Training Course of Design Compiler

- T. –W. Tseng, "ARES Lab 2008 Summer Training Course of Design Compiler"

REF:

- CIC Training Manual – Logic Synthesis with Design Compiler, July, 2006
- TSMC 0.18um Process 1.8-Volt SAGE-X™ Stand Cell Library Databook, September, 2003
- TPZ973G TSMC 0.18um Standard I/O Library Databook, Version 240a, December 10, 2003
- Artisan User Manual

Speaker: T. –J. Chen

Advanced Reliable
Systems (ARES) Lab.

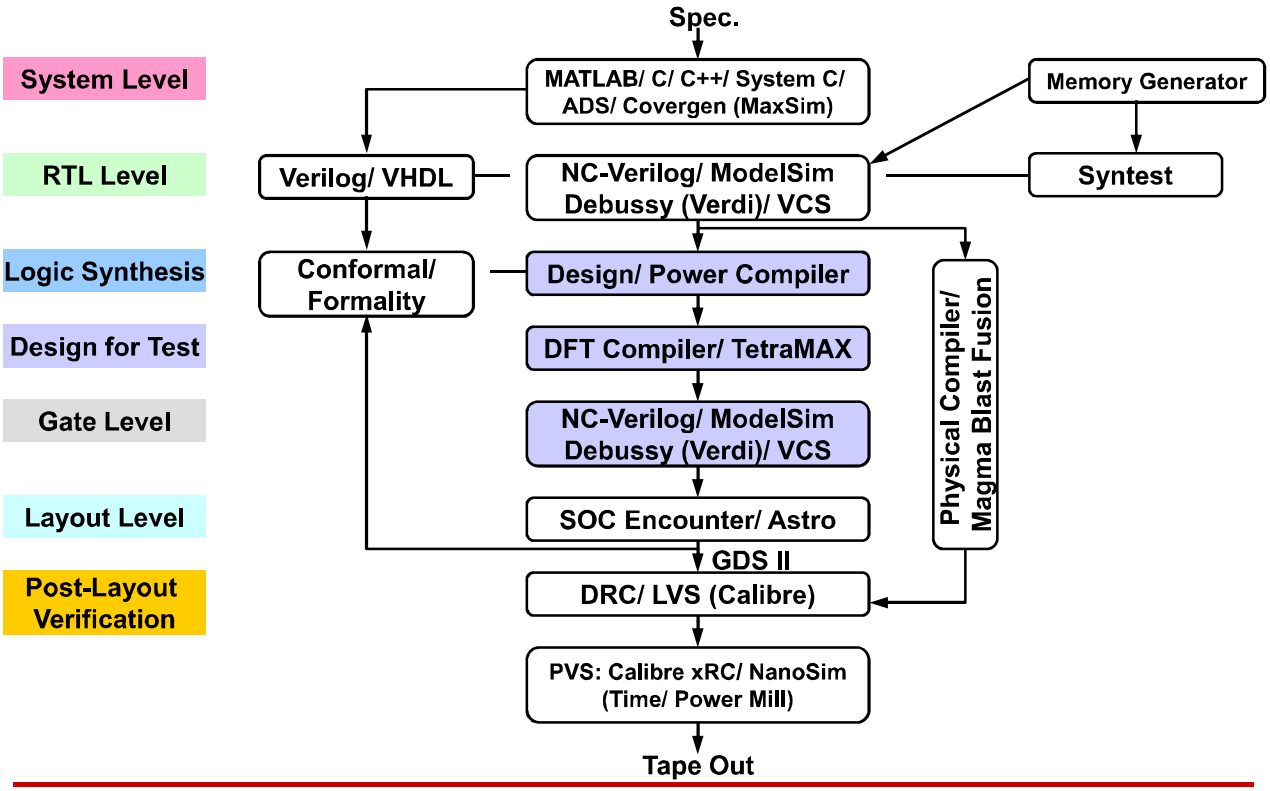
ADVANCED RELIABLE
SYSTEMS (ARES) LAB.

Outline

- Basic Concept of the Synthesis
- Synthesis Using Design Compiler

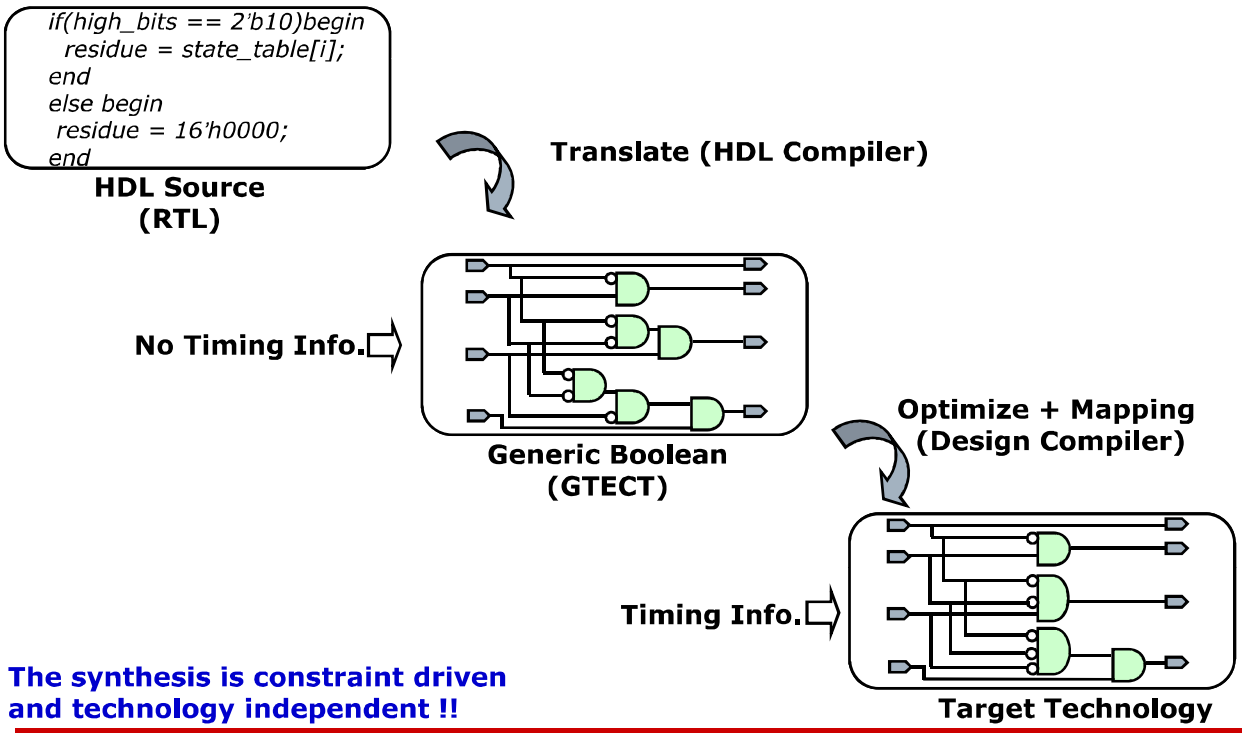
Basic Concept of the Synthesis

Cell-Based Design Flow



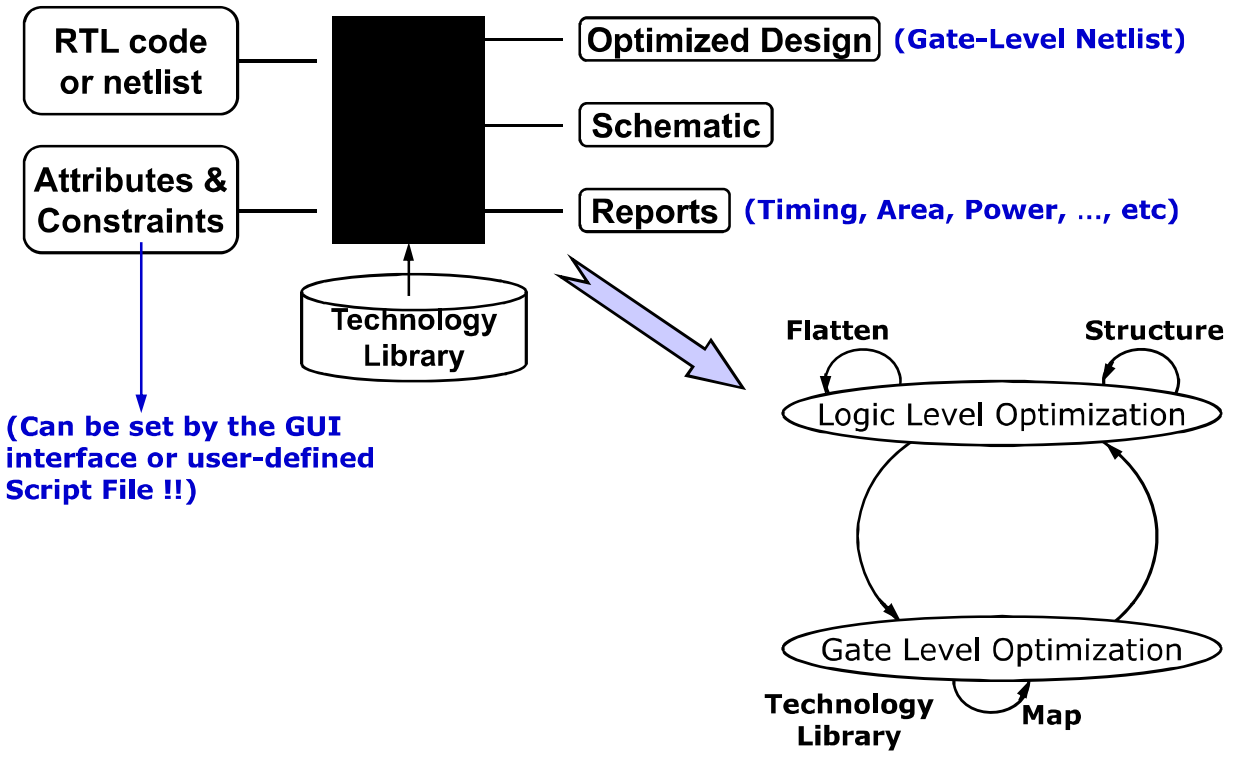
What is Synthesis

- Synthesis = translation + optimization + mapping



The synthesis is constraint driven and technology independent !!

Compile


RTL code or netlist
Attributes & Constraints
Optimized Design
(Gate-Level Netlist)
Schematic
Reports
(Timing, Area, Power, …, etc)
Technology Library
(Can be set by the GUI interface or user-defined Script File !!)
Flatten
Structure
Logic Level Optimization
Gate Level Optimization
Technology Library
Map

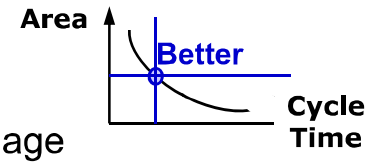
Synthesizable Verilog

- Verilog Basis
  - parameter declarations
  - wire, wand, wor declarations
  - reg declarations
  - input, output, inout declarations
  - continuous assignments
  - module instructions
  - gate instructions
  - always blocks
  - task statements
  - function definitions
  - for, while loop
- Synthesizable Verilog primitives cells
  - and, or, not, nand, nor, xor, xnor
  - bufif0, bufif1, notif0, notif1

Synthesizable Verilog (Cont')

- Operators
  - Binary bit-wise (~, &, |, ^, ~^)
  - Unary reduction (&, ~&, |, ~|, ^, ~^)
  - Logical (!, &&, ||)
  - 2's complement arithmetic (+, -, *, /, %)
  - Relational (>, <, >=, <=)
  - Equality (==, !=)
  - Logic shift (>>, <<)
  - Conditional (?:)
  - Concatenation ({ })

Notice Before Synthesis



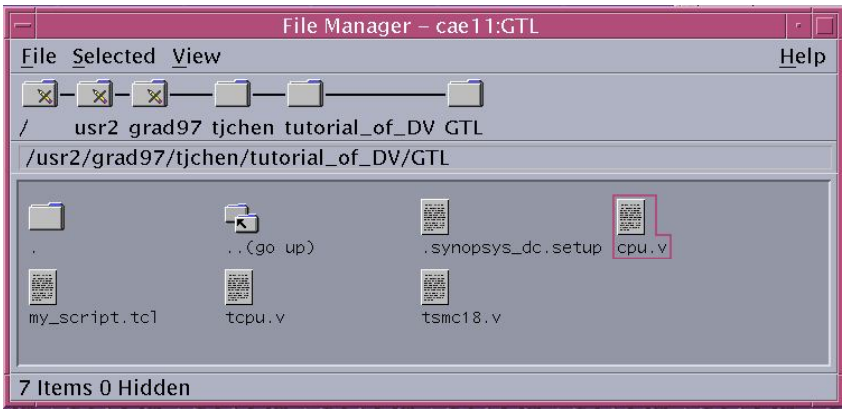
- Your RTL design
  - Functional verification by some high-level language
    - Also, the code coverage of your test benches should be verified (i.e. VN)
  - Coding style checking (i.e. n-Lint)
    - Good coding style will reduce most hazards while synthesis
    - Better optimization process results in better circuit performance
    - Easy debugging after synthesis
- Constraints
  - The area and timing of your circuit are mainly determined by your circuit architecture and coding style
  - There is always a trade-off between the circuit timing and area
  - In fact, a super tight timing constraint may be worked while synthesis, but failed in the Place & Route (P&R) procedure

Synthesis Using Design Compiler

Related Files

Folder	Name	Description
GTL	.synopsys_dc.setup	Design compiler setup file
	my_script.tcl	Synthesis script file
	my_design.v	Verilog files
	tmy_design.v	Test bench
	tsmc18.v	Verilog model of standard cells

Ex:



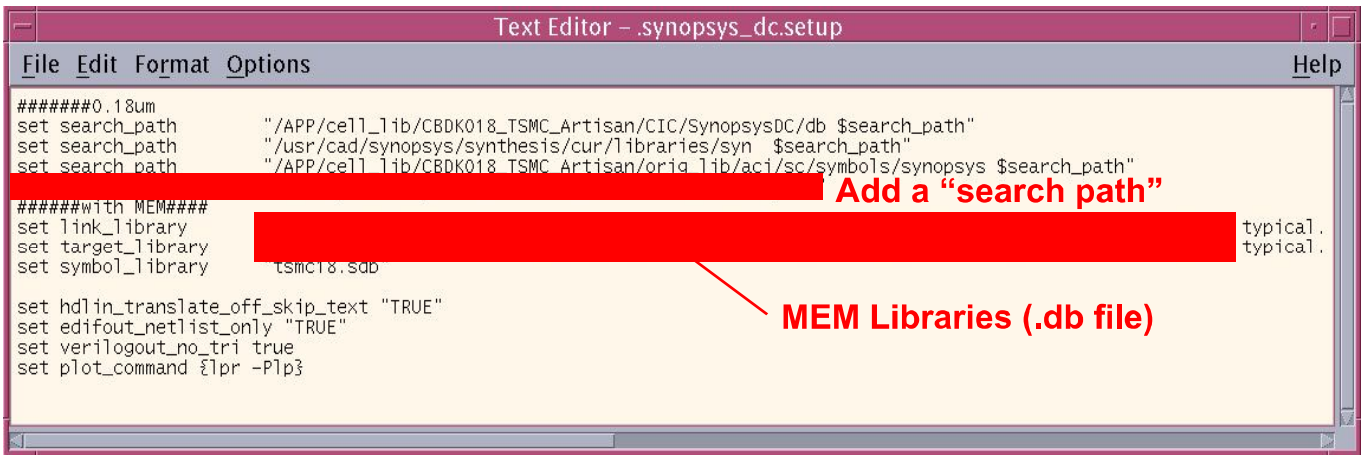
<.synopsys_dc.setup> File

- **link_library** : the library used for interpreting input description
  - Any cells instantiated in your HDL code
  - Wire load or operating condition modules used during synthesis
- **target_library** : the ASIC technology which the design is mapped
- **symbol_library** : used for schematic generation
- **search_path** : the path for unsolved reference library
- **synthetic_path** : designware library

<.synopsys_dc.setup> File (Cont')

- MEMs libraries are also included in this file

Ex:



(.synopsys_dc.setup File)

Note that the MEM DB files are converted from the LIB files which are generated from the Artisan !!

Settings for Using Memory

- Convert *.lib to *.db
  - *%> dc_shell –t*
  - *dc_shell-t> read_lib* [redacted]
  - *dc_shell-t> write_lib* [redacted] *-output *
    [redacted]

any memory LIB file

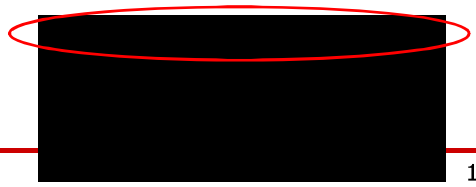
user library name, which should be the same as the library name in the Artisan

- Modify <.synopsys_dc.setup> File:
  - set link_library "* slow.db t13spsram512x32_slow.db dw_foundation.sldb"
  - set target_library "slow.db t13spsram512x32_slow.db"
  - add a "search path" to this file

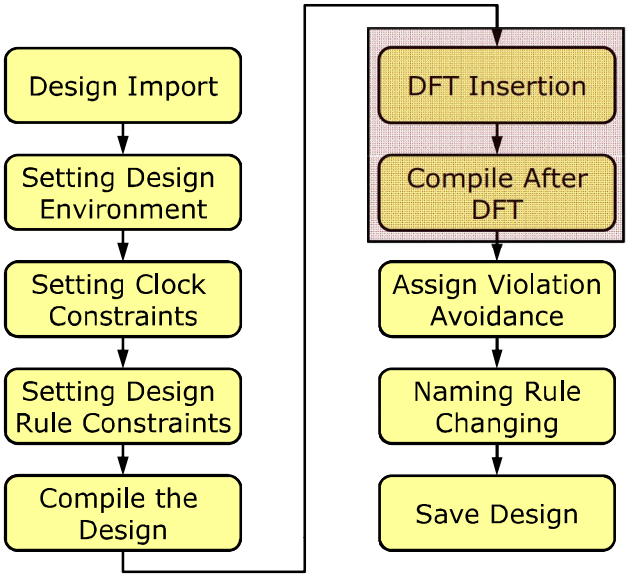
memory DB file

add to the file

- Before the synthesis, the memory HDL model should be blocked in your netlist



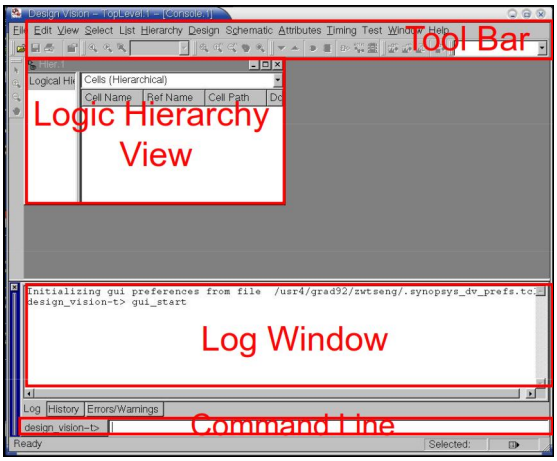
Synthesis Flow



Getting Started

- Prepare Files:
  - *.v files
  - *.db files (i.e. memory is used)
  - Synthesis script file (i.e. described later)

linux %> dv& (XG Mode)

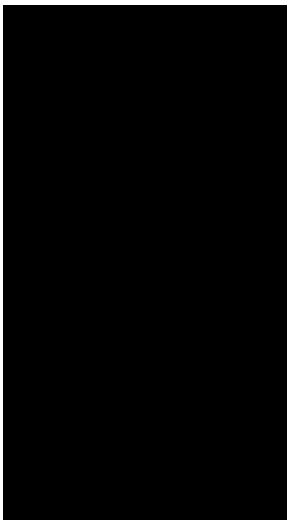


(GUI view of the Design Vision)

Read File

Design Import

- Read netlists or other design descriptions into Design Compiler
- ***File/Read***
- Supported formats
  - Verilog: .v
  - VHDL: .vhd
  - System Verilog: .sv
  - EDIF
  - PLA (Berkeley Espresso): .pla
  - Synopsys internal formats:
    - DB (binary): .db
    - Enhance db file: .ddc
    - Equation: .eqn
    - State table: .st



{ Command Line }

read_file –format verilog **file name**

PAD Parameters Extraction

- Input PAD
  - Input delay
  - Input driving
- Output PAD
  - Output delay
  - Output loading

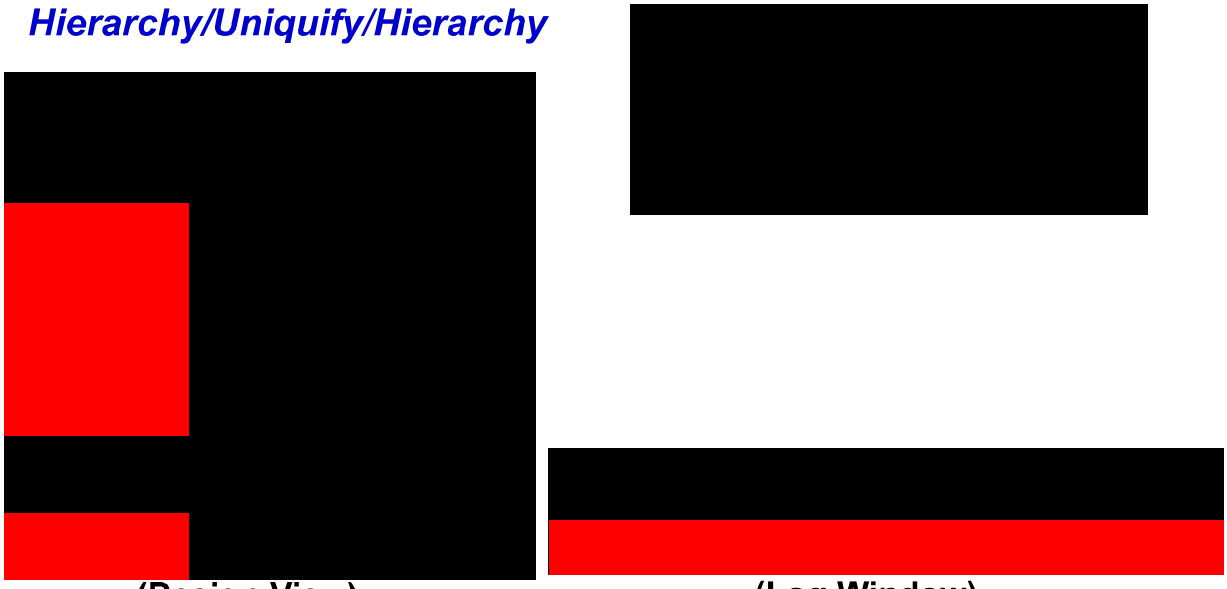
(chip_const.tcl)

{ Command Line }

```
current_design CHIP
characterize [get_cells CORE]
current_design CORE
write_script –format dctcl –o chip_const.tcl
```

Uniquify

- Select the most top design of the hierarchy
- ***Hierarchy/Uniquify/Hierarchy***



(Design View) (Log Window)

{ Command Line }

```
uniquify
```

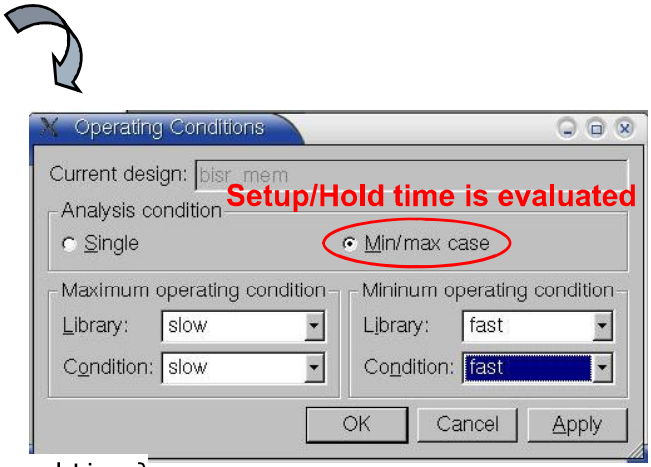
Design Environment

Setting Design Environment

- Setting Operating Environment
- Setting Input Driving Strength
- Setting Output Loading
- Setting Input/Output Delay
- Setting Wire Load Model

Setting Operating Condition

- ***Attributes/Operating Environment/Operating Conditions***



{ Command Line }

```
set_operating_conditions –max “slow” –max_library “slow” –min “fast”\
-min_library “fast”
```

Setting Drive Strength/Input Delay for PADs

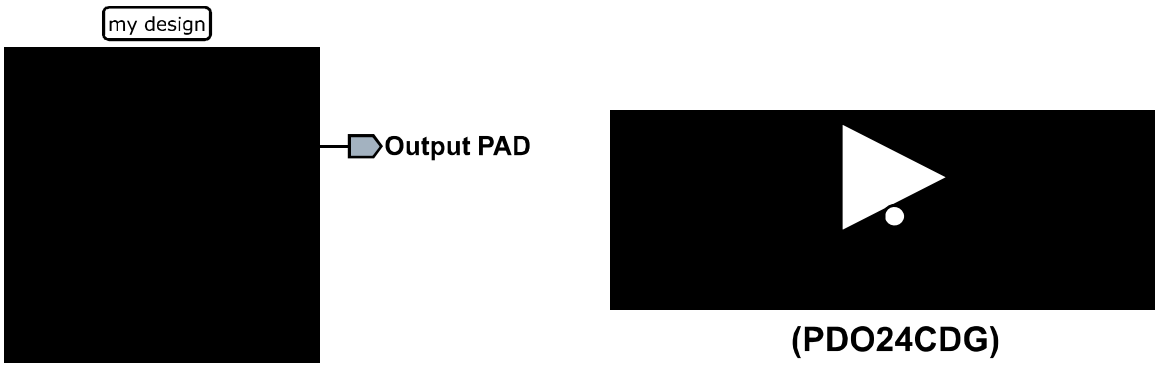
- Assume that we use the input PAD “PDIDGZ”

{ Command Line }

```
set_drive [expr 0.288001] [all_inputs]
set_input_delay [expr 0.34] –clock clk [all_inputs]
```

Setting Load/Output Delay for PADs

- Assume that we use the output PAD “PDO24CDG”

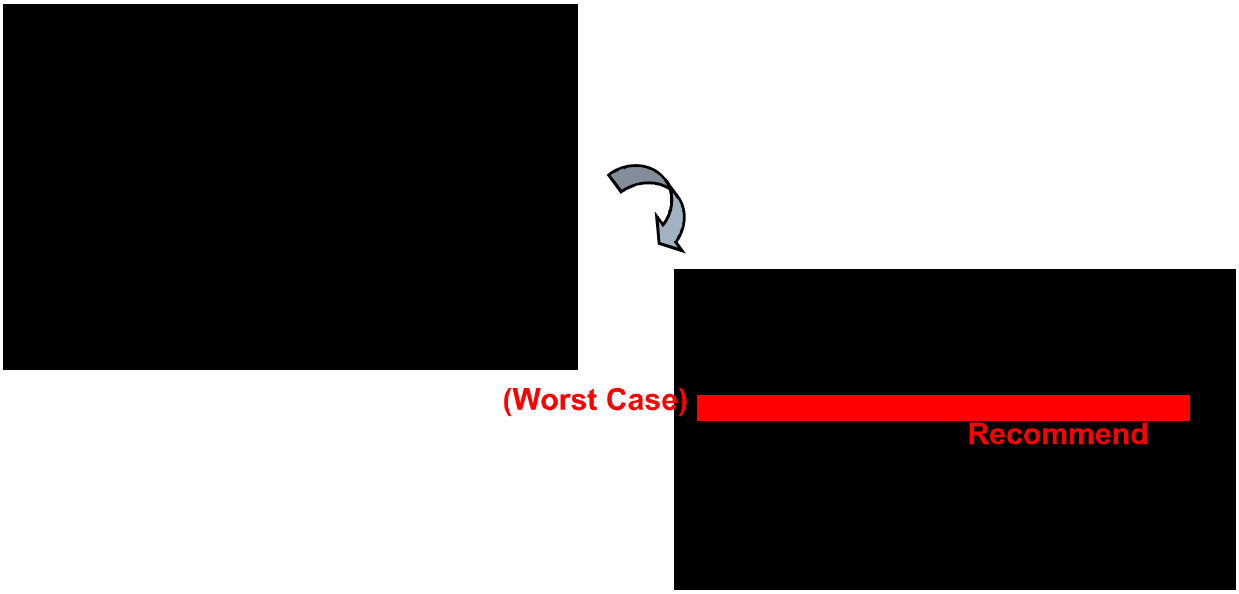


{ Command Line }

```
set_load [expr 0.06132] [all_outputs]
set_output_delay [expr 2] [all_outputs]
```

Setting Wire Load Model

- ***Attributes/Operating Environment/Wire Load***



{ Command Line }

```
set_wire_load_model –name “tsmc18_wl10” –library “slow”
set_wire_lode_mode “top”
```

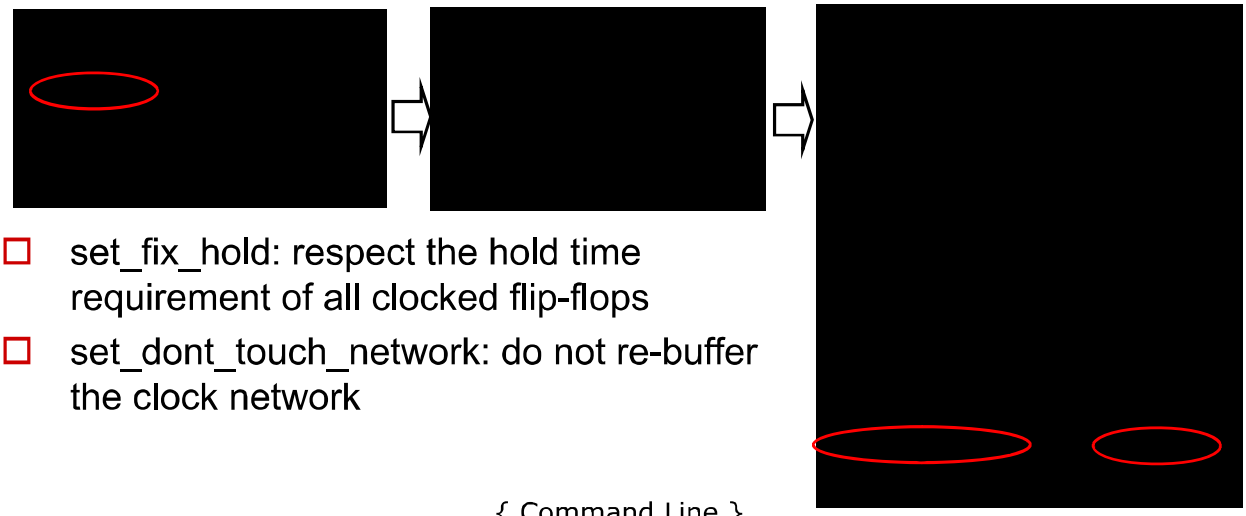
Clock Constraints

Setting Clock Constraints

- Period
- Waveform
- Uncertainty
  - Skew
- Latency
  - Source latency
  - Network latency
- Transition
  - Input transition
  - Clock transition
- Combination Circuit – Maximum Delay Constraints

Sequential Circuit → Specify Clock

- ***Select the "clk" pin on the symbol***
- ***Attributes/Specify Clock***



- set_fix_hold: respect the hold time requirement of all clocked flip-flops
- set_dont_touch_network: do not re-buffer the clock network

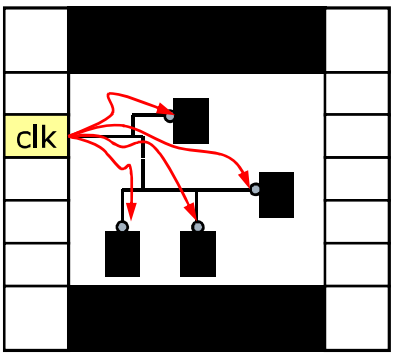
{ Command Line }

```
creat_clock –period 10 [get_ports clk]
set_dont_touch_network [get_clocks clk]
set_fix_hold [get_clocks clk]
```

Setting Clock Skew

- Different clock arrival time

Ex:



(Timing Report)

- experience
  - Small circuit: 0.1 ns
  - Large circuit: 0.3 ns

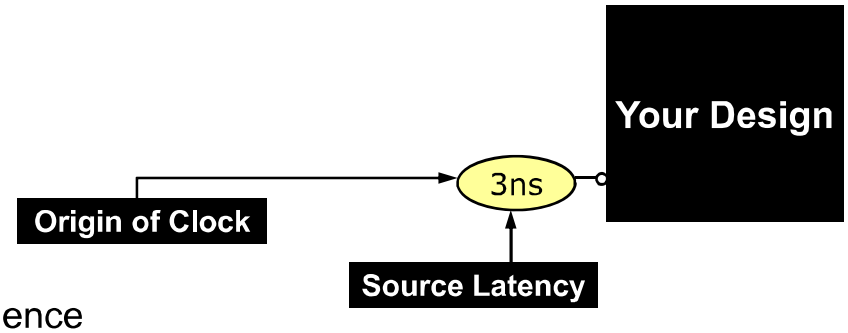
{ Command Line }

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

Setting Clock Latency

- *Source latency* is the propagation time from the actual clock origin to the clock definition point in the design
- This setting can be avoid if the design is without the clock generator

Ex:



- experience
  - Small circuit: 1 ns
  - Large circuit: 3 ns

{ Command Line }

```
set_clock_latency 1 [get_clocks clk]
```

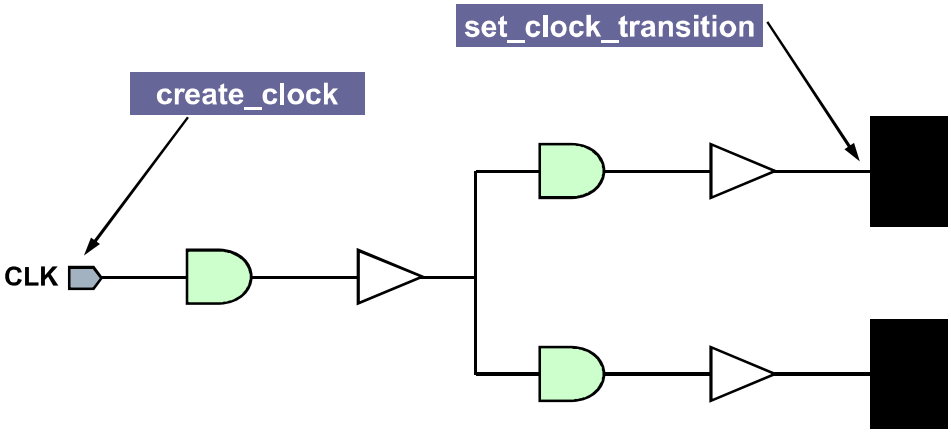
Setting Ideal Clock

- Since we usually let the clock tree synthesis (CTS) procedure performed in the P&R (i.e. set_dont_touch_network), the clock source driving capability is poor
- Thus, we can set the clock tree as an ideal network without driving issues
  - Avoid the hazard in the timing evaluation

{ Command Line }

```
set_ideal_network [get_ports clk]
```

Setting Clock Transition



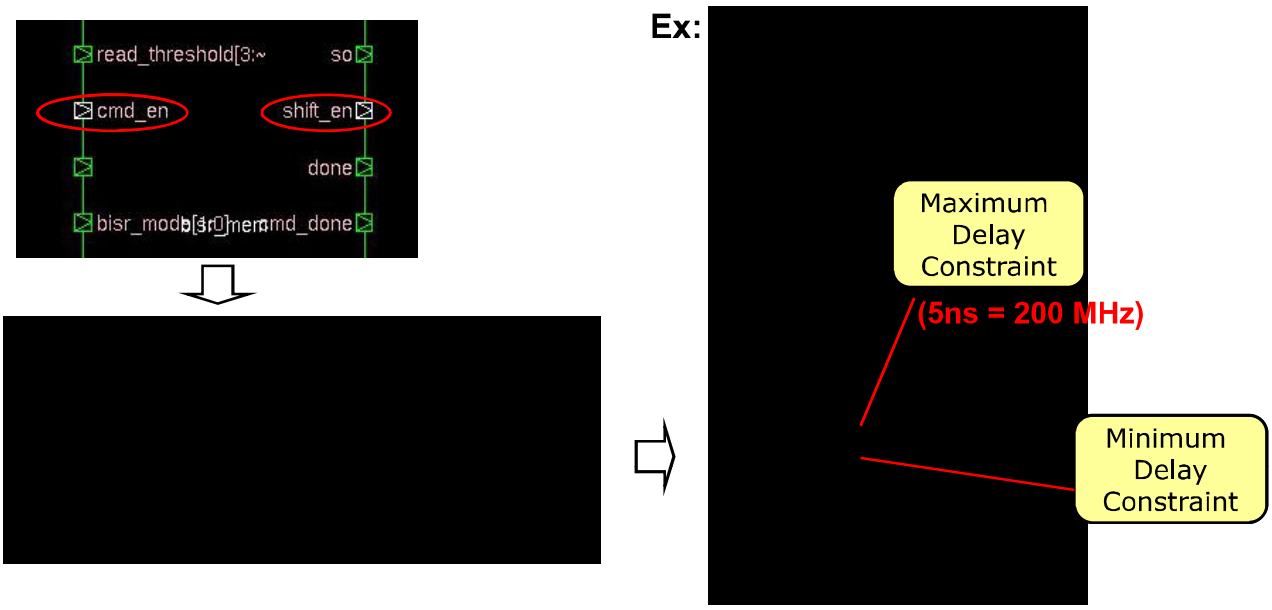
- experience
  - < 0.5ns
  - CIC tester: 0.5 ns

{ Command Line }

```
set_input_transition –max 0.5 [all_inputs]
```

Combination Circuit – Maximum Delay Constraints

- For combinational circuits primarily (i.e. design with no clock)
  - Select the start & end points of the timing path
  - ***Attributes/Optimization Constraints/Timing Constraints***



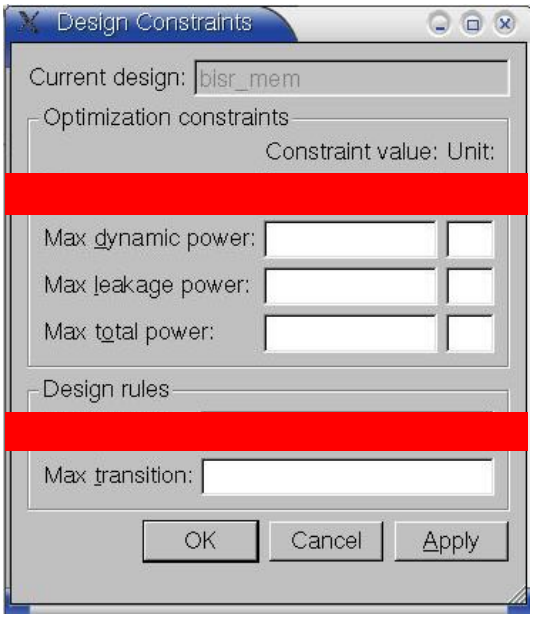
Design Rule Constraints

Setting Design Rule Constraints

- Area Constraint
- Fanout Constraint

Setting Area/Fanout Constraint

- ***Attributes/Optimization Constraints/Design Constraints***
- If you only concern the circuit area, but don't care about the timing
  - You can set the max area constraints to 0



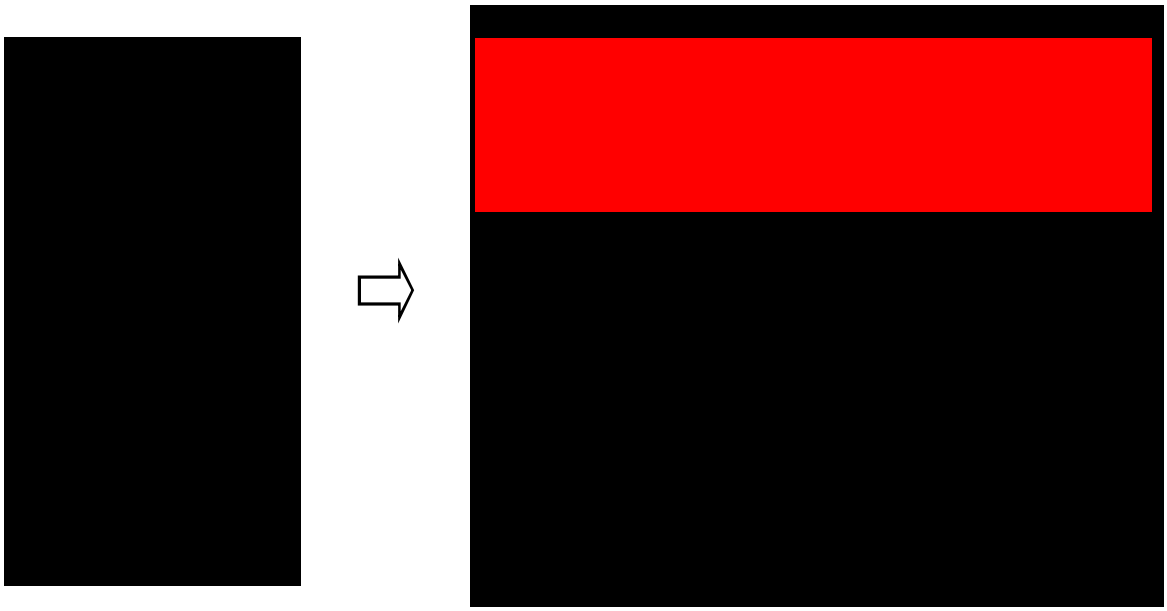
{ Command Line }

```
set_max_area 0
set_max_fanout 50 [get_designs CORE]
```

Compile the Design

Compile the Design

- ***Design/Compile Design***



{ Command Line }

compile –map_effort high –boundary_optimization

Assign Problem

Assign Violation Avoidance

- The syntax of “assign” may cause problems in the LVS

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

{ Command Line }

```
set_fix_multiple_port_nets –all –constants –buffer_constants [get_designs *]
```

Floating Port Removing

- Due to some ports in the standard cells are not used in your design

{ Command Line }

```
remove_unconnected_ports –blast_buses [get_cells –hierarchical *]
```

Chang Naming Rule Script

Naming Rule Changing

- Purpose: Let the naming-rule definitions in the gate-level netlist are the same as in the timing file (e.g. *.sdf file)
  - Also, the wrong naming rules may cause problems in the LVS

{ Command Line }

```
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"}}
define_name_rules name_rule -case_insensitive
change_names -hierarchy -rules name_rule
```

Save Design

Save Design

- Five design files:
  - *.spf: test protocol file for ATPG tools (i.e. TetraMax)
  - *.sdc: timing constraint file for P&R
  - *.vg: gate-level netlist for P&R
  - *.sdf: timing file for Verilog simulation
  - *.db: binary file (i.e. all the constraints and synthesis results are recorded)

{ Command Line }

```
write_test_protocol -f stil -out "CHIP.spf"
write_sdc CHIP.sdc
write -format verilog -hierarchy -output "CHIP.vg"
write_sdf -version 1.0  CHIP.sdf
write -format db    -hierarchy -output "CHIP.db"
```

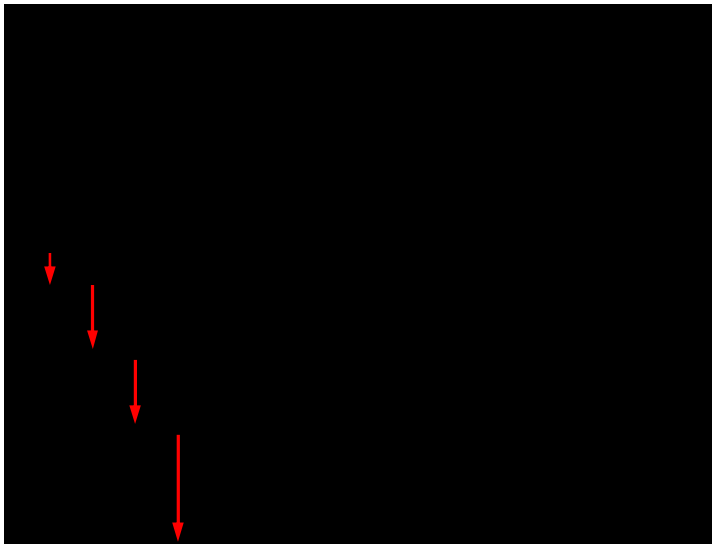
Synthesis Report

- Report Design Hierarchy
- Report Area
- Design View
- Report Timing
- Critical Path Highlighting
- Timing Slack Histogram

Report Design Hierarchy

- Hierarchy report shows the component used in your each block & its hierarchy
- ***Design/Report Design Hierarchy***

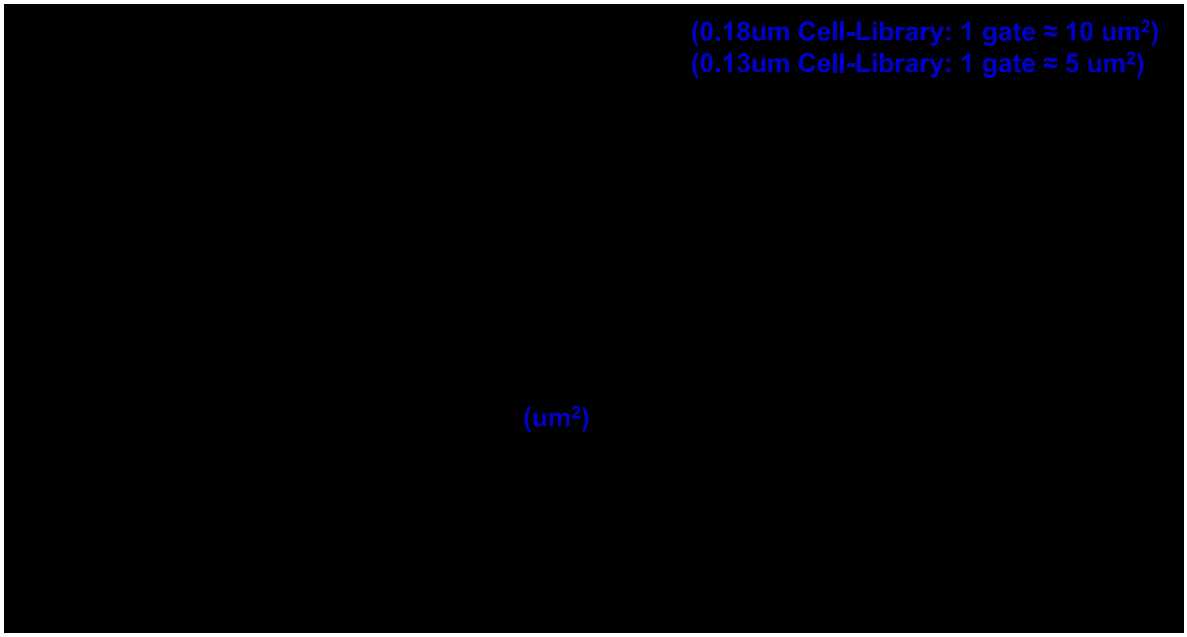
Ex:



Report Area

- *Design/Report Area*

Ex:

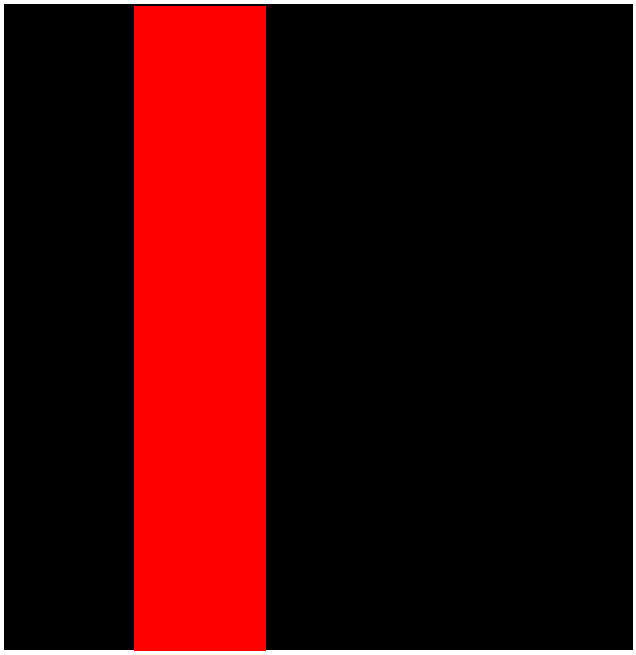


Design View

- ***List/Design View***

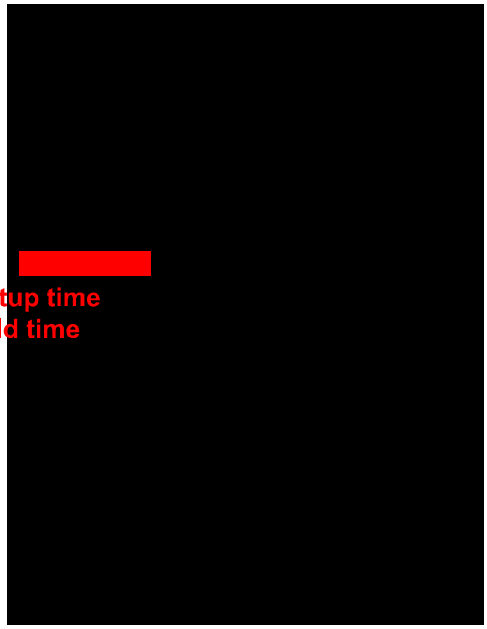
Ex:

All the block area are listed !!



Report Timing

- *Timing/Report Timing*



Ex:

setup time

Critical Path

```
Path Type: max

Point                                                   Incr
-----------------------------------------------------------
clock clk (rise edge)                                   5.00
clock network delay (ideal)                             0.00
memory_8k_64_2r_2c/sc_memory/CLK (sc_memory)            0.00
memory_8k_64_2r_2c/sc_memory/Q[0] (sc_memory)           0.71
memory_8k_64_2r_2c/aru/Q_sc[0] (aru)                    0.00
memory_8k_64_2r_2c/aru/U825/Y (AOI22X1)                 0.06
memory_8k_64_2r_2c/aru/U447/Y (INVX1)                   0.70
memory_8k_64_2r_2c/aru/U643/Y (AOI22X1)                 0.06
memory_8k_64_2r_2c/aru/U642/Y (OAI2BB1X1)               0.12
memory_8k_64_2r_2c/aru/data_out[47] (aru)               0.00
memory_8k_64_2r_2c/data_out[47] (memory_8k_64_2r_2c)
                                                        0.00
bisr/DO[47] (bisr)                                      0.00
bisr/bist/DO[47] (bist)                                 0.00
bisr/bist/m1/DO[47] (tpg)                               0.00
bisr/bist/m1/CMP/DO[47] (CMP)                           0.00
bisr/bist/m1/CMP/U207/Y (XNOR2X1)                       0.20
bisr/bist/m1/CMP/U185/Y (NAND4X1)                       0.07
bisr/bist/m1/CMP/U204/Y (NOR4X1)                        0.20
bisr/bist/m1/CMP/U202/Y (NAND4X1)                       0.06
bisr/bist/m1/CMP/U171/Y (AND2X2)                        0.13
bisr/bist/m1/CMP/U174/Y (INVX1)                         1.43
bisr/bist/m1/CMP/U170/Y (NAND2X2)                       0.64
bisr/bist/m1/CMP/U334/Y (OAI22X1)                       0.32
bisr/bist/m1/CMP/syndrome_reg[0]/D (DFFNRX1)            0.00
data arrival time

clock clk (fall edge)                                  10.00      10.00
clock network delay (ideal)                             0.00      10.00
bisr/bist/m1/CMP/syndrome_reg[0]/CKN (DFFNRX1)          0.00      10.00 f
library setup time                                     -0.09       9.91
data required time                                                 9.91
```

Slack = Data Require Time – Data Arrival Time

***** End Of Report *****

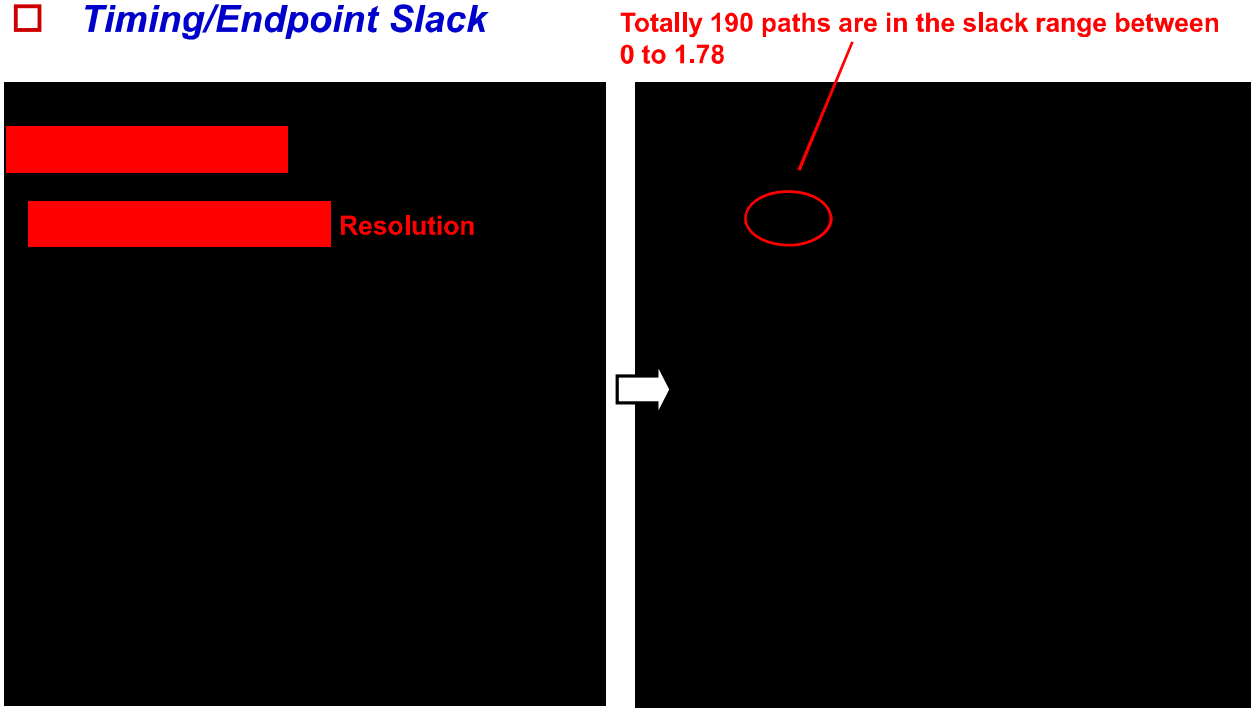
Critical Path Highlighting

- *View/Highlight/Critical Path*

Ex:

Timing Slack Histogram

- *Timing/Endpoint Slack*



Edit Your Own Script File

- For convenient, you should edit your own synthesis script file. Whenever you want to synthesis a new design, you just only change some parameters in this file.
- Execute Script File
  - ***File/Execute Script***
  - ***Or use "source your_ script.dc" in dc_shell command line***

Ex:

Gate-Level Simulation

- Include the Verilog model of standard cell and gate-level netlist to your test bench

- Add the following Synopsys directives to the test bench

Lab.

- cp –r –f /usr2/grad97/tjchen/tutorial_of_DV/Lab .