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^{TM} 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. –W. Tseng

Advanced Reliable
Systems (ARES) Lab.

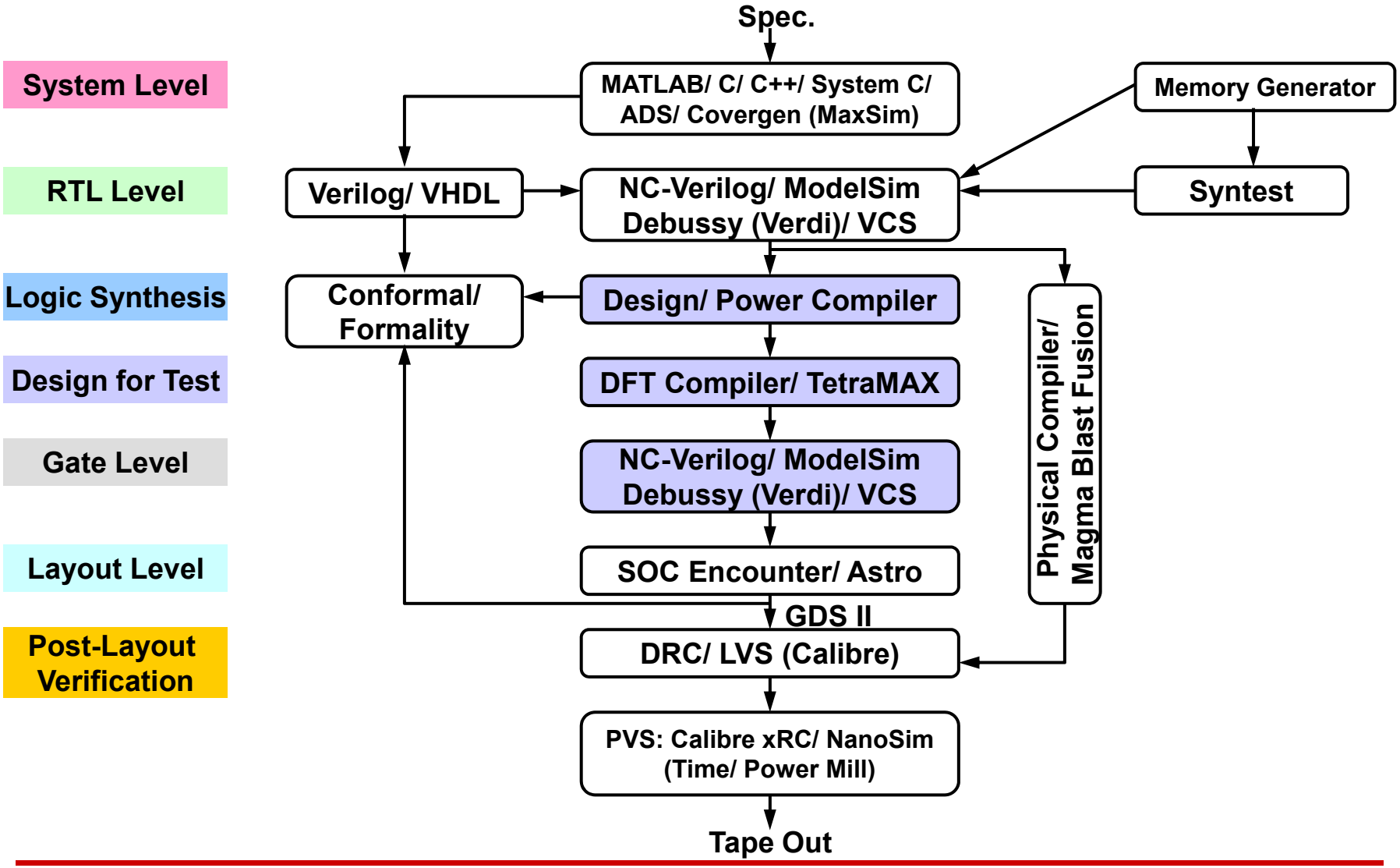
Advanced REliable System Lab

Outline

- Basic Concept of the Synthesis
- Synthesis Using Design Compiler
- Simulation-Based Power Estimation Using PrimePower
- Artisan Memory Compiler
- LAB

Basic Concept of the Synthesis

Cell-Based Design Flow



What is Synthesis

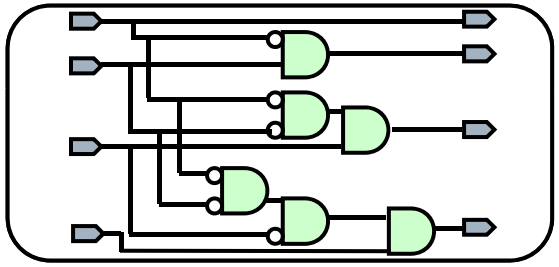
- Synthesis = translation + optimization + mapping

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

HDL Source (RTL)

Translate (HDL Compiler)

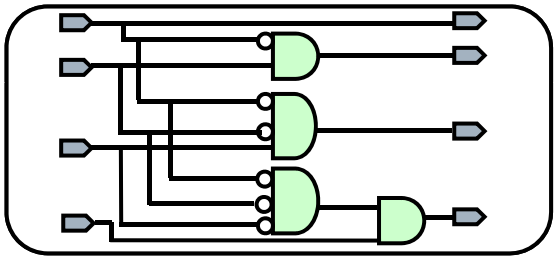
No Timing Info.



Generic Boolean (GTECT)

Optimize + Mapping (HDL Compiler)

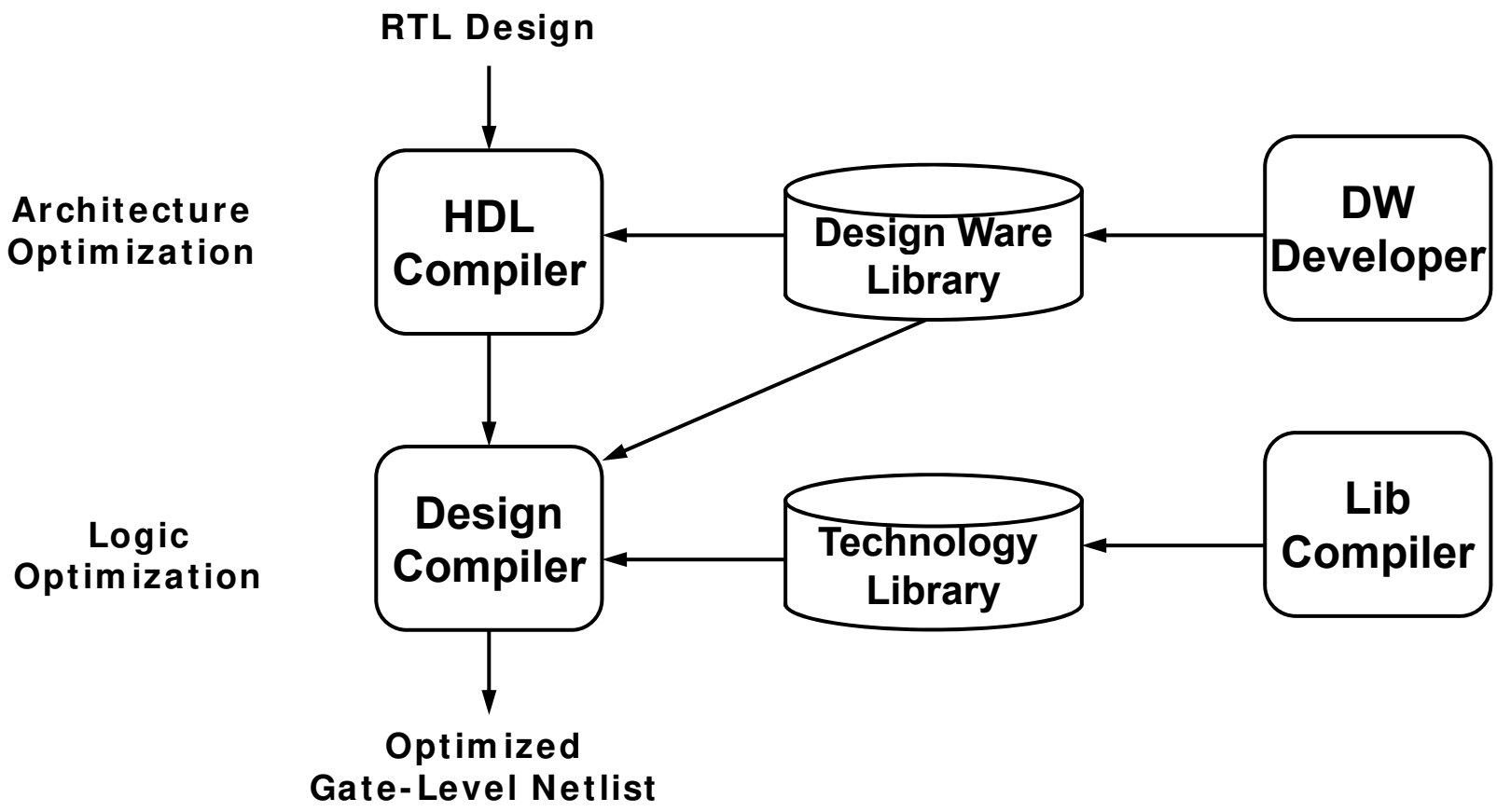
Timing Info.



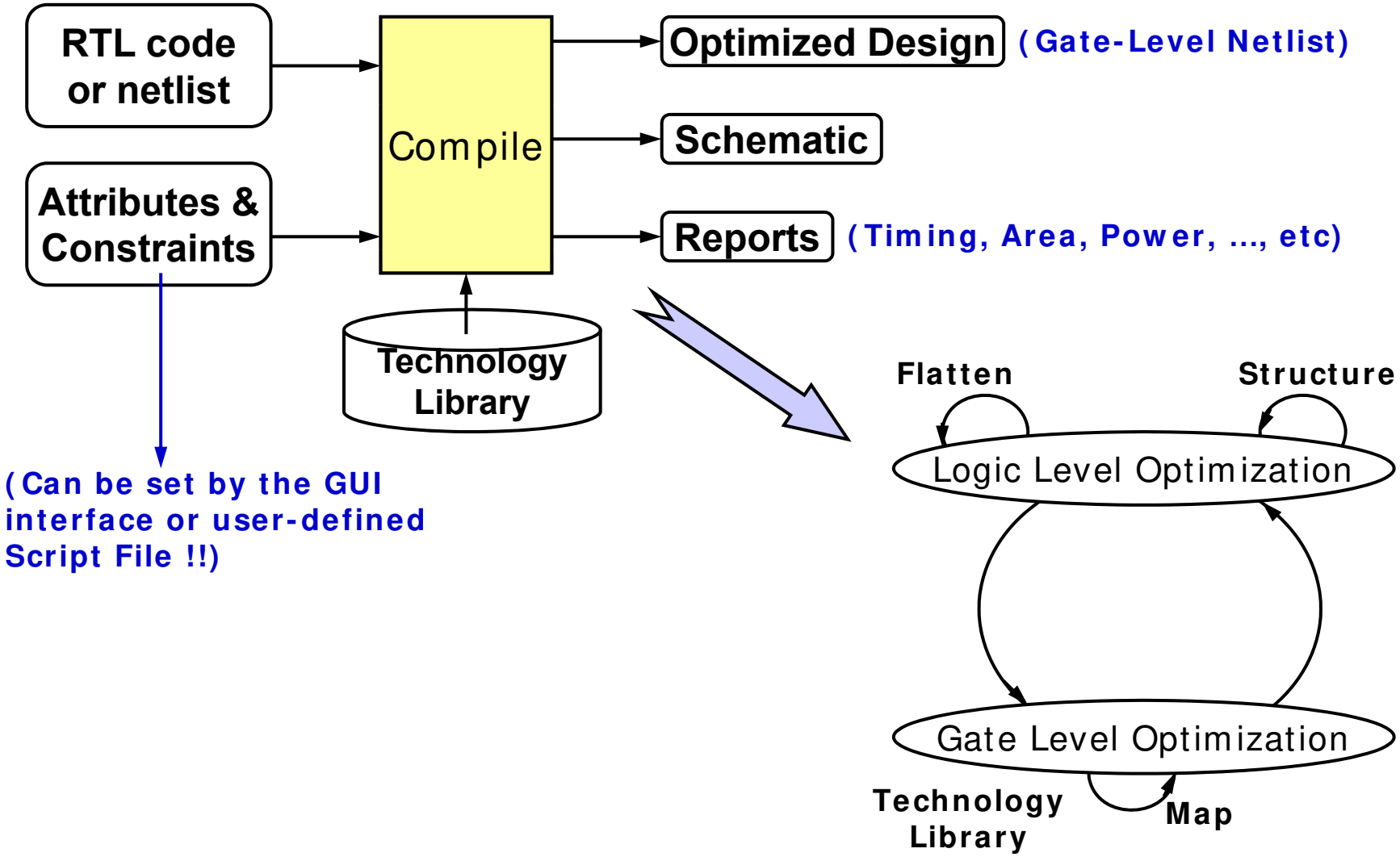
Target Technology

The synthesis is constraint driven and technology independent !!

Logic Synthesis Overview



Compile


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

Logic Level Optimization

- Operate with Boolean representation of a circuit
- Has a global effect on the overall area/speed characteristic of a design
- Strategy
  - Structure
  - Flatten (default OFF)
  - If both are true, the design is “first flattened and then structured”

Ex:

$f = acd + bcd + e$
$g = ae' + be'$
$h = cde$

⇨

$f = xy + e$
$g = xe'$
$h = ye$
$x = a + b$
$y = cd$

(Structure)

$f0 = at$
$f1 = d + t$
$f2 = t'e$
$t = b + c$

⇨

$f0 = ab + ac$
$f1 = b + c + d$
$f2 = b'c'e$

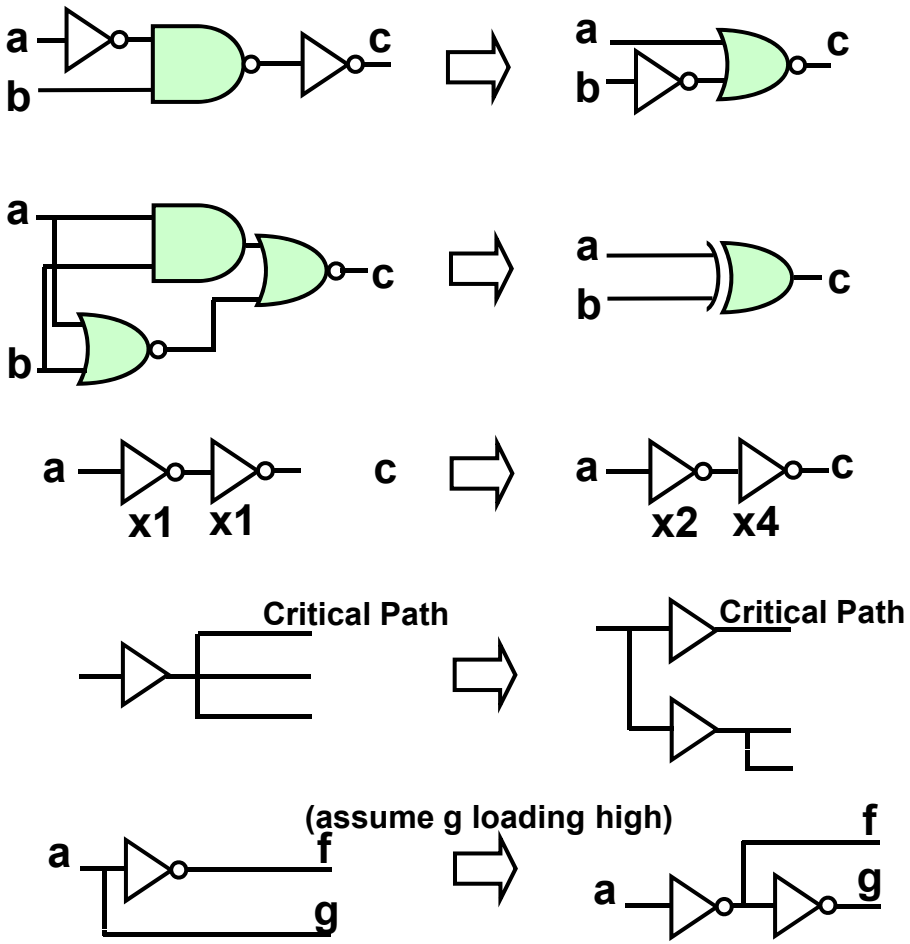
(Flatten)

Gate Level Optimization - Mapping

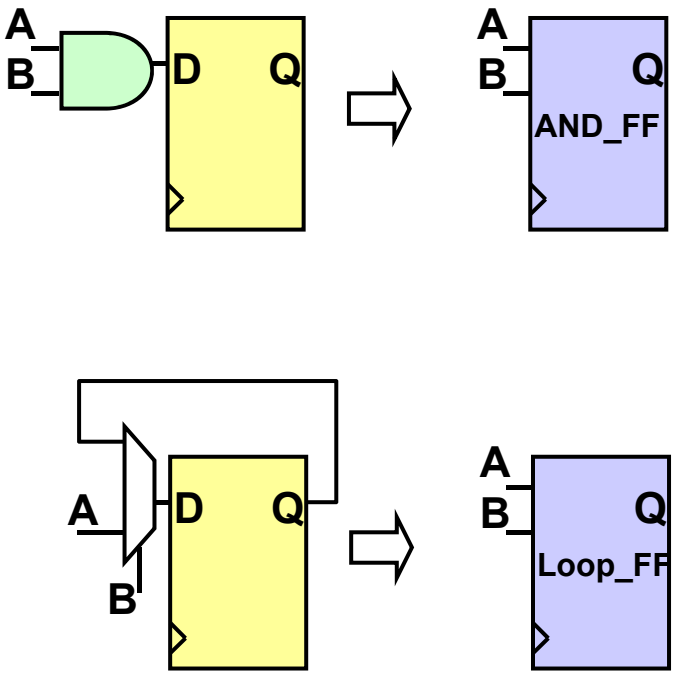
- Combinational Mapping
  - Mapping rearranges components, combining and re-combining logic into different components
  - May use different algorithms such as cloning, resizing, or buffering
  - Try to meet the design rule constraints and the timing/area goals
- Sequential Mapping
  - Optimize the mapping to sequential cells technology library
  - Analyze combinational logics surrounding a sequential cell to see if it can absorb the logic attribute with HDL
  - Try to save speed and area by using a more complex sequential cells

Mapping

Combinational Mapping

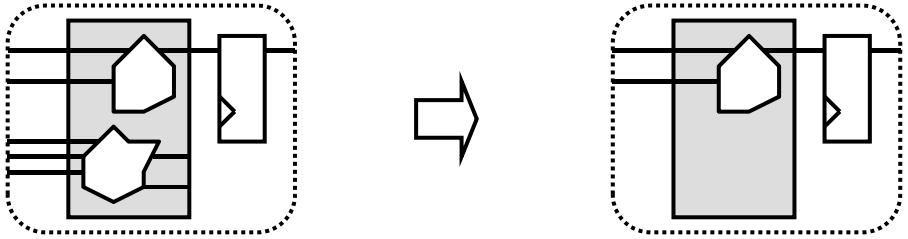


Sequential Mapping

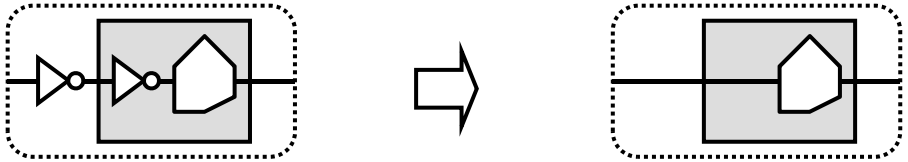


Boundary Optimization

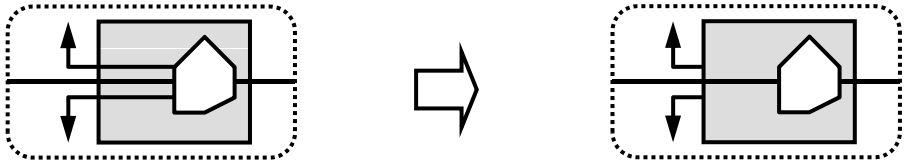
- Design Compiler can do some optimizations across boundaries
  1. Removes logic driving unconnected output ports



  2. Removes redundant inverters across boundaries



  3. Propagates constants to reduce logic



Static Timing Analysis

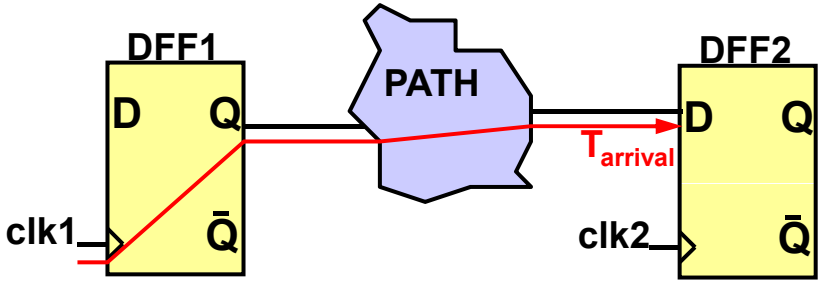
- Main steps of STA
  - Break the design into sets of timing paths
  - Calculate the delay of each path
  - Check all path delays to see if the given timing constraints are met
- Four types of paths
  - Register - Register (Reg - Reg)
  - Primary Input - Register (PI - Reg)
  - Register - Primary Output (Reg - PO)
  - Primary Input - Primary Output (PI - PO)

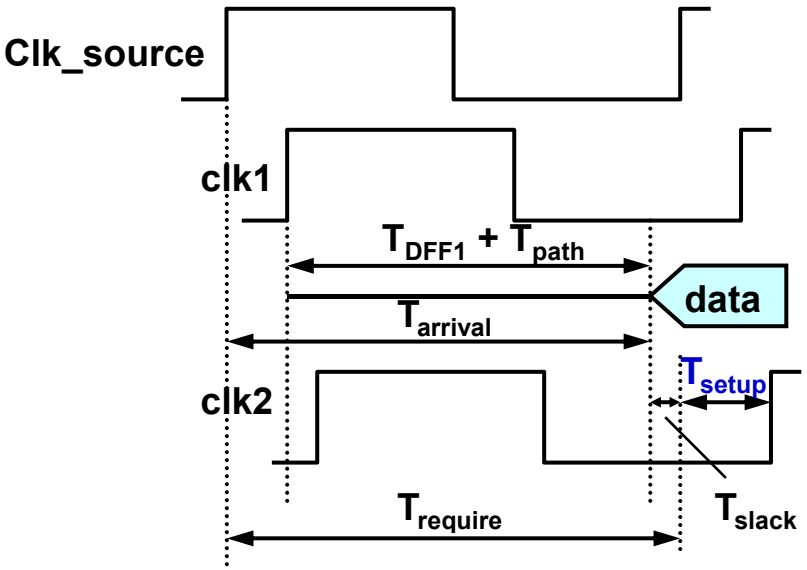
Static Timing Analysis (Cont')

- Setup Time

- To meet the setup time requirement:
  - $T_{require} >= T_{arrival}$
- Reg to Reg
  - $T_{arrival} = T_{clk1} + T_{DFF1(clk->Q)} + T_{PATH}$
  - $T_{require} = T_{clk2} - T_{DFF2(setup)}$
  - $T_{slack} = T_{require} - T_{arrival}$

(T_{slack} > 0 denotes "no timing violation")

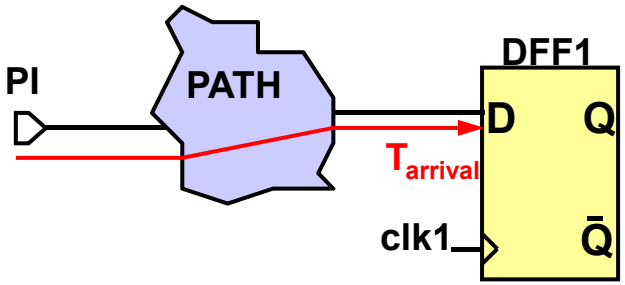


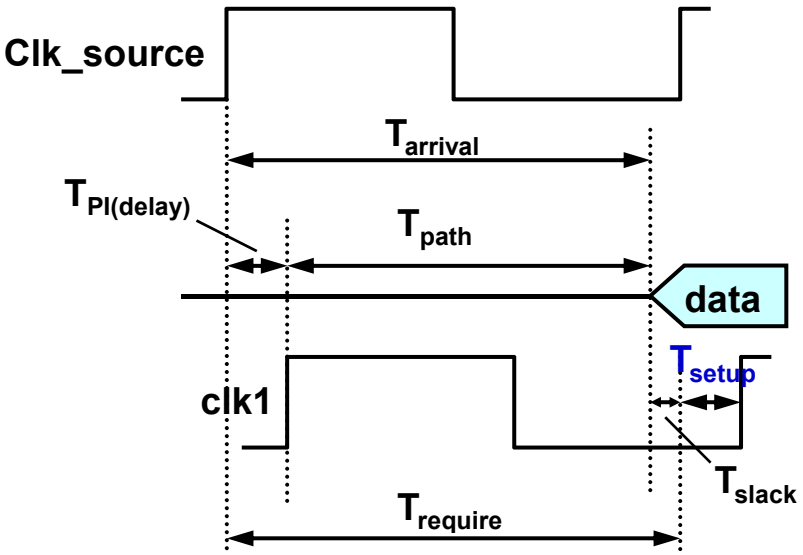


Static Timing Analysis (Cont')

- Setup Time

- PI to Reg
  - $T_{arrival} = T_{PI(delay)} + T_{PATH}$
  - $T_{require} = T_{clk1} - T_{DFF1(setup)}$
  - $T_{slack} = T_{require} - T_{arrival}$



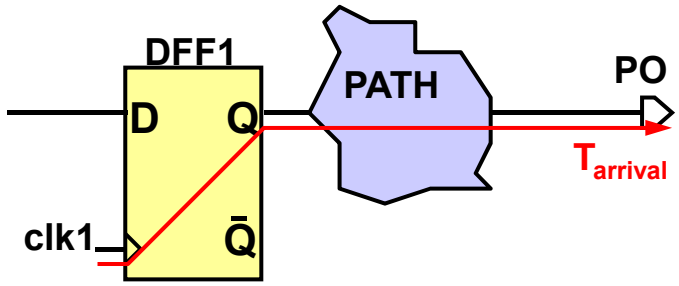


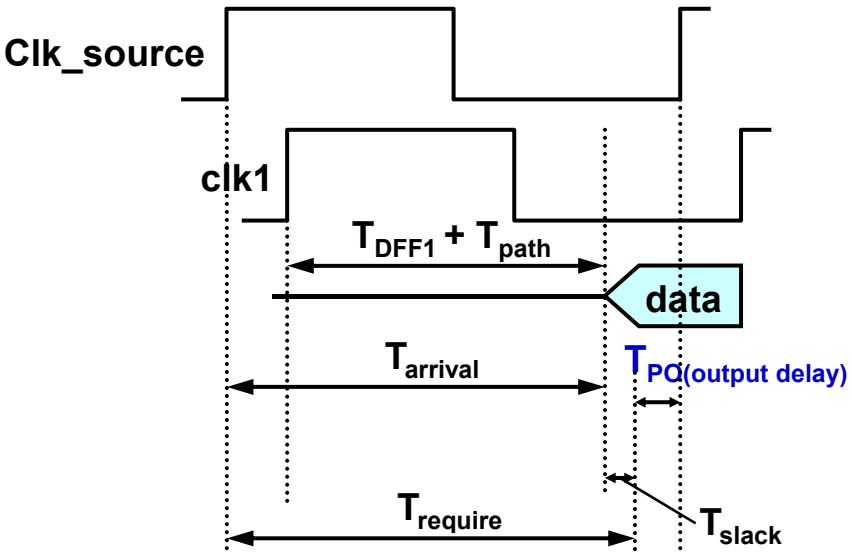
Static Timing Analysis (Cont')

- Reg to PO

- Setup Time

- $T_{arrival} = T_{clk1} + T_{DFF1(clk\text{->}Q)} + T_{PATH}$
- $T_{require} = T_{cycle} - T_{PO(output\ delay)}$
- $T_{slack} = T_{require} - T_{arrival}$

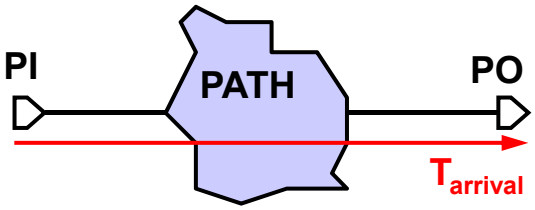


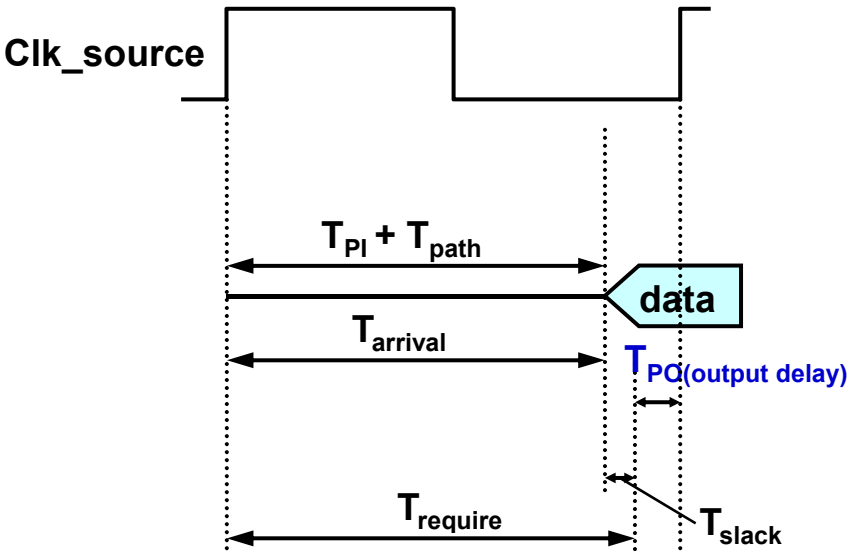


Static Timing Analysis (Cont')

- Setup Time

- PI to PO
  - $T_{arrival} = T_{PI(delay)} + T_{PATH}$
  - $T_{require} = T_{cycle} - T_{PO(output\ delay)}$
  - $T_{slack} = T_{require} - T_{arrival}$

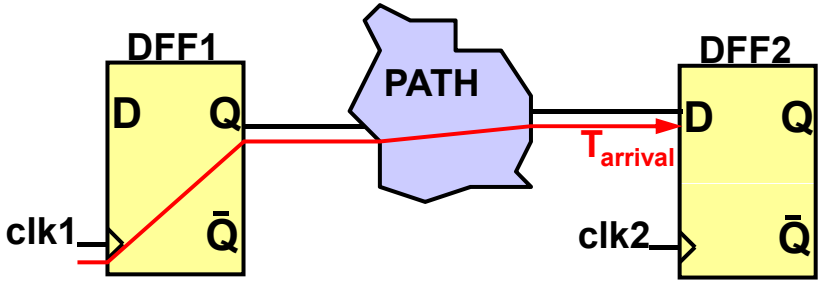


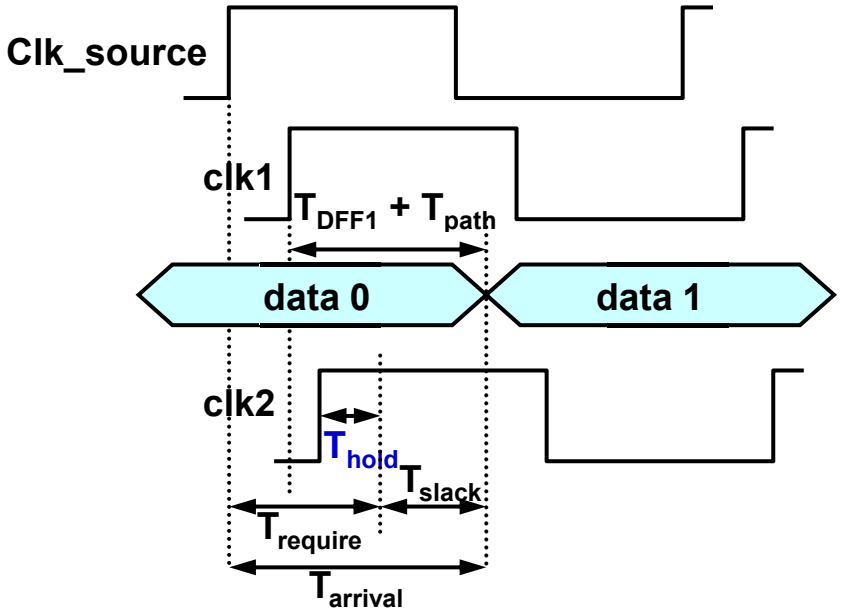


Static Timing Analysis (Cont')

- Hold Time

- To meet the hold time requirement:
  - $T_{require} <= T_{arrival}$
- Reg to Reg
  - $T_{arrival} = T_{clk1} + T_{DFF1(clk->Q)} + T_{PATH}$
  - $T_{require} = T_{clk2} + T_{DFF2(hold)}$
  - $T_{slack} = T_{arrival} - T_{require}$





Static Timing Analysis (Cont')

- Hold Time

- PI to Reg
  - $T_{arrival} = T_{PI(delay)} + T_{PATH}$
  - $T_{require} = T_{clk1} + T_{DFF(hold)}$
  - $T_{slack} = T_{arrival} - T_{require}$
- Reg to PO
  - $T_{arrival} = T_{clk1} + T_{DFF(clk->Q)} + T_{PATH}$
  - $T_{require} = - T_{PO(output\ delay)}$
  - $T_{slack} = T_{arrival} - T_{require}$
- PI to PO
  - $T_{arrival} = T_{PI(delay)} + T_{PATH}$
  - $T_{require} = - T_{PO(output\ delay)}$
  - $T_{slack} = T_{arrival} - T_{require}$

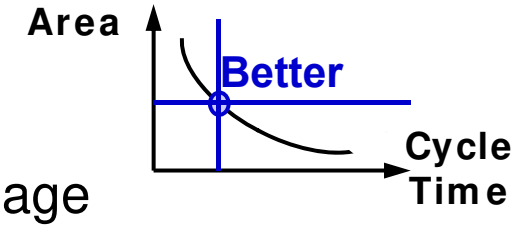
Synthesizable Verilog

- Verilog Basis
  - parameter declarations
  - wire, wand, wor declarations
  - reg declarations
  - input, output, inout
  - continuous assignment
  - module instructions
  - gate instructions
  - always blocks
  - task statement
  - 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

<.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:

Text Editor – .synopsys_dc.setup

File Edit Format Options

```
#######0.18um
set search_path        "/usr1/teacher/jfli/cell_lib/CBDK018_TSMC_Artisan/CIC/SynopsysDC $search_path"
set search_path        "/usr/cad/synopsys/synthesis/cur/libraries/syn  $search_path"
set search_path        "/usr1/teacher/jfli/cell_lib/CBDK018_TSMC_Artisan/orig_lib/aci/sc/symbols/synopsys $search_path"
set search_path        "/usr4/grad92/zwtseng/HOY/rom_base_bist/8_bit_bist/bist_256x8_with_ECC/0331_bist_FJU_ARTISAN/MEM $search_path"

######with MEM#####
#set link_library       "RA1SHD256x8_fast@-40C_syn.db RA1SHD256x8_fast@0C_syn.db RA1SHD256x8_typical_syn.db RA1SHD256x8_slow_syn.db typ
set link_library       RA1SHD256x8_fast@-40C_syn.db RA1SHD256x8_fast@0C_syn.db RA1SHD256x8_typical_syn.db RA1SHD256x8_slow_syn.db typ
set target_library     RA1SHD256x8_fast@-40C_syn.db RA1SHD256x8_fast@0C_syn.db RA1SHD256x8_typical_syn.db RA1SHD256x8_slow_syn.db typ

######without MEM#####
#set link_library       "typical.db fast.db slow.db tpz973gbc.db tpz973gtc.db tpz973gwc.db dw_foundation.sldb dw01.sldb dw02.sldb dw03.
#set target_library     "typical.db fast.db slow.db"

set symbol_library     "tsmc18.sdb"
#set synthetic_library  "dw_foundation.sldb dw01.sldb dw02.sldb dw03.sldb dw04.sldb dw05.sldb dw06.sldb dw07.sldb dw08.sldb"


set hdlin_translate_off_skip_text "TRUE"
set edifout_netlist_only "TRUE"
set verilogout_no_tri true
set plot_command {lpr -Plp}
```

MEM Libraries (.db file)

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

(.synopsys_dc.setup File)

Settings for Using Memory

- Convert *.lib to *.db
  - *%> dc_shell –t*
  - *dc_shell-t> read_lib t13spsram512x32_slow_syn.lib*
  - *dc_shell-t> write_lib t13spsram512x32 -output \ t13spsram512x32_slow_syn.db*

any memory LIB file

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

memory DB file

- 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 to the file

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

```
//`include "sr_memory_1k.v"

module bisr_mem(clk,rst,ams,CSI
,bisr_mode,cmd_done,BGO,CSO,shi

parameter WORD_LENGTH = 64;
parameter ADR_LEN  = 13;
```

Test Pins Reservation

- You can add the floating test pins to your design before synthesis
  - se: scan enable
  - si: scan input
  - so: scan output
  - scantest: control signal for memory shadow wrapper (i.e. memory is used)
- Ex:

```
input  [WORD_LENGTH-1:0] DI_S;
input  [ADR_LEN-1:0]     ADDR_S;
input                    bira_en;
input                    test_done;
input        [1:0]       bisr_mode;
//----pins for scans----
input                    se;
input                    si;
input                    scantest;
output                   so;
```

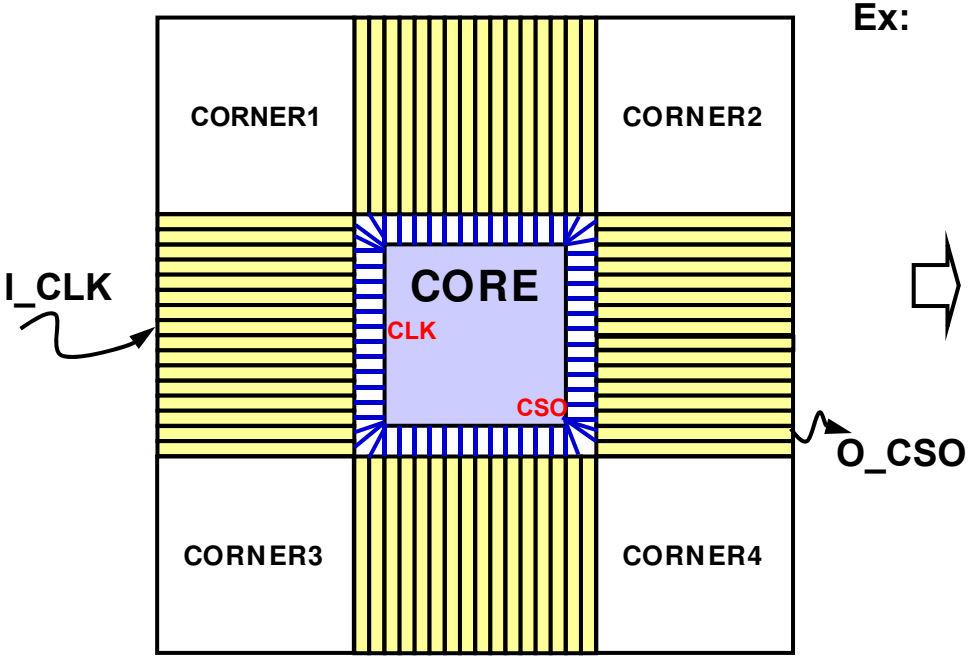
Normal IO Declaration

Test IO Declaration

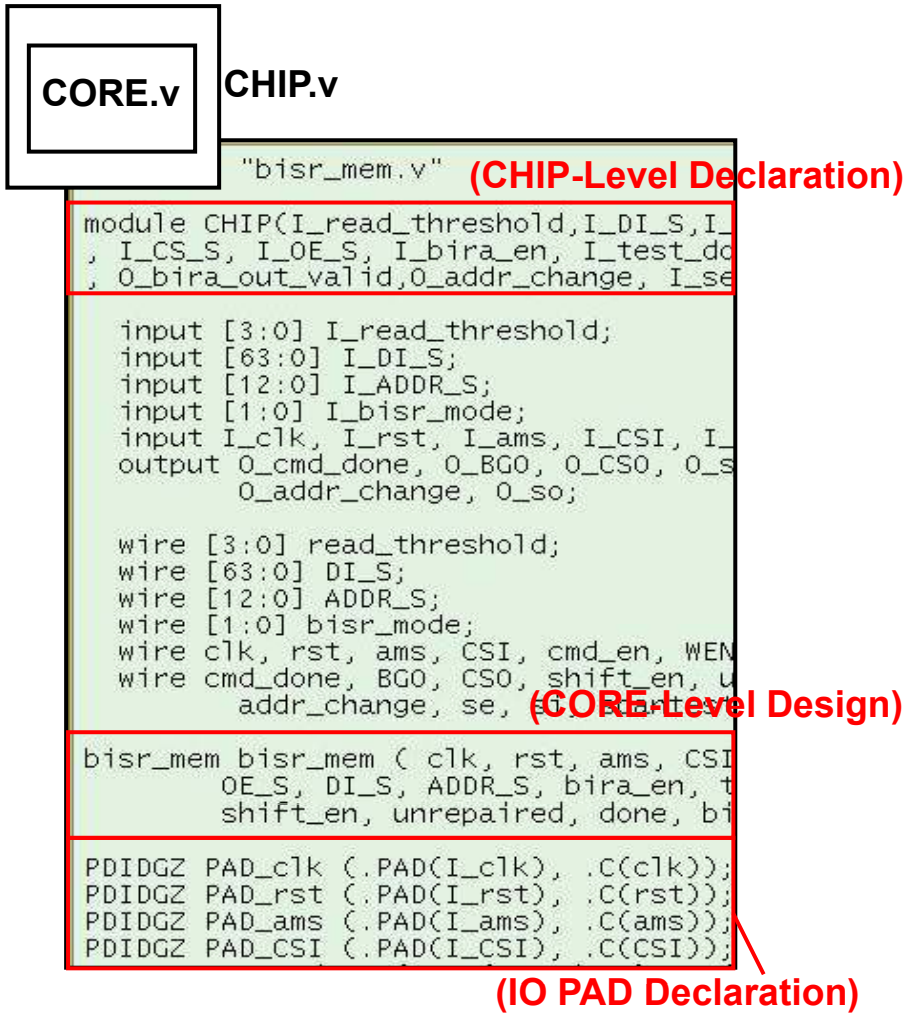
- The pins will be connected to scans after the scan chain insertion

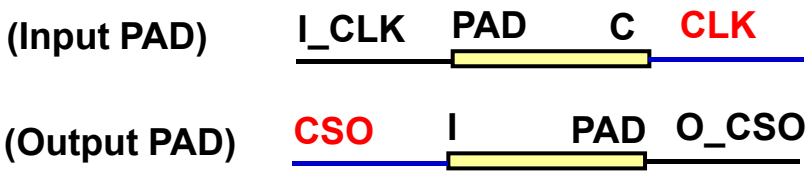
CHIP-Level Netlist

- The CHIP-level netlist consists of your Core-level netlist and the PADs



Ex:





How To Choose the I O PAD

- You can reference the Databook of the IO PAD in CIC Design Kit
- Generally, the “PDIDGZ” is used as the input PAD
- Trade-off when considering the output PAD
  - High driving → SSN ↑
  - Low driving → Delay ↑

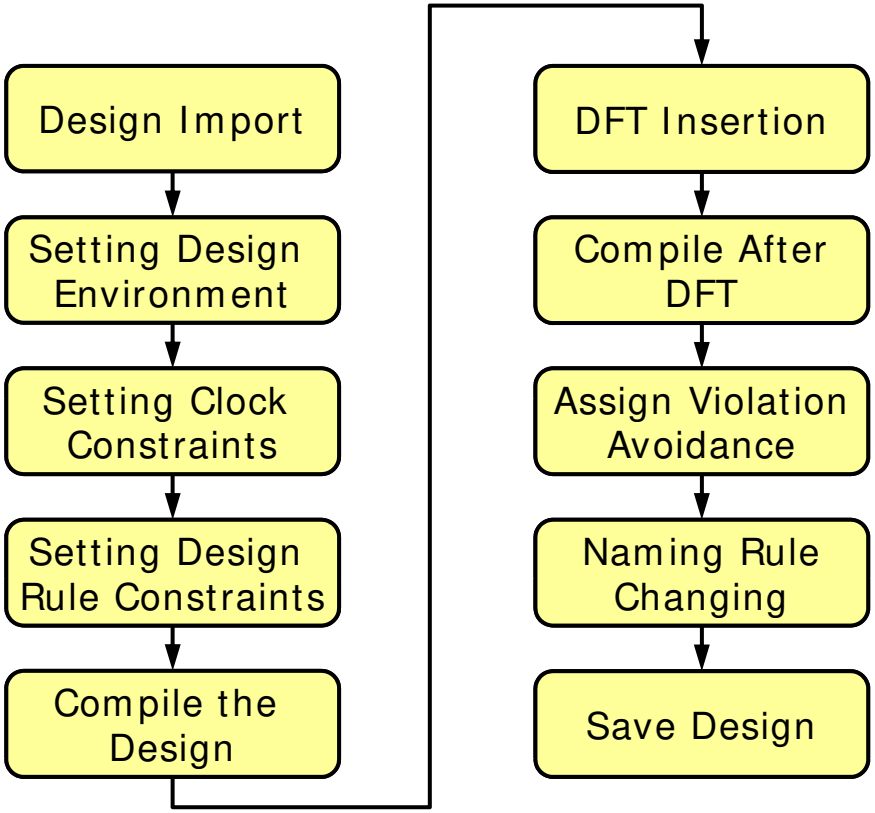
➢ Note that the loading of the CIC tester is 40pf

Ex:

Cell Name	Path	Parameter	Group1	Group2	Group3
			(< 20.00000)pf	(20.00000~70.00000)pf	(> 70.00000)pf
PDO02CDG	I->PAD	t_{PLH}	2.5270+0.2584*Cload	2.5400+0.2580*Cload	2.5267+0.2583*Cload
		t_{PHL}	2.0840+0.2787*Cload	2.0700+0.2790*Cload	2.0833+0.2787*Cload
Cell Name	**Path**	**Parameter**	**Group1**	**Group2**	**Group3**
			(< 20.00000)pf	(20.00000~70.00000)pf	(> 70.00000)pf
PDO04CDG	I->PAD	t_{PLH}	2.1950+0.1292*Cload	2.1980+0.1291*Cload	2.1767+0.1293*Cload
		t_{PHL}	1.8650+0.1403*Cload	1.8830+0.1394*Cload	1.8867+0.1393*Cload
Cell Name	**Path**	**Parameter**	**Group1**	**Group2**	**Group3**
			(< 40.00000)pf	(40.00000~100.00000)pf	(> 100.00000)pf
PDO08CDG	I->PAD	t_{PLH}	2.4200+0.0651*Cload	2.4440+0.0646*Cload	2.4490+0.0645*Cload
		t_{PHL}	2.1400+0.0721*Cload	2.2430+0.0698*Cload	2.2630+0.0696*Cload
Cell Name	**Path**	**Parameter**	**Group1**	**Group2**	**Group3**
			(< 40.00000)pf	(40.00000~100.00000)pf	(> 100.00000)pf
PDO12CDG	I->PAD	t_{PLH}	2.7010+0.0457*Cload	2.8060+0.0433*Cload	2.8280+0.0431*Cload
		t_{PHL}	2.3840+0.0523*Cload	2.5930+0.0477*Cload	2.6840+0.0467*Cload
Cell Name	**Path**	**Parameter**	**Group1**	**Group2**	**Group3**
			(< 50.00000)pf	(50.00000~120.00000)pf	(> 120.00000)pf
PDO16CDG	I->PAD	t_{PLH}	3.0230+0.0339*Cload	3.1813+0.0330*Cload	3.2480+0.0324*Cload
		t_{PHL}	2.6615+0.0411*Cload	2.8910+0.0370*Cload	3.0548+0.0354*Cload
Cell Name	**Path**	**Parameter**	**Group1**	**Group2**	**Group3**
			(< 50.00000)pf	(50.00000~120.00000)pf	(> 120.00000)pf
PDO24CDG	I->PAD	t_{PLH}	3.5425+0.0293*Cload	3.8240+0.0244*Cload	4.0332+0.0224*Cload
		t_{PHL}	3.1785+0.0341*Cload	3.5245+0.0280*Cload	3.8053+0.0253*Cload

[REF: TPZ973G TSMC 0.18um Standard I/O Library Databook, Version 240a, December 10, 2003]

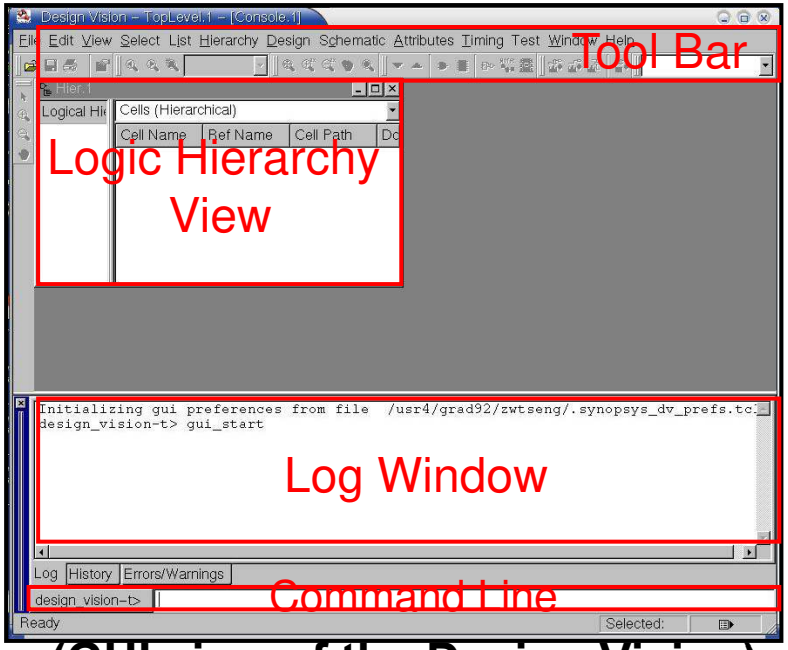
Synthesis Flow



Getting Started

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

linux %> source /APP/cshbank/synthesis.csh
linux %> source /APP/verdi/CIC/debussy.cshrc
linux %> dv –db_mode& (DB Mode)
or
linux %> dv& (XG Mode)

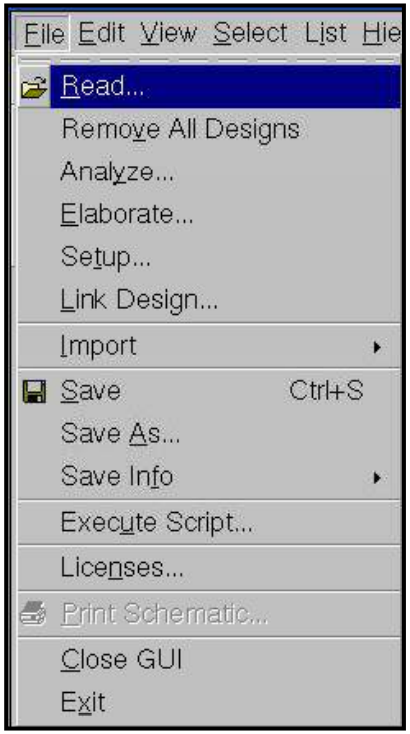


(GUI view of the Design Vision)

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

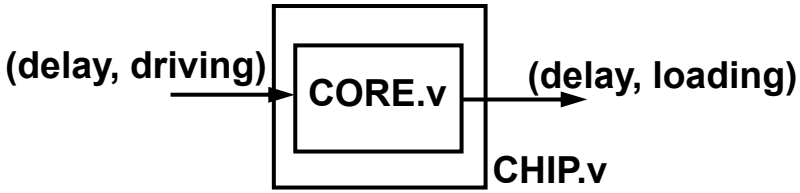


{ Command Line }

read_file –format verilog **file name**

PAD Parameters Extraction

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




```
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {ADDR_S[0]}]
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {bira_en}]
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {test_done}]
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {bisr_mode[1]}]
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {bisr_mode[0]}]
set_load -pin_load 0.06132 [get_ports {cmd_done}]
set_load -pin_load 0.06132 [get_ports {BGO}]
set_load -pin_load 0.06132 [get_ports {CSO}]
set_load -pin_load 0.06132 [get_ports {shift_en}]
set_load -pin_load 0.06132 [get_ports {unrcpaircd}]
set_load -pin_load 0.06132 [get_ports {done}]
set_load -pin_load 0.06132 [get_ports {bira_out_valid}]
set_load -pin_load 0.06132 [get_ports {addr_change}]
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {se}]
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {si}]
set_driving_cell -lib_cell PDIDGZ -library tpz973gbc -pin C -from_pin PAD \
-no_design_rule [get_ports {scantest}]
set_load -pin_load 0.06132 [get_ports {so}]
```

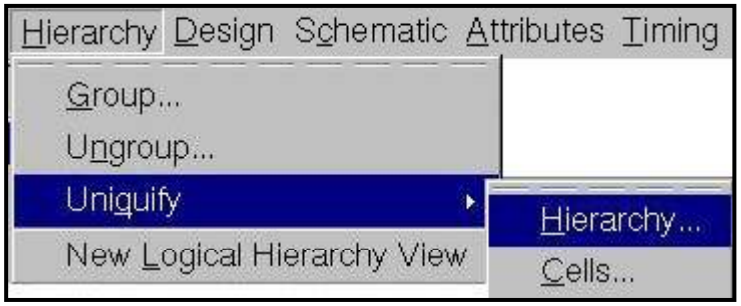
(chip_const.tcl)

{ 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***



Name	Design Area	Dont Touch
MEM	0	undefined
ROM	0	undefined
SES_ID	0	undefined
SYN_DEC_8_0	0	undefined
SYN_DEC_8_1	0	undefined
SYN_DEC_8_2	0	undefined
SYN_DEC_8_3	0	undefined
SYN_DEC_8_4	0	undefined
SYN_DEC_8_5	0	undefined
SYN_DEC_8_6	0	undefined
SYN_DEC_8_7	0	undefined
addr_present	0	undefined
addr_previous1	0	undefined
addr_previous2	0	undefined
b_to_g_0	0	undefined
b_to_g_1	0	undefined

(Design View)

```
design_vision-xg-t> uniquify
Removing uniquified design 'b_to_g'.
Removing uniquified design 'SYN_DEC_8'.
  Uniquified 2 instances of design 'b_to_g'.
  Uniquified 8 instances of design 'SYN_DEC_8'.
```

(Log Window)

{ Command Line }

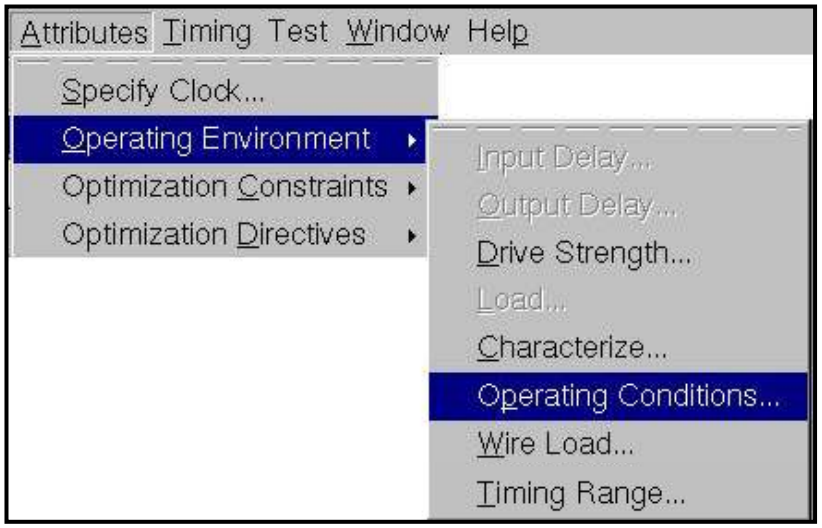
uniquify

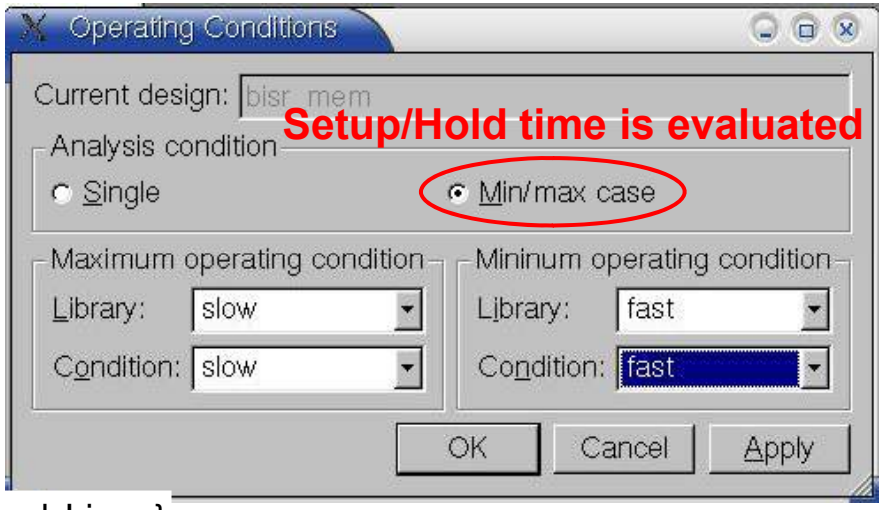
Design Environment

- 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***



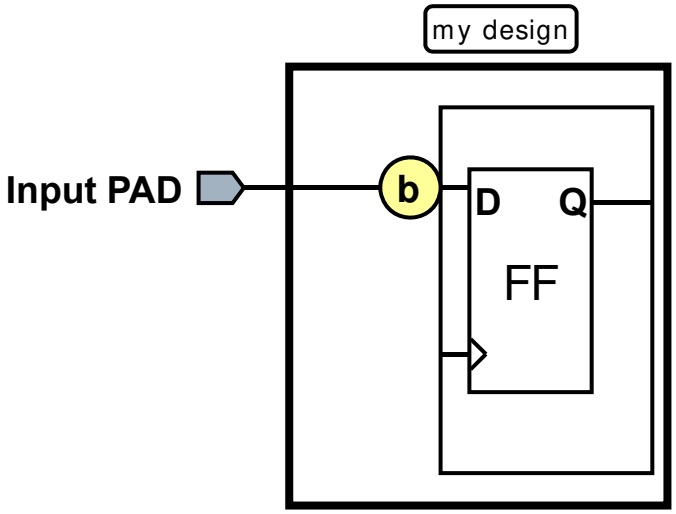


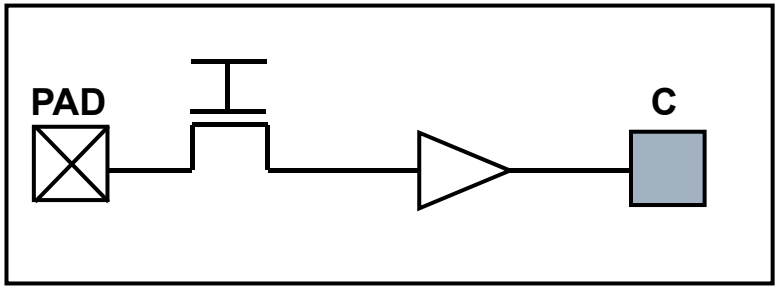
{ Command Line }

```
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”





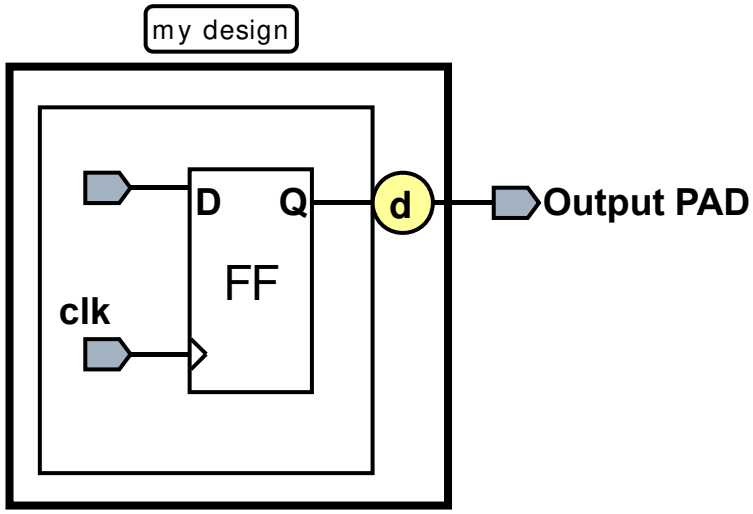
(PDIDGZ)

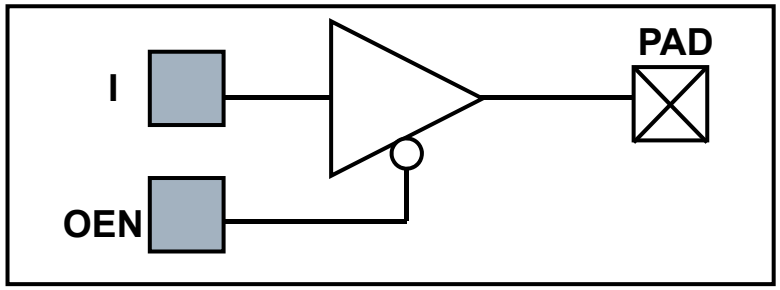
{ Command Line }

```
set_drive [expr 0.288001] [all_inputs]
set_input_delay [expr 0.34] [all_inputs]
```

Setting Load/ Output Delay for PADs

- Assume that we use the output PAD “PDO24CDG”





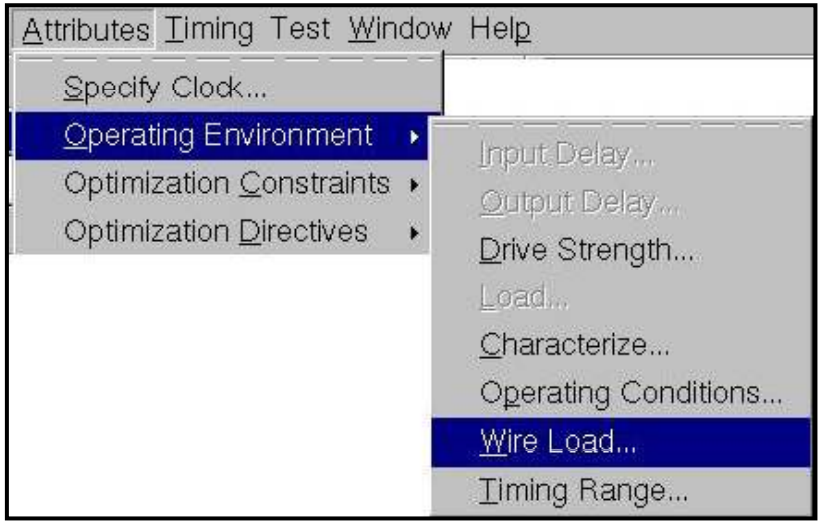
(PDO24CDG)

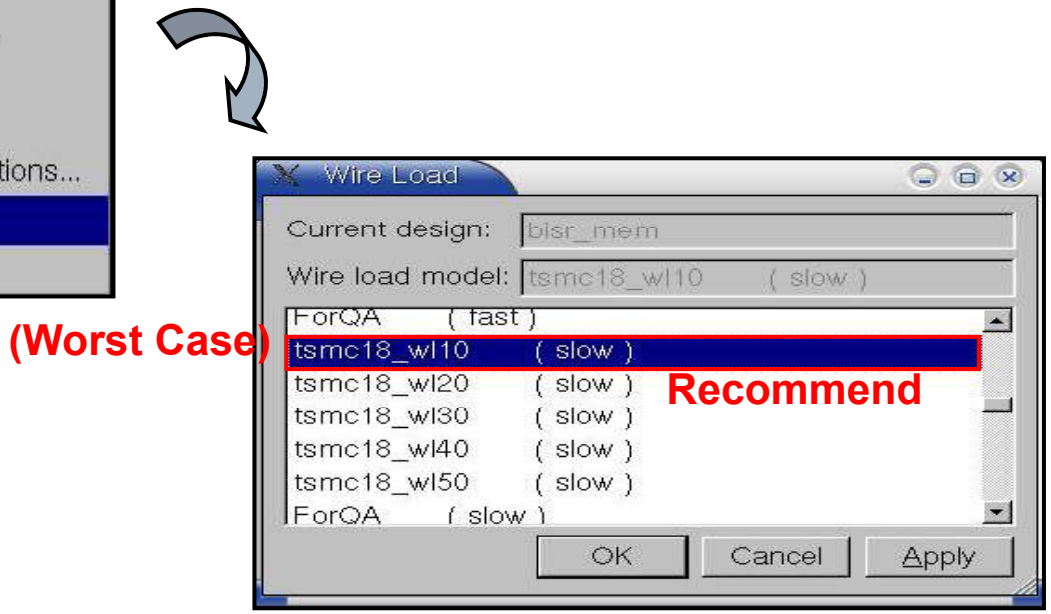
{ Command Line }

```
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 }

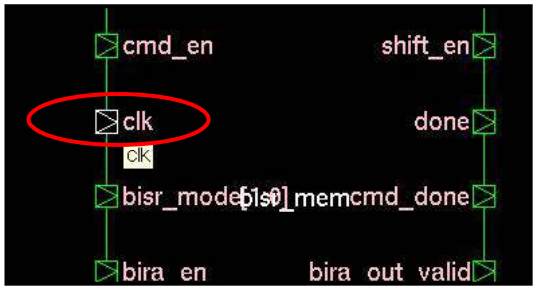
```
set_wire_load_model –name “tsmc18_wl10” –library “slow”
set_wire_lode_mode “top”
```

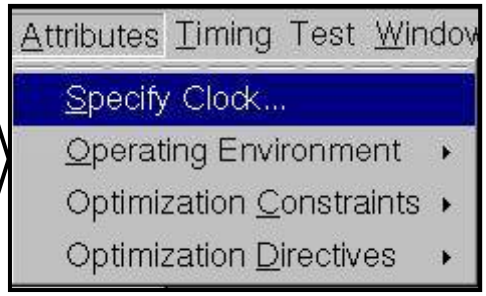
Clock Constraints

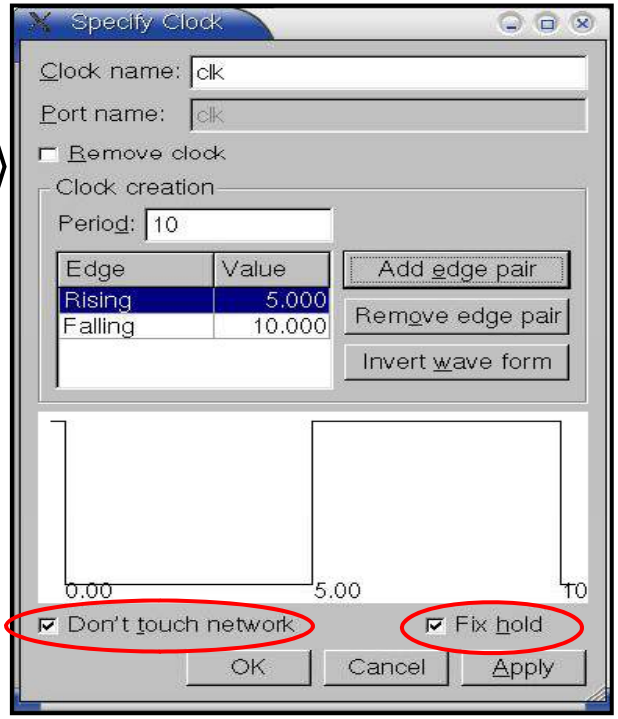
- Period
- Waveform
- Uncertainty
  - Skew
- Latency
  - Source latency
  - Network latency
- Transition
  - Input transition
  - Clock transition
- Combination Circuit – Maximum Delay Constraints

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

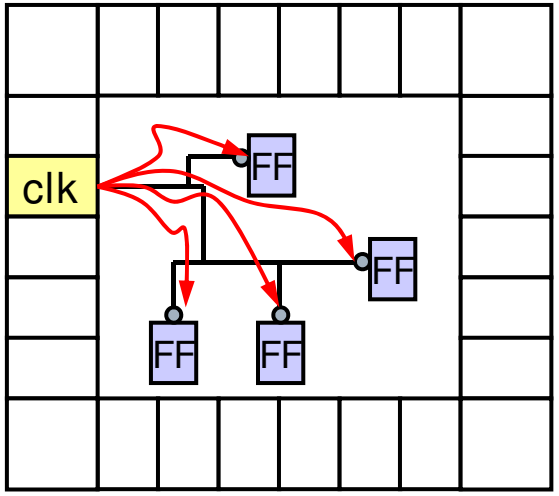
{ Command Line }

```
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:




```
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
```

(Timing Report)

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

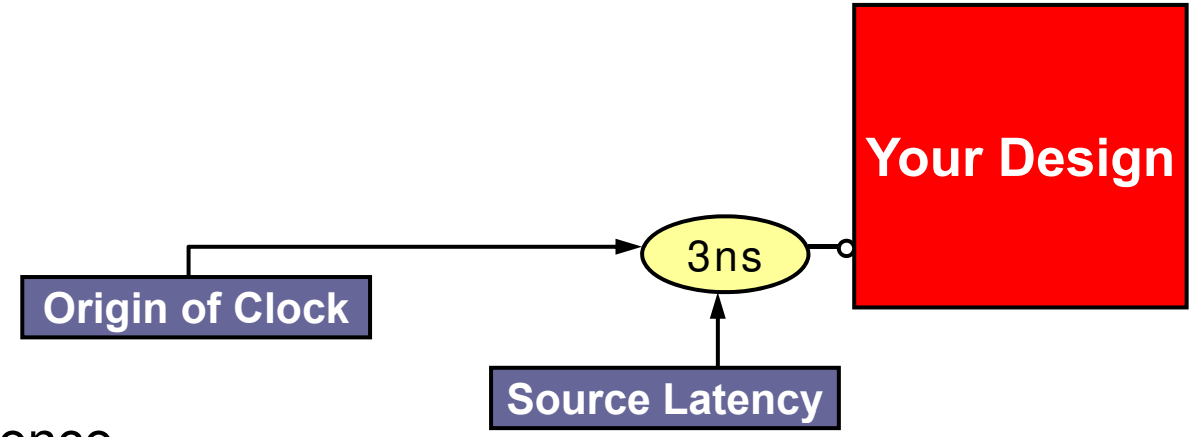
{ Command Line }

```
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]
```

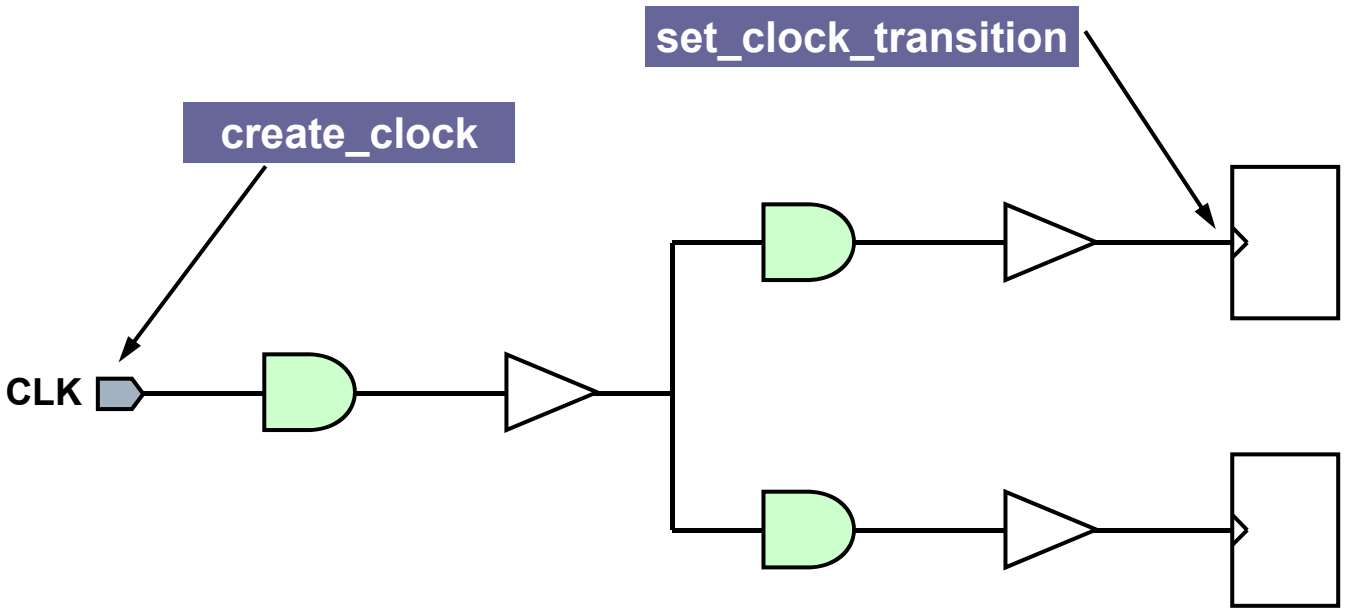
Setting Ideal Clock

- 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_clocks clk]
```

Setting Clock Transition



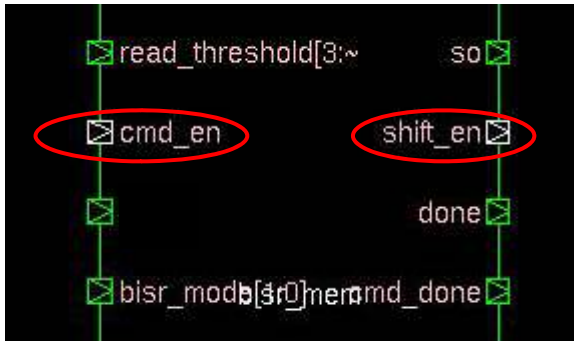
- experience
  - < 0.5ns
  - CIC tester: 0.5 ns

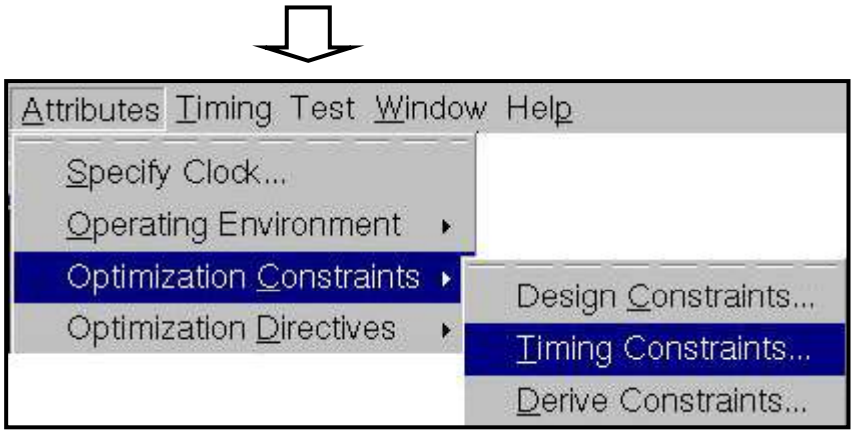
{ Command Line }

```
set_input_transition –max 0.5 [get_clocks clk]
```

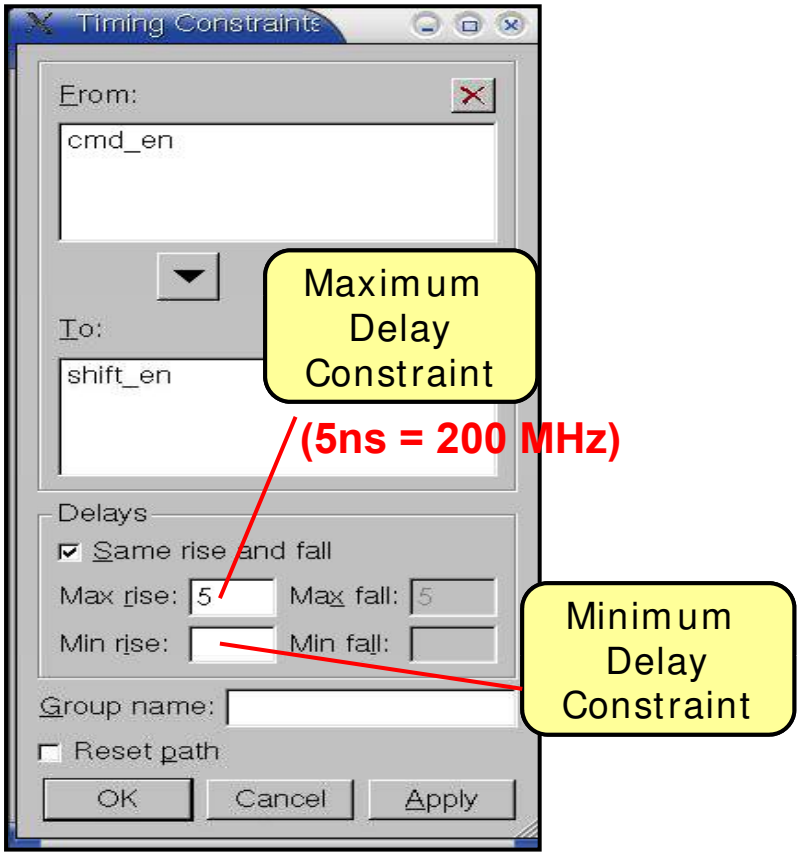
Combination Circuit – Maximum Delay Constraints

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





Ex:



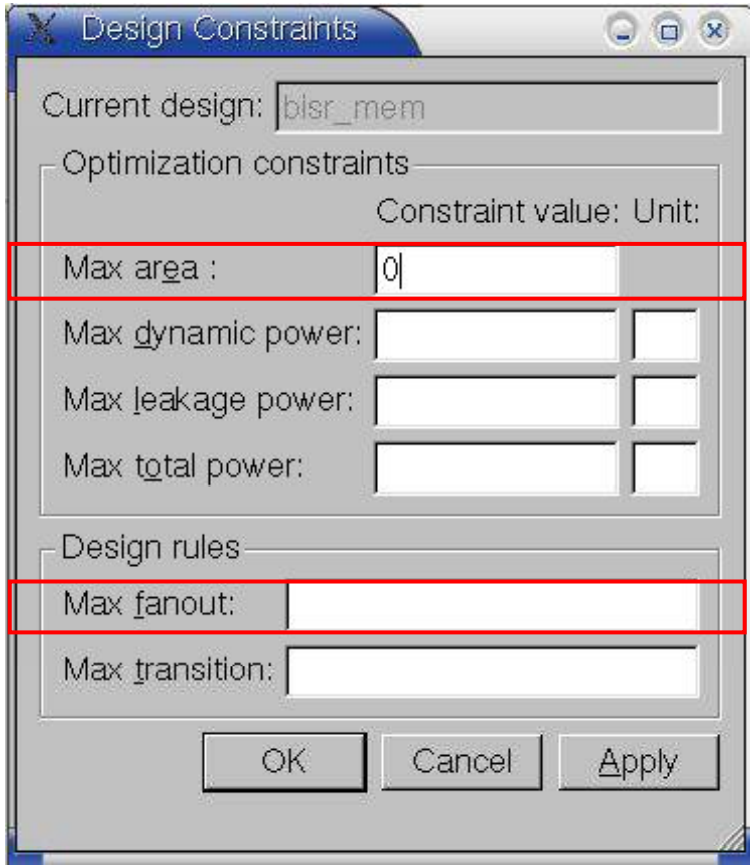
Design Rule Constraints

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

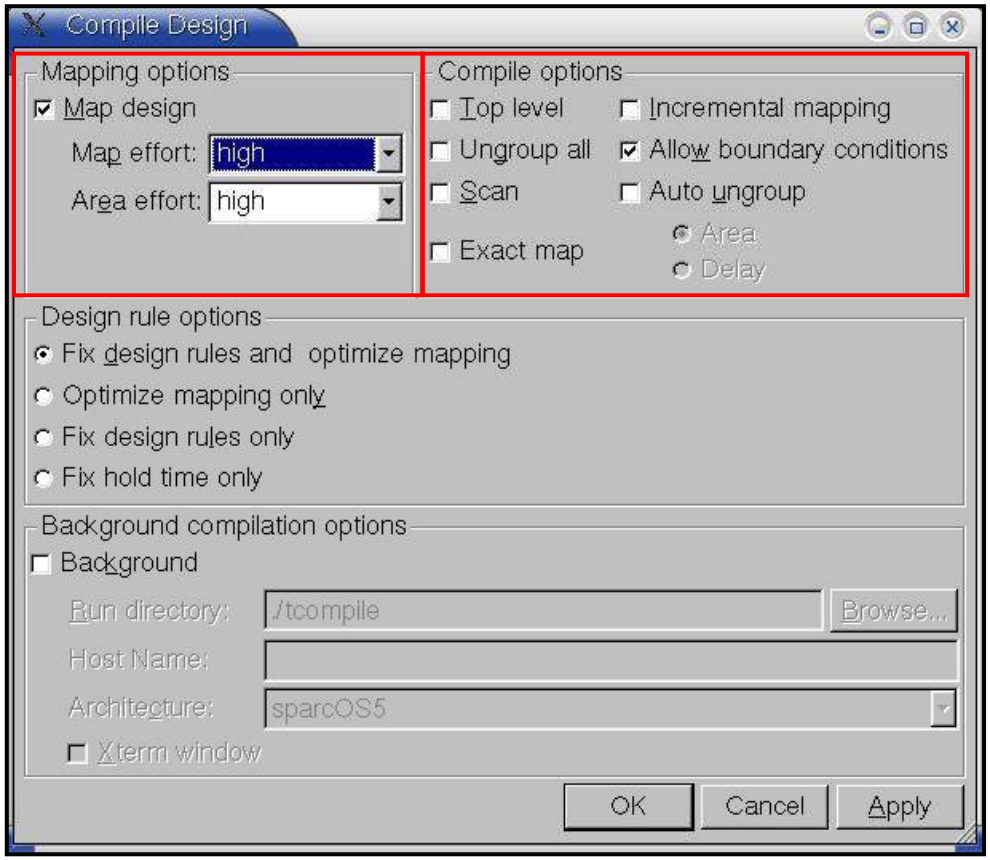


{ Command Line }

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

Compile the Design

Design/Compile Design



{ Command Line }

compile –map_effort high –boundary_optimization

Example for DFT Insertion

DFT Insertion

{ Command Line }

DB Mode

```
####DB mode####
set_dft_configuration -autofix
set_dft_configuration -shadow_wrapper
set_scan_configuration -style multiplexed_flip_flop
set_scan_configuration -clock_mixing no_mix
set_scan_configuration -methodology full_scan
set_scan_signal test_scan_in -port si
set_scan_signal test_scan_out -port so
set_scan_signal test_scan_enable -port se
set_dft_signal test_mode -port scantest
set_test_hold 0 rst
set_test_hold 1 scantest
set_test_hold 1 se
create_test_clock -period 100 -waveform [list 40 60] [find port "clk"]
set_port_configuration -cell RA1SHD256x8 -clock clk
set_port_configuration -cell RA1SHD256x8 -port "Q" -tristate -read {"OEN" 0} -clock clk
set_port_configuration -cell RA1SHD256x8 -port "A" -write {"WEN" 0} -clock clk
set_port_configuration -cell RA1SHD256x8 -port "D" -write {"WEN" 0} -clock clk
set_wrapper_element RA1SHD256x8 -type shadow
set_wrapper_element FJU_MEM -type shadow
set_fix_multiple_port_nets -all -constants -buffer_constants [get_designs *]
insert_dft
```

Example for DFT Insertion (Cont')

{ Command Line }

XG Mode

```
####XG mode####
create_port –dir in scan_in
create_port –dir out scan_out
create_port –dir in scan_en
compile –scan –boundary_optimization
set_scan_configuration –internal_clocks single –chain_count 1 –clock_mixing no_mix
set_dft_signal –view exist –type TestClock –timing {45 55} –port {clk}
set_dft_signal –view exist –type Reset –active 1 –port reset
set_dft_signal –view spec –type ScanEnable –port scan_en –active 1
set_dft_signal –view spec –type ScanDataIn –port scan_in
set_dft_signal –view spec –type ScanDataOut –port scan_out
set_scan_path chain1 –view spec –scan_data_in scan_in –scan_data_out scan_out
create_test_protocol
dft_drc
preview_dft –show scan_clocks
set_false_path –from [get_ports scan_en]
insert_dft
```

Compile After DFT

Compile After DFT

{ Command Line }

compile -scan
check_scan
report_test -scan_path
estimate_test_coverage

- The fault coverage will be shown as below:

```
    Uncollapsed Stuck Fault Summary Report
-----------------------------------------------
fault class                       code   #faults
------------------------------  ----  ---------
Detected                          DT      24997
Possibly detected                 PT        171
Undetectable                      UD        475
ATPG untestable                   AU       1822
Not detected                      ND        165
-----------------------------------------------
total faults                              27630
test coverage                             92.37%
fault coverage                            90.78%
-----------------------------------------------
           Pattern Summary Report
-----------------------------------------------
#internal patterns                          227
    #basic_scan patterns                    227
-----------------------------------------------
```

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.1 – Hierarchy

```
****************************************
Report : hierarchy
Design : bisr_mem
Version: X-2005.09-SP4
Date   : Fri Jul 27 14:53:58 2007
****************************************

Information: This design contains unmapped logic. (RPT-7)

bisr_mem
   GTECH_OR2                          gtech
   bisr
      GTECH_AND2                      gtech
      GTECH_BUF                       gtech
      GTECH_NOT                       gtech
      bira_top
         GTECH_AND2                   gtech
         GTECH_NOT                    gtech
         GTECH_OR2                    gtech
         bitmap
            GTECH_AND2                gtech
            GTECH_AND3                gtech
            GTECH_AND4                gtech
            GTECH_BUF                 gtech
            GTECH_NOT                 gtech
            GTECH_OR2                 gtech
```

Report Area

- ***Design/Report Area***

Ex:

```
****************************************
Report : area
Design : bisr_mem
Version: X-2005.09-SP4
Date   : Fri Jul 27 15:31:16 2007
****************************************

Library(s) Used:
    gtech (File: /usr/cad/synopsys/synthesis/cur/libraries/syn/gtech.db)
    USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/MEM/DB/memory_8k_32_fast@-40C_syn.db)
    USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/MEM/DB/sc_memory_fast@-40C_syn.db)
    USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/MEM/DB/sr_memory_fast@-40C_syn.db)

Number of ports:              105
Number of nets:               248
Number of cells:                3
Number of references:           3

Combinational area:          0.000000
Noncombinational area:    3271507.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:          3271507.000000
Total area:                 undefined

Information: This design contains unmapped logic. (RPT-7)
Information: This design contains black box (unknown) components. (RPT-8)

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

(0.18um Cell-Library: 1 gate ≈ 10 um^2)
(0.13um Cell-Library: 1 gate ≈ 5 um^2)

(um^2)

Design View

- ***List/Design View***

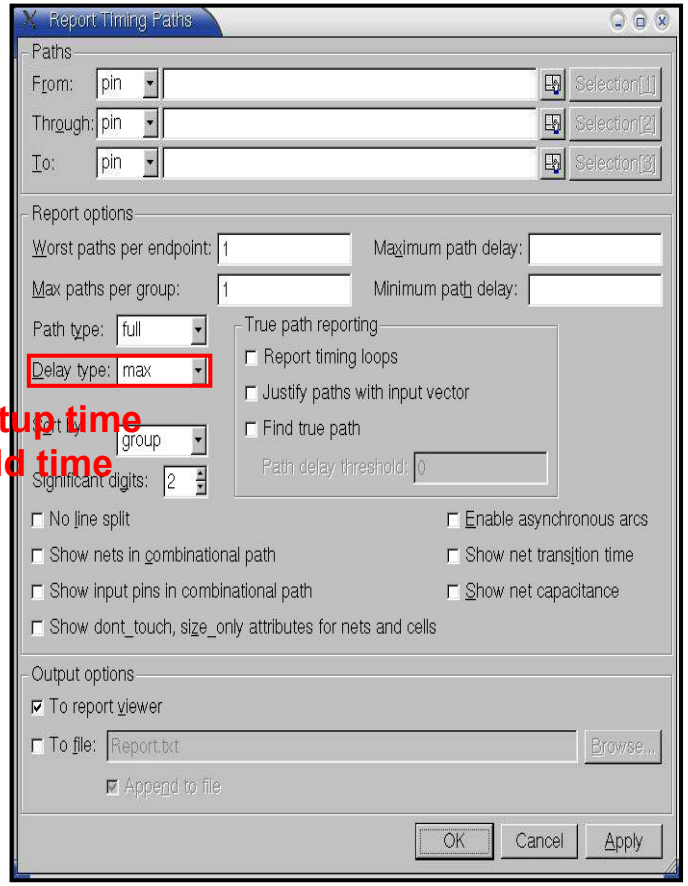
All the block area are listed !!

Ex:

array_or	851.558	undefined	undefined	undefined
bitmap	15501	undefined	undefined	undefined
bisr_mem	3.34175e+06	undefined	undefined	undefined
memory_8k...	3.29505e+06	undefined	undefined	undefined
aru	7717.25	undefined	undefined	undefined
remapping_...	4440.74	undefined	undefined	undefined
address_de...	8276.08	undefined	undefined	undefined
finj	1293.97	undefined	undefined	undefined
bisr	46686	undefined	undefined	undefined
bira_top	26953.8	undefined	undefined	undefined
remap	8016.62	undefined	undefined	undefined
remap_DW...	192.931	undefined	true	false
multi_bit	1593.35	undefined	undefined	undefined
bitmap_DW...	83.16	undefined	undefined	undefined
fsm	1816.21	undefined	undefined	undefined
bist	17413.7	undefined	undefined	undefined
tpg	15151.8	undefined	undefined	undefined
ADDR	2421.62	undefined	undefined	undefined
ADDR_DW...	435.758	undefined	true	false
ADDR_DW...	472.349	undefined	true	false
CMP	9433.67	undefined	undefined	undefined
DATA	219.542	undefined	undefined	undefined
DECO	804.989	undefined	undefined	undefined
ROM	1147.61	undefined	undefined	undefined
ctr	2261.95	undefined	undefined	undefined

Report Timing

- ***Timing/Report Timing***



max: setup time
min: hold time

Ex:

setup time

Critical Path

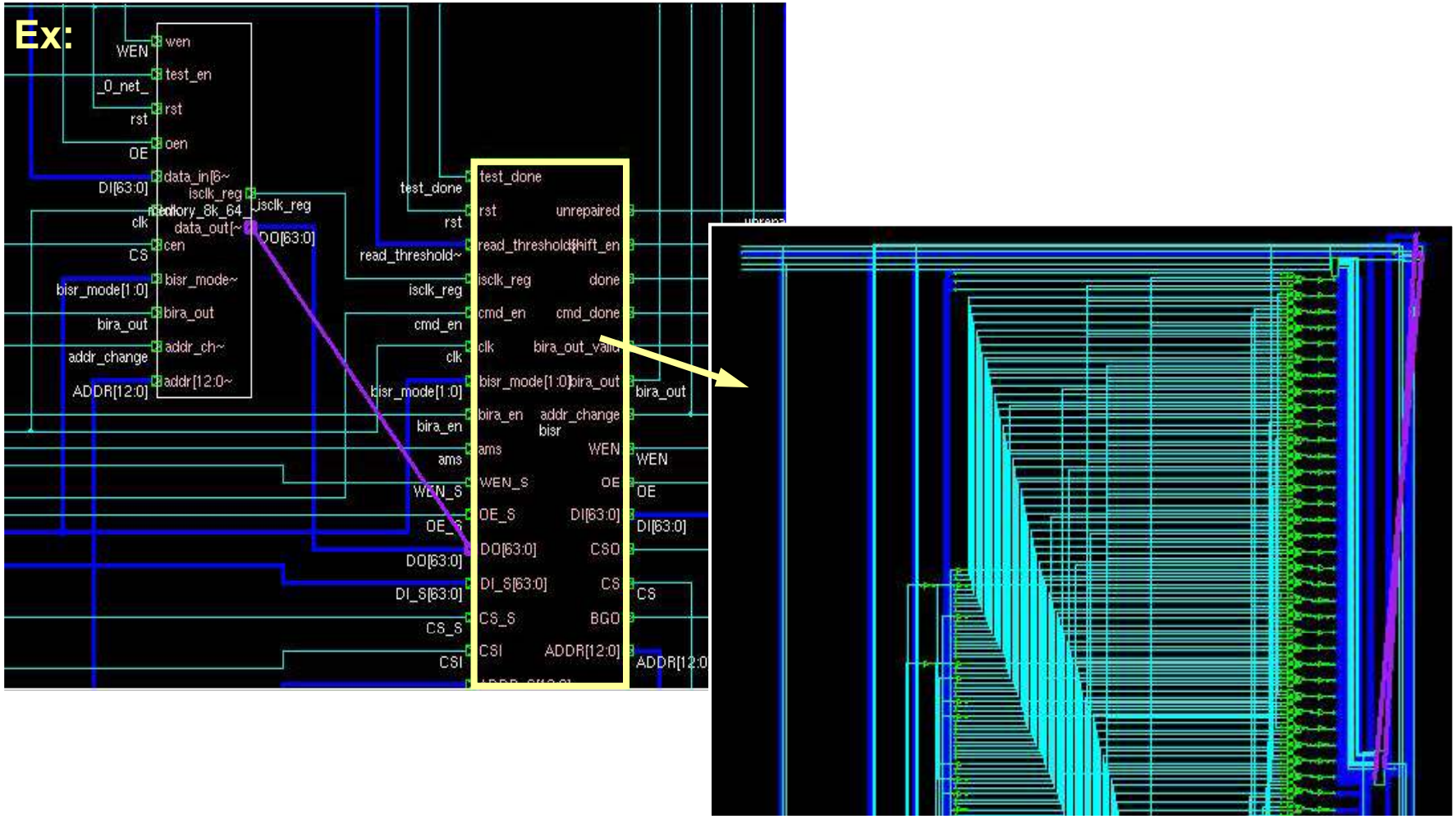
```
Path Type: max

Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock clk (rise edge)                                    5.00       5.00
clock network delay (ideal)                              0.00       5.00
memory_8k_64_2r_2c/sc_memory/CLK (sc_memory)             0.00       5.00 r
memory_8k_64_2r_2c/sc_memory/Q[0] (sc_memory)            0.71       5.71 r
memory_8k_64_2r_2c/aru/Q_sc[0] (aru)                     0.00       5.71 r
memory_8k_64_2r_2c/aru/U825/Y (AOI22X1)                  0.06       5.78 f
memory_8k_64_2r_2c/aru/U447/Y (INVX1)                    0.70       6.47 r
memory_8k_64_2r_2c/aru/U643/Y (AOI22X1)                  0.06       6.53 f
memory_8k_64_2r_2c/aru/U642/Y (OAI2BB1X1)                0.12       6.64 r
memory_8k_64_2r_2c/aru/data_out[47] (aru)                0.00       6.64 r
memory_8k_64_2r_2c/data_out[47] (memory_8k_64_2r_2c)
                                                         0.00       6.64 r
bisr/DO[47] (bisr)                                       0.00       6.64 r
bisr/bist/DO[47] (bist)                                  0.00       6.64 r
bisr/bist/m1/DO[47] (tpg)                                0.00       6.64 r
bisr/bist/m1/CMP/DO[47] (CMP)                            0.00       6.64 r
bisr/bist/m1/CMP/U207/Y (XNOR2X1)                        0.20       6.85 r
bisr/bist/m1/CMP/U185/Y (NAND4X1)                        0.07       6.92 f
bisr/bist/m1/CMP/U204/Y (NOR4X1)                         0.20       7.11 r
bisr/bist/m1/CMP/U202/Y (NAND4X1)                        0.06       7.17 f
bisr/bist/m1/CMP/U171/Y (AND2X2)                         0.13       7.30 f
bisr/bist/m1/CMP/U174/Y (INVX1)                          1.43       8.74 r
bisr/bist/m1/CMP/U170/Y (NAND2X2)                        0.64       9.38 f
bisr/bist/m1/CMP/U334/Y (OAI22X1)                        0.32       9.70 r
bisr/bist/m1/CMP/syndrome_reg[0]/D (DFFNRX1)             0.00       9.70 r
data arrival time                                                   9.70

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
-----------------------------------------------------------------------------
data required time                                                  9.91
data arrival time                                                  -9.70
-----------------------------------------------------------------------------
slack (MET)                                                         0.21
```

Slack = Data Require Time – Data Arrival Time

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

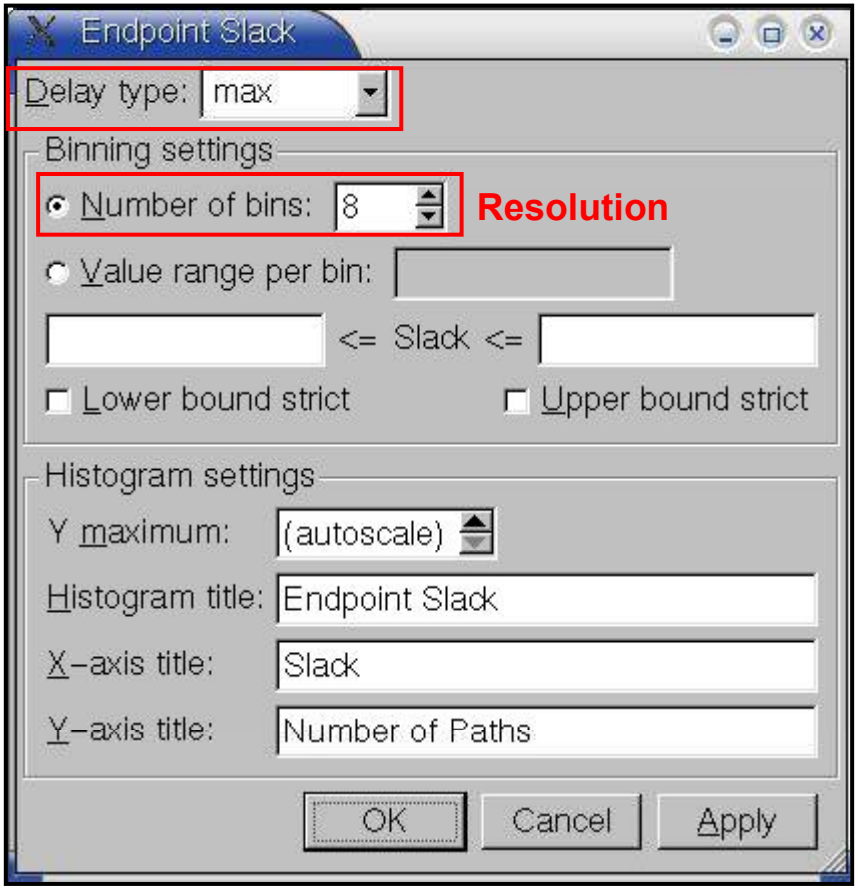
Critical Path Highlighting

- ***View/Highlight/Critical Path***

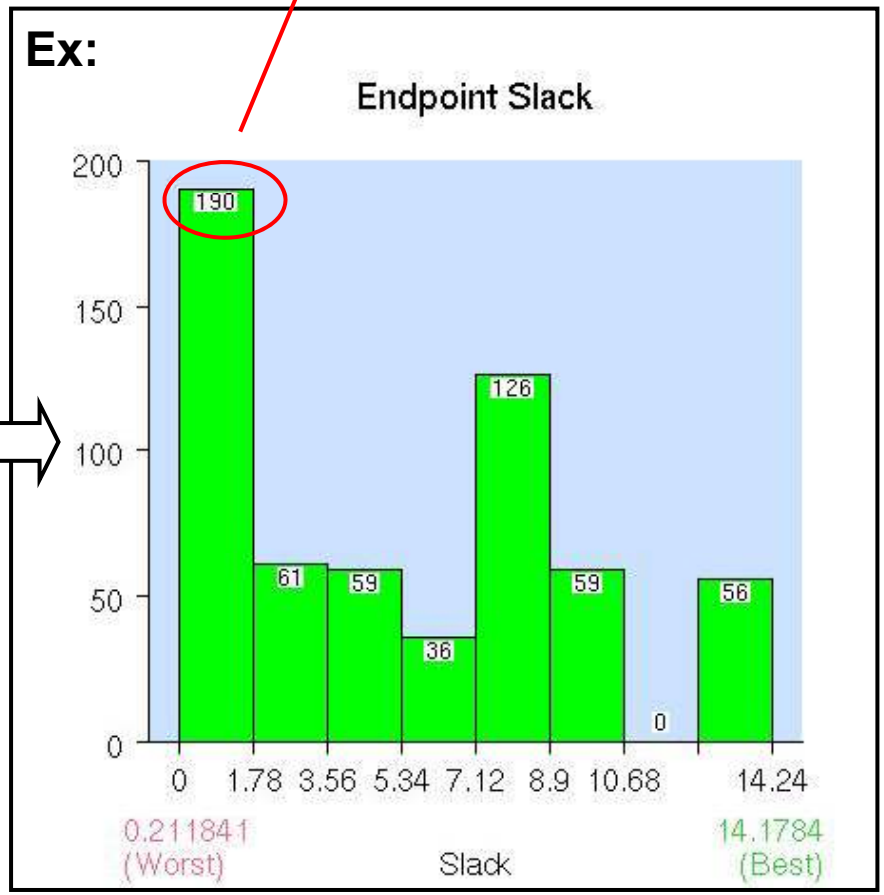


Timing Slack Histogram

- ***Timing/Endpoint Slack***



Totally 190 paths are in the slack range between 0 to 1.78



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
  - ***Setup/Execute Script***
  - ***Or use "source your_ script.dc" in dc_shell command line***

Ex:

```
####################Parameters#####################################
#set cycle 10
#set t_in 5
#set t_out 0.5
#set in_pad_delay 0.34
#set out_pad_delay 0.96
#################Set Current Design##############################
current_design bisr_mem
uniquify
#################Operating Evironment############################
set_operating_conditions -max "slow" -max_library "slow" -min "fast"\
-min_library "fast"
set_wire_load_model  -name "tsmc18_wl10" -library "slow"
set_wire_load_mode "top"

#################Clock Settings##################################
create_clock -period 10 [get_ports clk]
set_dont_touch_network [get_clocks clk]
set_fix_hold [get_clocks clk]
set_clock_uncertainty  0.1 [get_clocks clk];
set_clock_latency  1 [get_clocks clk];
set_ideal_network [get_ports clk]
set_input_transition -max 0.5 [all_inputs];

#################In/Out Delay, Driving, and Loading Settings#######
set_input_delay  [expr 0.34] -clock clk [all_inputs]
set_input_delay  [expr 0.34] -clock clk [get_ports clk]
#set_input_delay  [expr 1+0.34] -clock clk [get_ports clk]
set_output_delay [expr 0.5+1.5] -clock clk [all_outputs]
set_load [expr 0.06132] [all_outputs]
set_drive [expr 0.288001] [all_inputs]

#################Design Rule Constraint##########################
set_max_fanout 50 [get_designs bisr_mem]
#set_max_area 0

#################Avoid Multi-Instance Wrning#####################
```

Gate-Level Simulation

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

```
`include "tsmc18.v"
`include "bisr_mem.vg"
`include "memory_8k_32.v"
`include "sc_memory.v"
`include "sr_memory.v"
```

Standard Cell Library

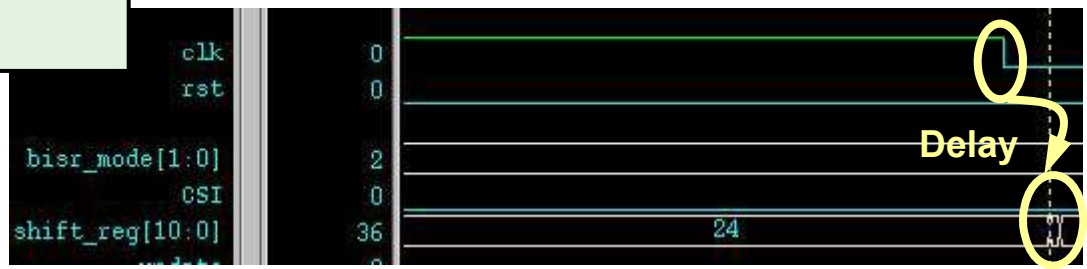
Gate- Level Netlist

- Add the following Synopsys directives to the test bench

```
initial
begin
$fsdbDumpfile("bisr_mem.fsdb");
$fsdbDumpfile("bisr.fsdb");
$fsdbDumpvars;
end

initial
begin
$sdf_annotate("bisr_mem.sdf",bisr_mem);
end

initial begin
  #0                clk = 1'b0;
  forever #50.0000     clk = ~clk;
end
```

***.sdf File**

Instance Name



Simulation-Based Power Estimation Using PrimePower

<.synopsys_pp.setup> File

- Modify the <.synopsys_dc.setup> file, and then save as <.synopsys_pp.setup >

Add the path where you put this file

```
#######0.18um
set search_path         "/usr1/teacher/jfli/cell_lib/CBDK018_TSMC_Artisan/CIC/SynopsysDC $search_path"
set search_path         "/usr/cad/synopsys/synthesis/cur/libraries/syn  $search_path"
set search_path         "/usr1/teacher/jfli/cell_lib/CBDK018_TSMC_Artisan/orig_lib/aci/sc/symbols/synop
set search_path         "/usr4/grad92/zwtseng/HOY/rom_base_bist/8_bit_bist/bist_256x8_with_ECC/0331_bis
set search_path         "/usr4/grad92/zwtseng/HOY/rom_base_bist/BISD_DEMO_0628 $search_path"

######with MEM####
set link_library        "HTW_BISD.db RA1SHD256x8_fast@-40C_syn.db RA1SHD256x8_fast@0C_syn.db RA1SHD256x
set target_library      "RA1SHD256x8_fast@-40C_syn.db RA1SHD256x8_fast@0C_syn.db RA1SHD256x8_typical_sy

######without MEM####
set symbol_library      "tsmc18.sdb"

set hdlin_translate_off_skip_text "TRUE"
set edifout_netlist_only "TRUE"
set verilogout_no_tri true
set plot_command {lpr -Plp3
```

Delete the directives

(.synopsys_pp.setup)

VCD File Generation

- Add the following Synopsys directives to the test bench

```
initial
begin
  $dumpfile("bisr_mem.vcd");
  $dumpvars;
end
```

- Then, run the Verilog simulation

Instructions

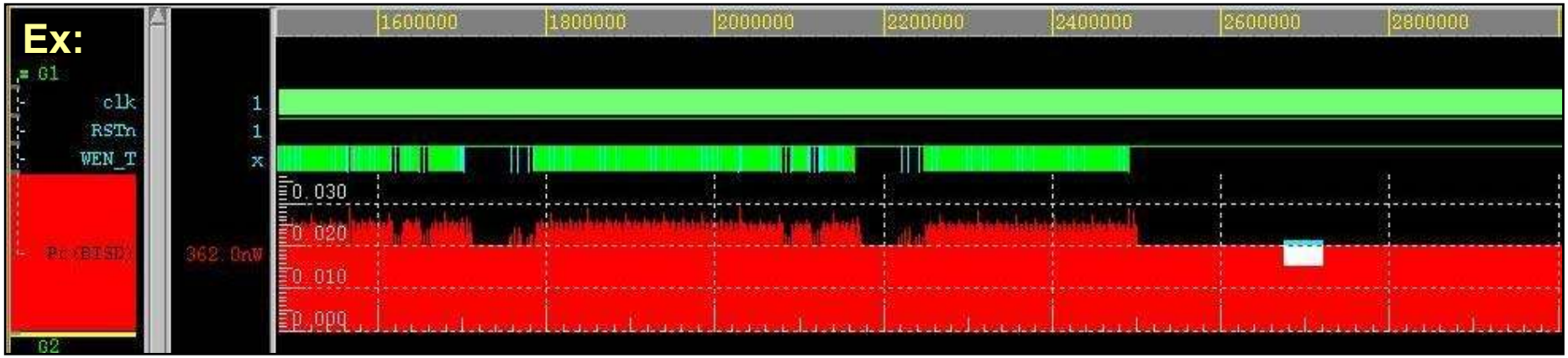
- *linux %> source /APP/cshbank/primepower.csh*
- *linux %> pp_shell*
- *pp_shell> read_verilog CORE.vg*
- *pp_shell> current_design CORE*
  TOP Module Name
- *pp_shell> read_vcd –strip_path test/CORE CORE.vcd*
  Module Name of Test Bench, Instance Name
- *pp_shell> set_waveform_options –interval 1 –format fsdb –file pwr1*
  Resolution, Waveform Format, Waveform File Name
- *pp_shell> calculate_power –waveform –statistics*
- *pp_shell> report_power –file pwr_rpt –hier 2*
  Power Report File Name, Estimation Hierarchy

Power Report

Ex:

```
Definitions:
    Total Power     = Dynamic + Leakage
    Dynamic Power   = Switching + Internal
    Switching Power = load capacitance charge or discharge power
    Internal Power  = power dissipated within a cell
    X-tran Power    = component of dynamic power-dissipated into x-transitions
    Glitch Power    = component of dynamic power-dissipated into detectable
                      glitches at the nets
    Leakage Power   = reverse-biased junction leakage + subthreshold leakage


-----------------------------------------------------------------------------
Level          Instance_Name (Cell_Name)   #_of_States
-----------------------------------------------------------------------------
 Total      Dynamic    Leakage    Switching  Internal   X-tran     Glitch
 Power      Power      Power      Power      Power      Power      Power
 in Watt    in Watt    in Watt    in Watt    in Watt    in Watt    in Watt
            (% of Tot) (% of Tot) (% of Dyn) (% of Dyn) (% of Dyn) (% of Dyn)
-----------------------------------------------------------------------------

----------------------------- Instances -------------------------------------
*--------- 0  pp_root (.) 71517932
 2.596e-04  2.592e-04  4.558e-07  8.722e-06  2.504e-04  1.561e-08  3.207e-07
            ( 99.82%)  (  0.18%)  (  3.37%)  ( 96.63%)  (  0.01%)  (  0.12%)

-*-------- 1  HTW_BISD (HTW_BISD) 71517932
 2.596e-04  2.592e-04  4.558e-07  8.722e-06  2.504e-04  1.561e-08  3.207e-07
            ( 99.82%)  (  0.18%)  (  3.37%)  ( 96.63%)  (  0.01%)  (  0.12%)

--*------- 2  HTW_BISD/BISD (BISD) 53627079
 1.948e-04  1.944e-04  3.621e-07  7.789e-06  1.867e-04  1.485e-08  2.986e-07
            ( 99.81%)  (  0.19%)  (  4.01%)  ( 95.99%)  (  0.01%)  (  0.15%)

---*------ 3  HTW_BISD/BISD/BISD_top (BISD_top) 23228922
 8.787e-05  8.771e-05  1.603e-07  2.217e-06  8.549e-05  1.460e-08  5.682e-08
            ( 99.82%)  (  0.18%)  (  2.53%)  ( 97.47%)  (  0.02%)  (  0.06%)

----*----- 4  HTW_BISD/BISD/BISD_top/m1 (bit_map) 13505944
 5.174e-05  5.164e-05  1.040e-07  7.136e-07  5.092e-05  1.348e-08  9.382e-10
            ( 99.80%)  (  0.20%)  (  1.38%)  ( 98.62%)  (  0.03%)  (  0.00%)
```

Power Waveforms



- For example, the figure shows the power waveforms of the BIST execution. We can observe that the power varies drastically while the memory is accessed

Artisan Memory Compiler

Overview

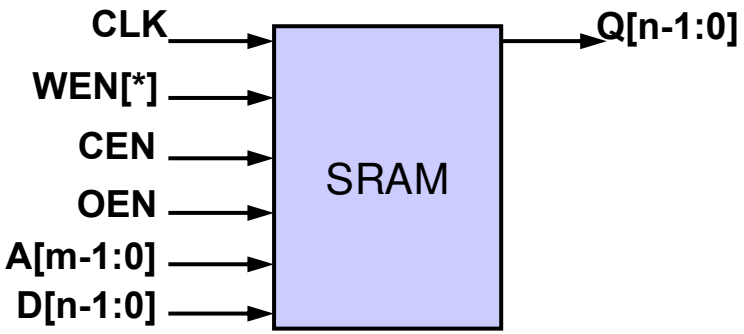
- Artisan SRAM Types:

Generator	Product Name	Executable
High-Speed/Density Single-Port SRAM	SRAM-SP	ra1sh
High-Speed/Density Dual-Port SRAM	SRAM-DP	ra2sh
High-Density Single-Port SRAM	SRAM-SP-HD	ra1shd
High-Density Dual-Port SRAM	SRAM-DP-HD	ra2shd
Low-Power Single-Port SRAM	SRAM-SP-LP	ra1shl

[REF: Artisan User Manual]

- Only ra1shd and ra2sh are supported in school
- Generated files:
  - Memory Spec. (i.e. used for layout-replacement procedure in CIC flow)
  - Memory Data Sheet
  - Simulation models: Verilog Model & VHDL Model
  - Memory Libraries for P&R: Synopsys Model & VCLEF Footprint
  - Timing Files: TLF Model & PrimeTime Model

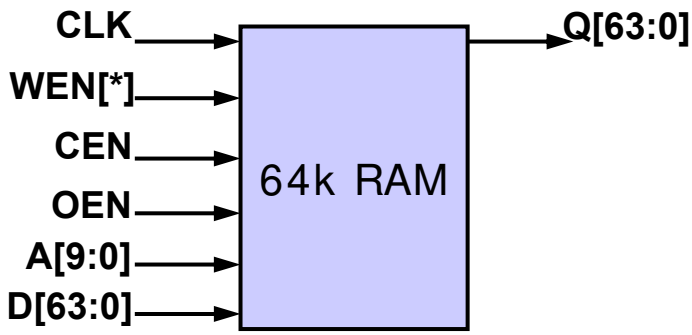
Pin Descriptions for Single-Port SRAM

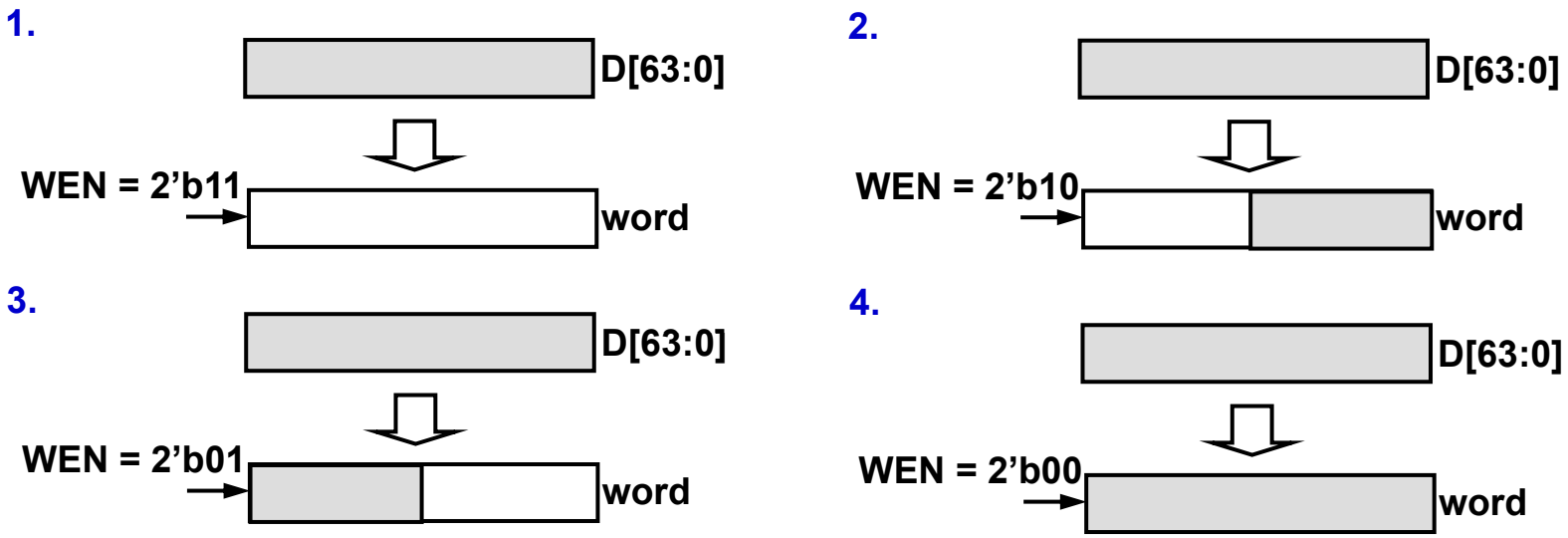


Name	Type	Description
Basic Pins		
CLK	Input	Clock
WEN[*]	Input	Write enable, active low. *If word-write mask is enabled, this becomes a bus
CEN	Input	Chip enable, active low
OEN	Input	Tri-state output enable
A[m-1:0]	Input	Address (A[0]=LSB)
D[n-1:0]	Input	Data inputs (D[0]=LSB)
Q[n-1:0]	Output	Data outputs (Q[0]=LSB)

Example for Word-Write Mask

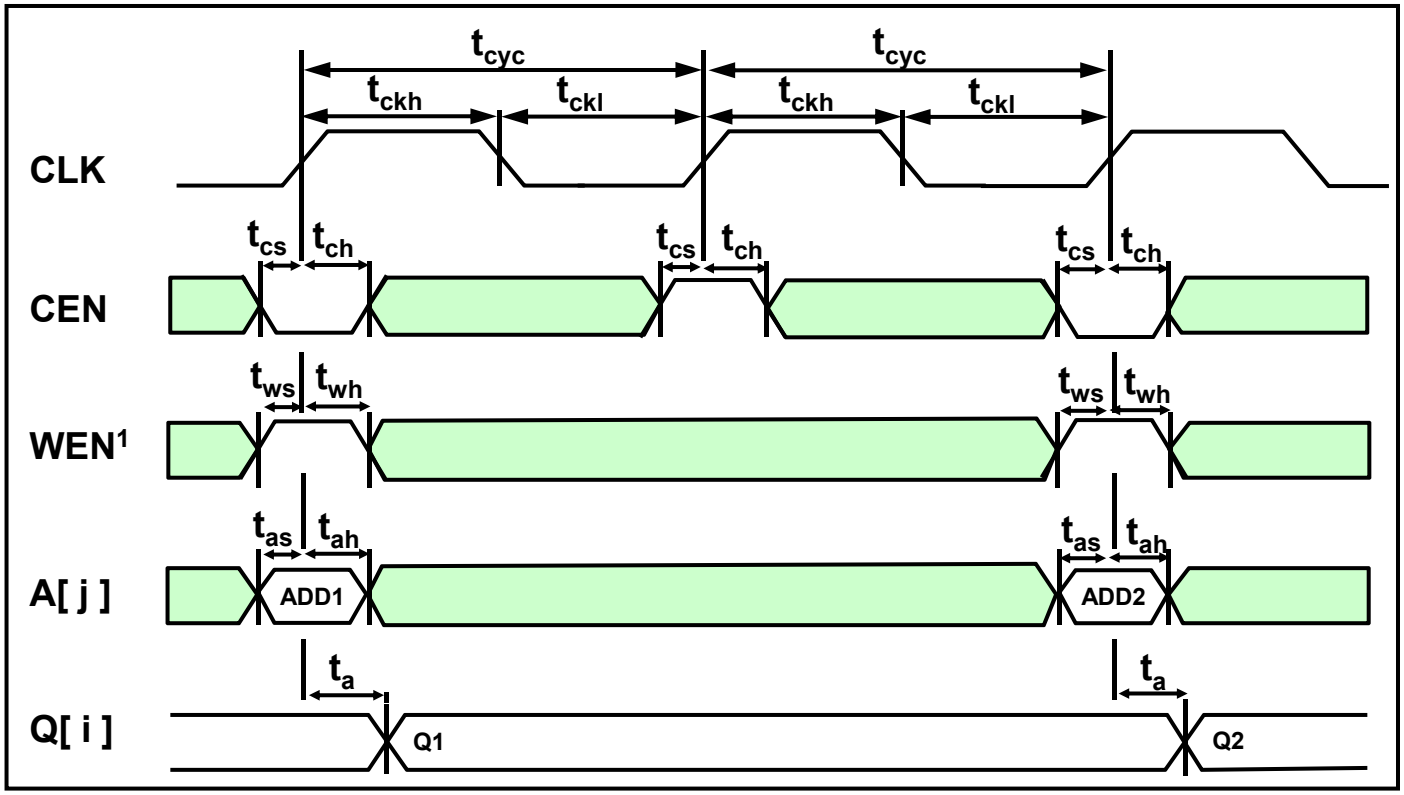
- Word Width: 64 bits
  - Word Partition Size: 32 bits
  - Mask Width = WEN Width = 2
  - WEN[1:0]
    - 11: No write
    - 10: Write to LSB part
    - 01: Write to MSB part
    - 00: Write to the whole word





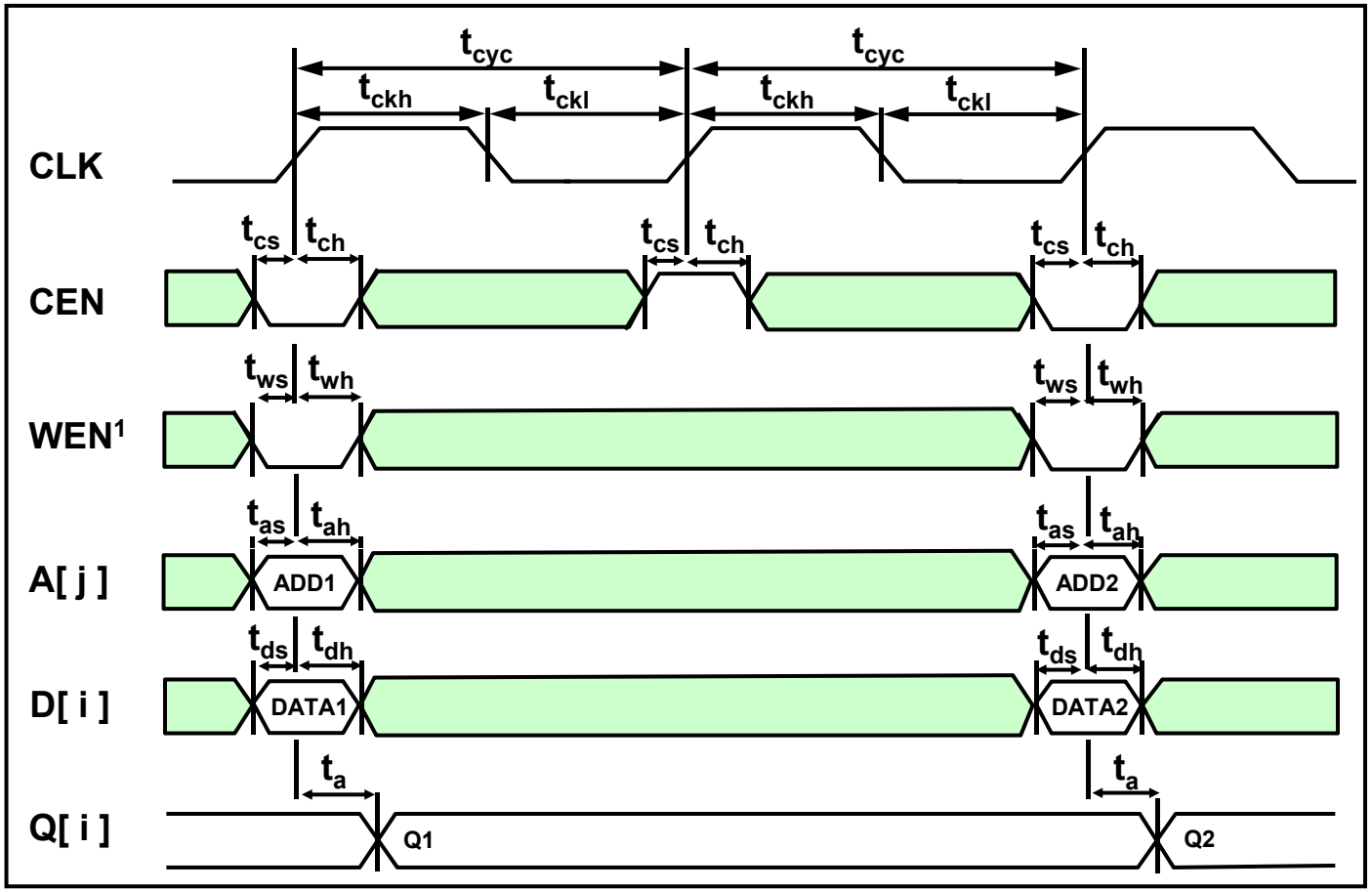
Waveforms for Single-Port SRAM

- Read Cycle



Waveforms for Single-Port SRAM (Cont')

- Write Cycle

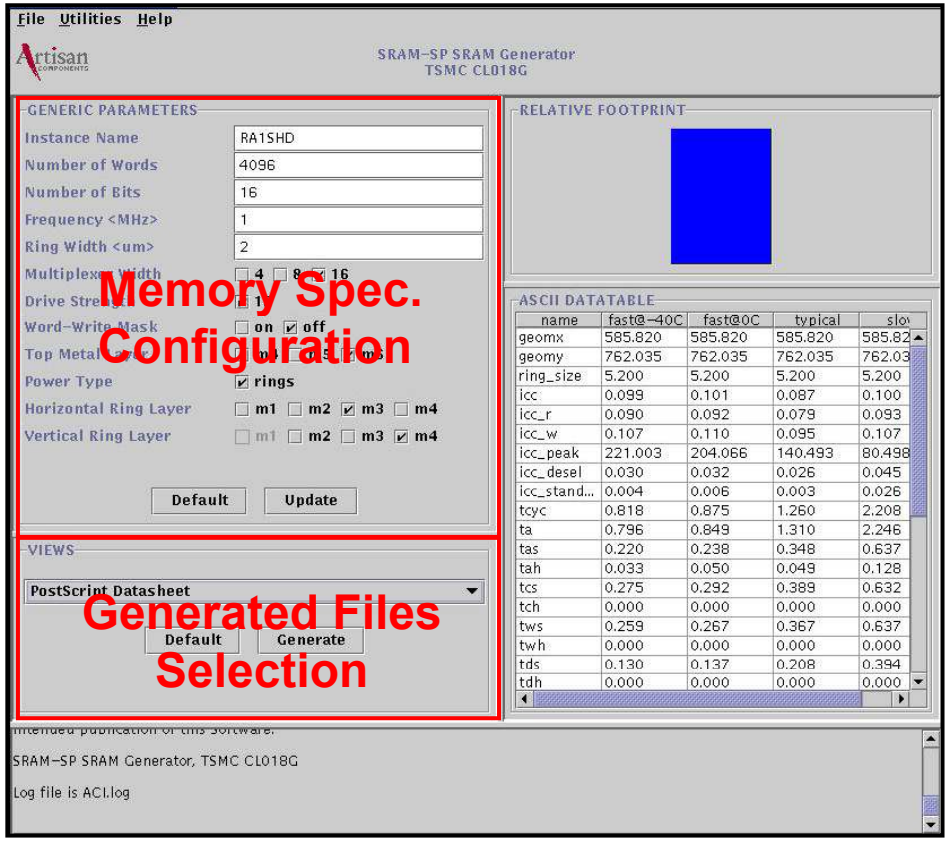


Getting Started

- *linux %> ssh -l "user name" cae18.ee.ncu.edu.tw* ← **Connect to Unix**

(1-port RAM) *unix%> ~jfli/cell_lib/CBDK018_TSMC_Artisan/CIC/Memory/ra1shd/bin/ra1shd*

(2-port RAM) *unix%> ~jfli/cell_lib/CBDK018_TSMC_Artisan/CIC/Memory/ra2sh/bin/ra2sh*

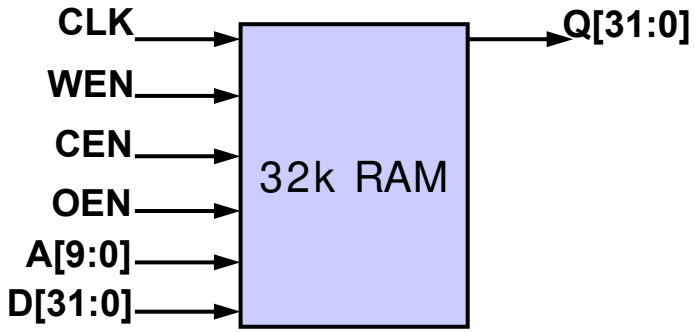


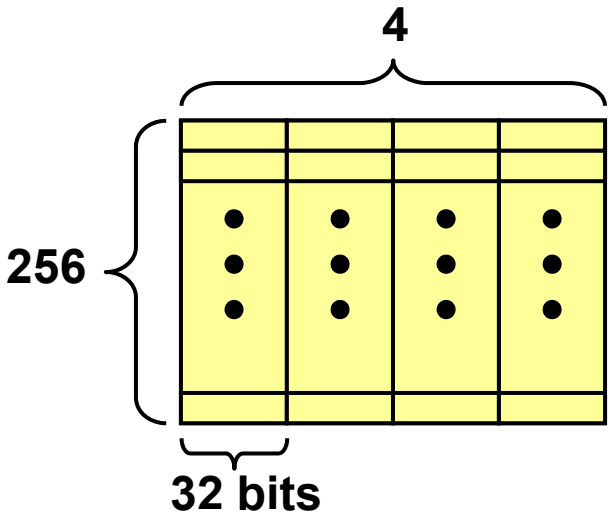
(GUI view of the Artisan)

Memory Spec Configuration (Example 1)

- Instance Name: mem_32k
- Number of Words: 1024
- Number of Bits: 32
- Frequency <MHz>: 100
- Ring Width <um>: 2
- Multiplexer Width
  - ☑ 4 ☐ 8 ☐ 16
- Drive Strength
- Word-Write Mask
  - ☐ on ☑ off
- Top Metal Layer
  - ☐ m4 ☑ m5 ☐ m6
- Power Type
- Horizontal Ring Layer
  - ☐ m1 ☐ m2 ☑ m3 ☐ m4
- Vertical Ring Layer
  - ☐ m2 ☐ m3 ☑ m4

Ex: 32k RAM (no mask write)

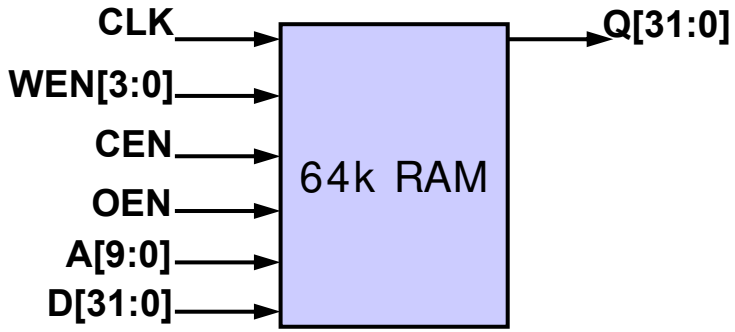


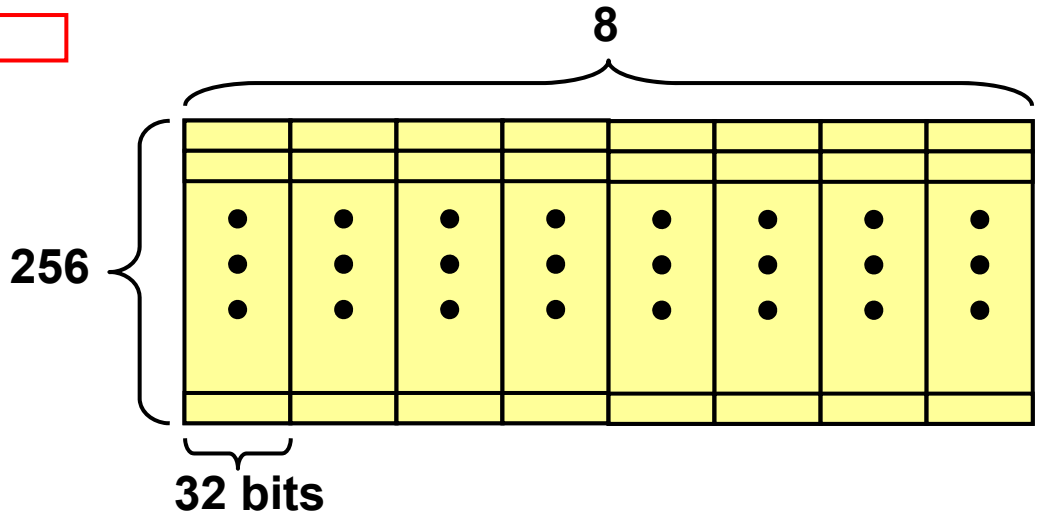


Memory Spec Configuration (Example 2)

- Instance Name: mem_64k
- Number of Words: 2048
- Number of Bits: 32
- Frequency <MHz>: 100
- Ring Width <um>: 2
- Multiplexer Width
  - ☐4 ☑8 ☐16
- Drive Strength
- Word-Write Mask
  - ☑on ☐off
    - Word Partition Size: 8
- Top Metal Layer
  - ☐m4 ☑m5 ☐m6
- Power Type
- Horizontal Ring Layer
  - ☐m1 ☐m2 ☑m3 ☐m4
- Vertical Ring Layer
  - ☐m2 ☐m3 ☑m4

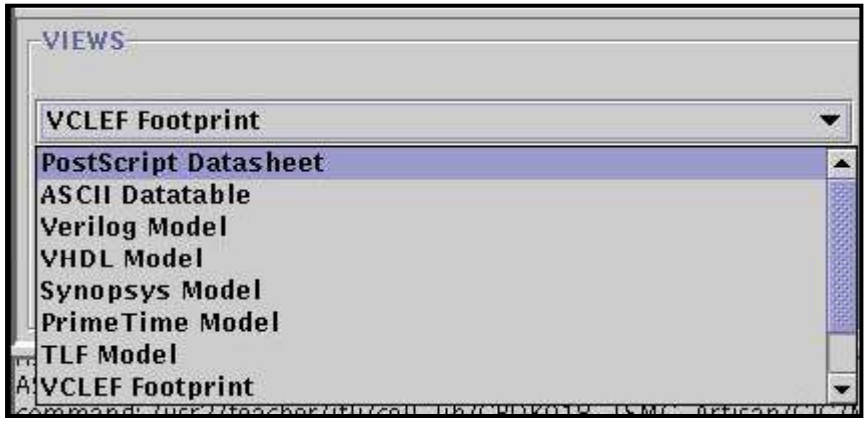
Ex: 64k RAM (with mask write)





File Generation (Method 1)

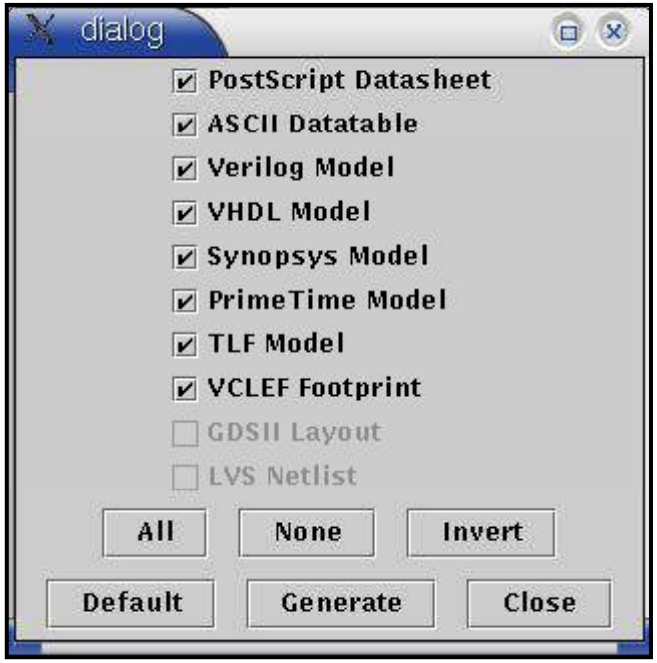
- Pop-up window
  - PostScript Datasheet (.ps)
    - Convert to PDF file: *ps2pdf *.ps*
  - ASCII Datatable (.dat)
  - Verilog Model (.v)
  - VHDL Model (.vhd)
  - Synopsys Model (.lib)
    - The default library name is “USERLIB”
  - PrimeTime Model
  - TLF Model
  - VCLEF Footprint (.vclef)



(File Selection)

File Generation (Method 2)

- From the menu



- Spec. Generation
  - The memory spec. file will be used for the Layout Replacement procedure in the CIC server

LAB