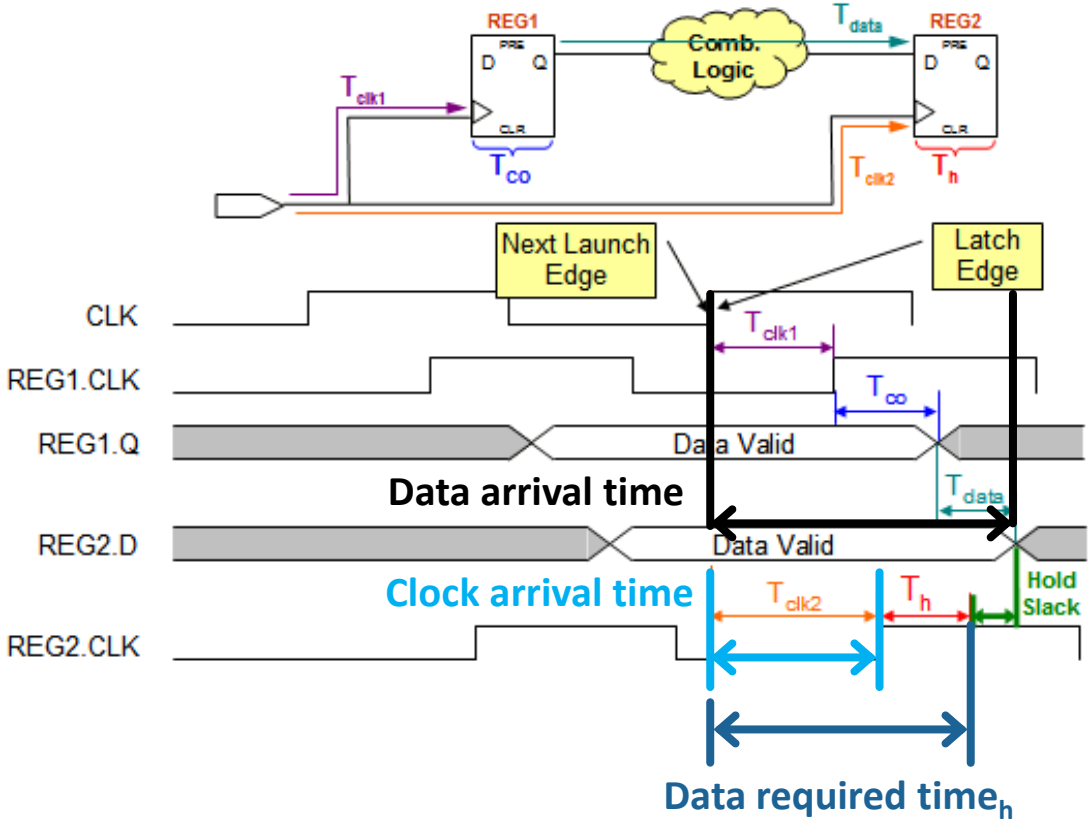


$$RT_h = (D_p) + T_{clk2} + T_h + \text{Uncertainty}$$

$$AT = (D_p) + T_{clk1} + T_{co} + T_{data}$$

$$\text{Hold Slack} = AT - RT_h$$

Hold Slack



$$RT_h = \cancel{(D_p)} + T_{clk2} + T_h + \text{Uncertainty}$$

$$AT = \cancel{(D_p)} + T_{clk1} + T_{co} + T_{data}$$

$$\text{Hold Slack} = AT - RT_h$$

Same edge, clock period doesn't affect hold slack