Graduate Institute of Electronics Engineering, NTU

Digital System Design

Synopsys Synthesis Overview

Chihhao Chao

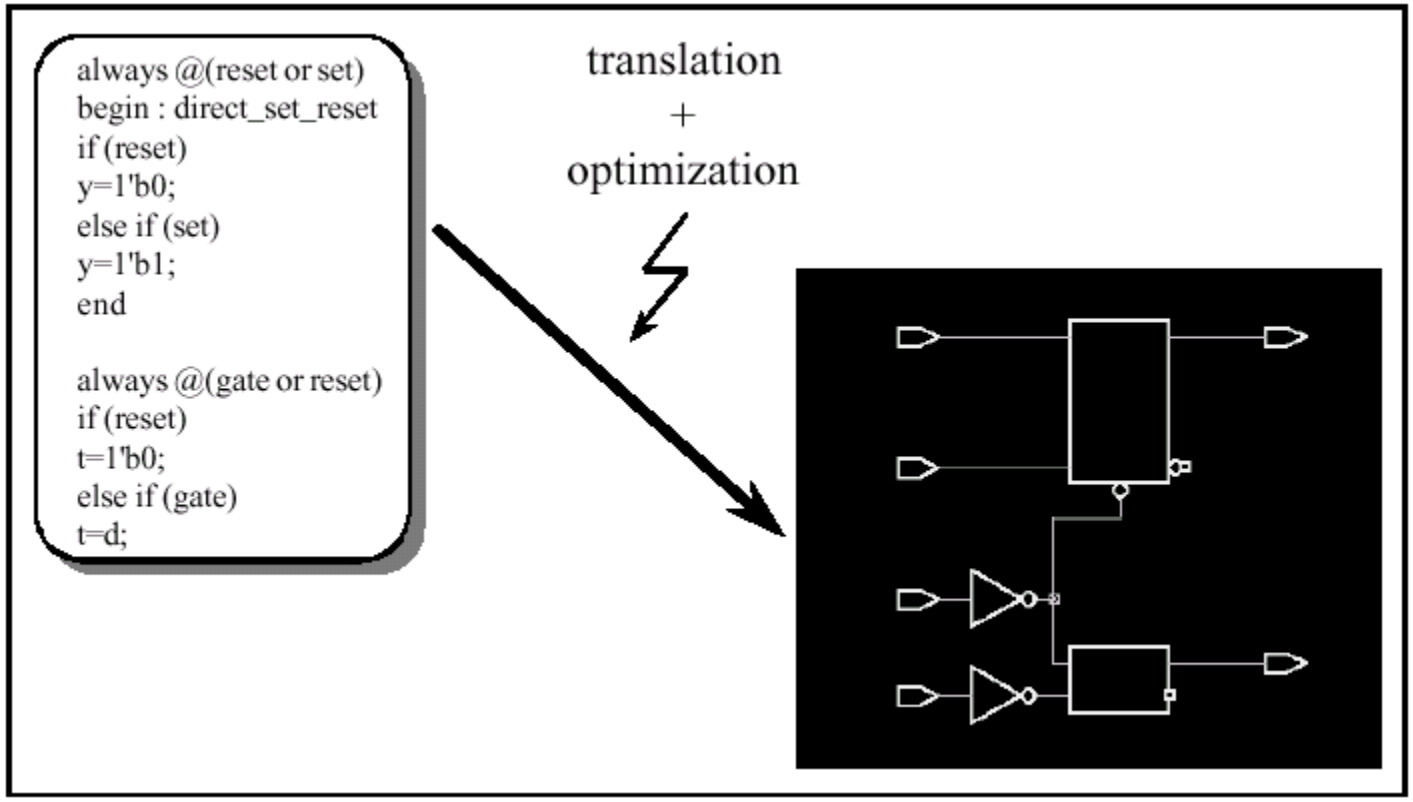
2009.4.1

Outline

- Introduction
- Synopsys Graphical Environment
- Setting Design Environment
- Setting Design Constraints
- Synthesis Report and Analysis

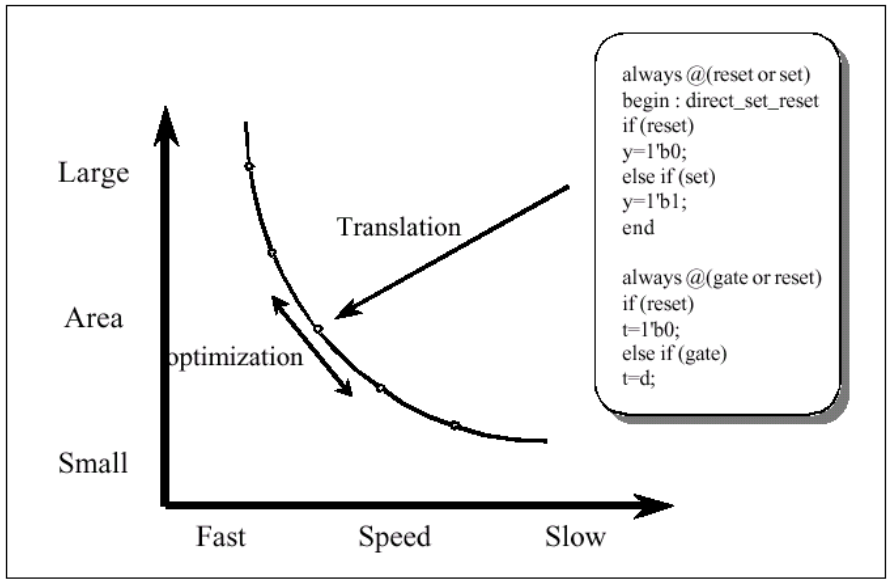
What is Synthesis

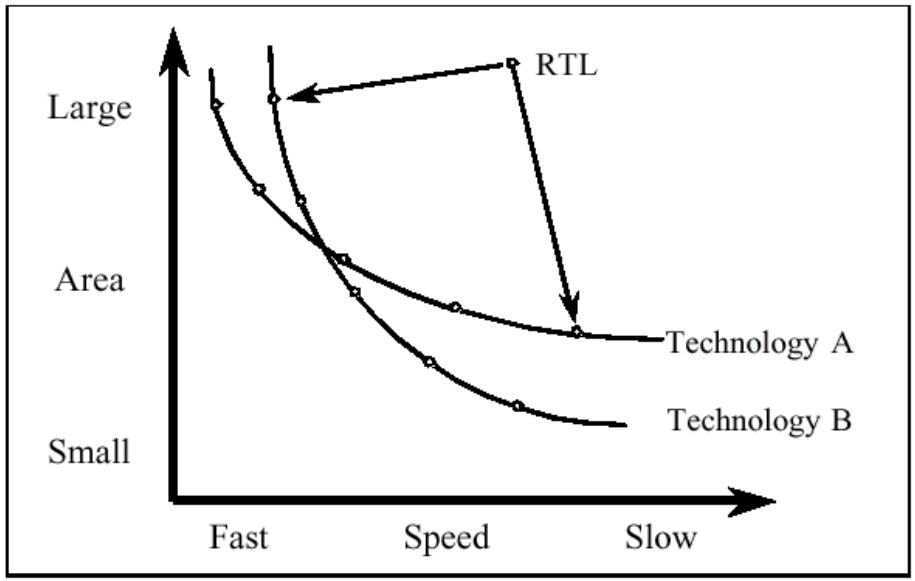
- Synthesis = translation + optimization



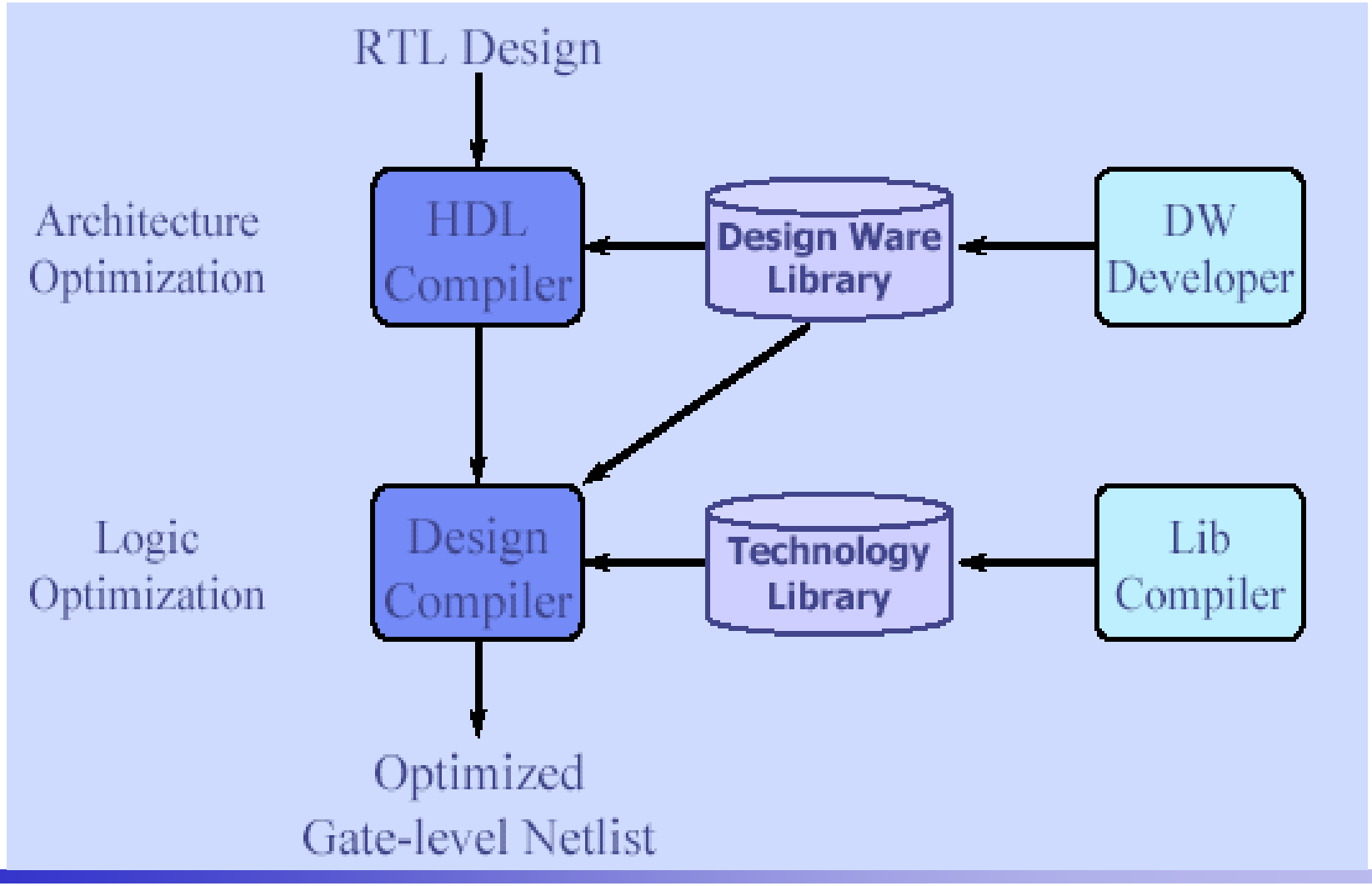
Trade-off between Speed and Area

- Synthesis is Constraint Driven
- Technology Independent





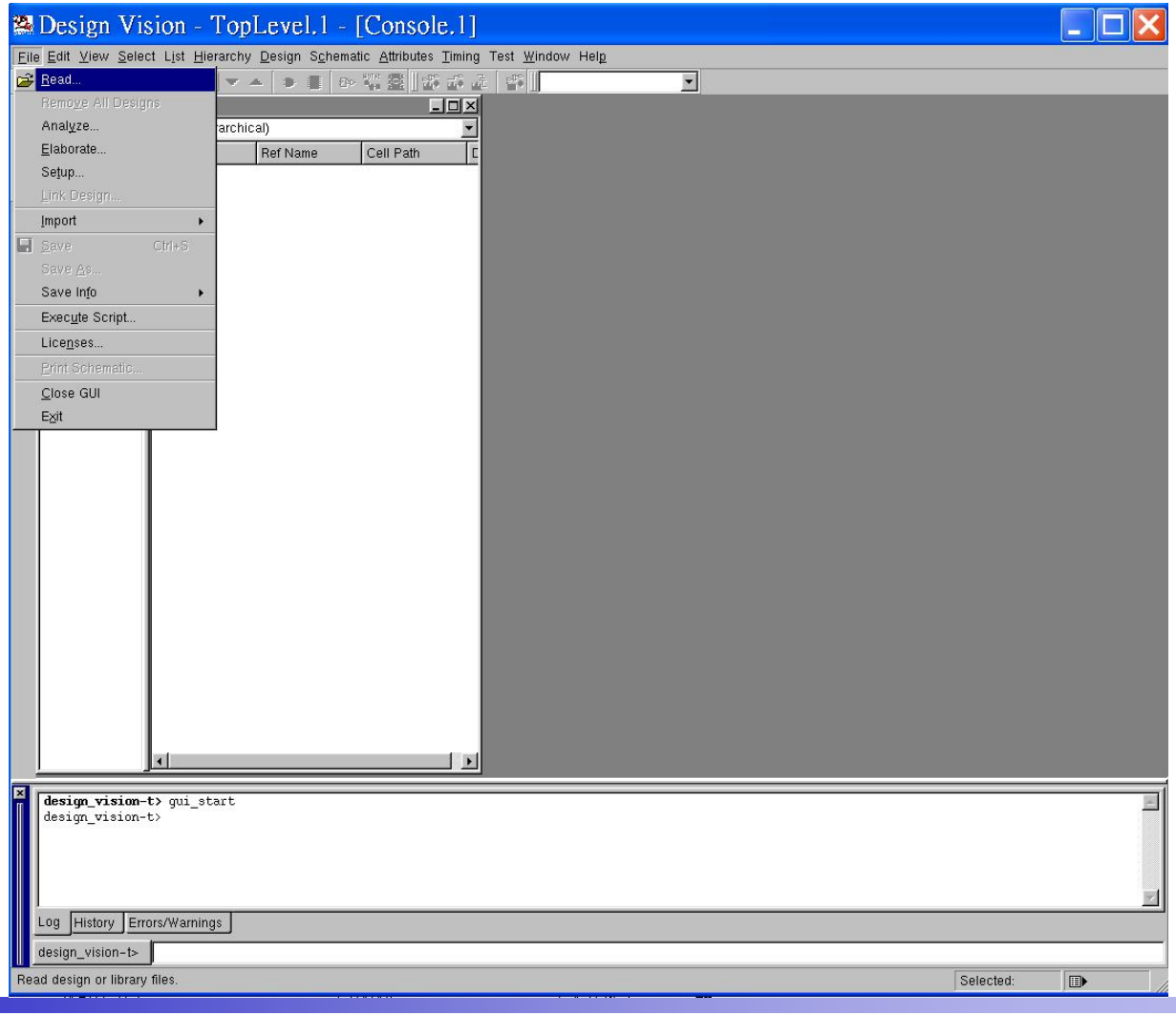
Logic Synthesis Overview



Tools We will Use

Tool	Purpose
Design Vision	User Graphical Interface of synopsys synthesis tool
HDL Compiler	Translate Verilog descriptions into Design Compiler
Design Compiler	Constraint driven logic optimizer
Design Time	Static Timing Analysis (STA) engine
DfT Compiler	Design for Testing
Design Ware	Enable synthesis using DesignWare library

Design Vision

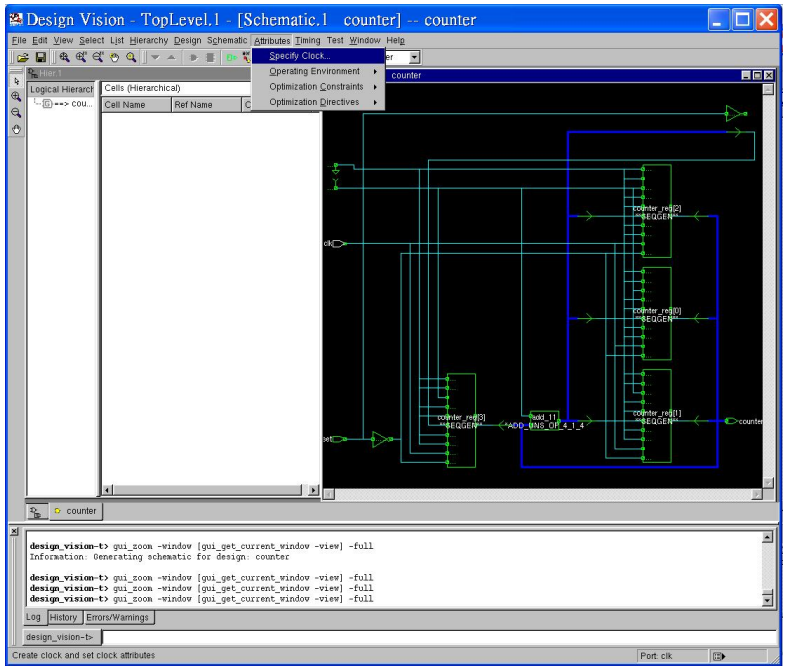


HDL Compiler

```
always @(reset or set)
begin : direct_set_reset
  if (reset)
    y=1'b0;
  else if (set)
    y=1'b1;
end
always @(gate or reset)
  if (reset)
    t=1'b0;
  else if (gate)
    t=d;
```

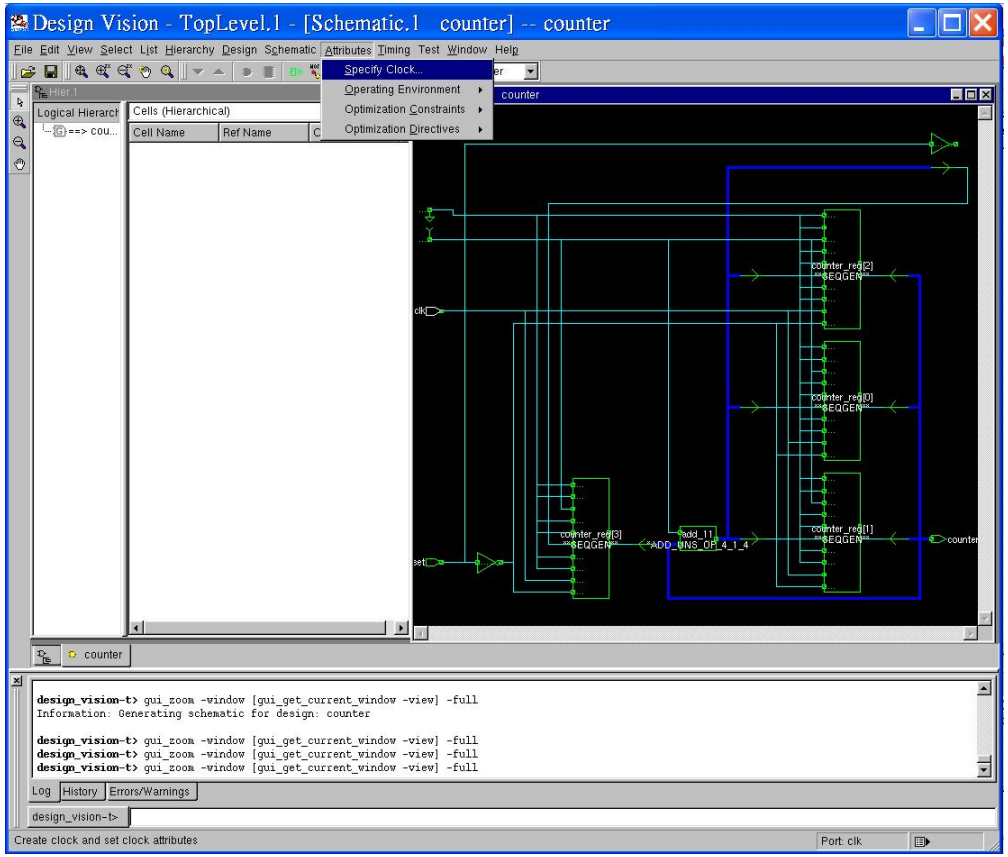
HDL Comipler

HDL Compiler translates Verilog HDL descriptions into Design Compiler as Synopsys *design block*



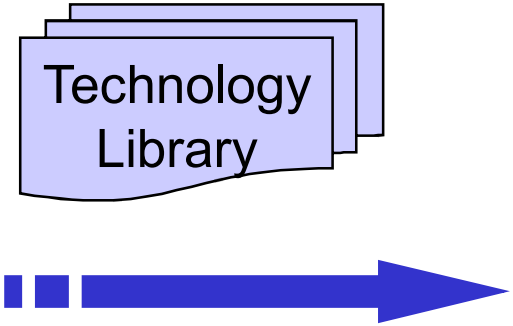
HDL Compiler

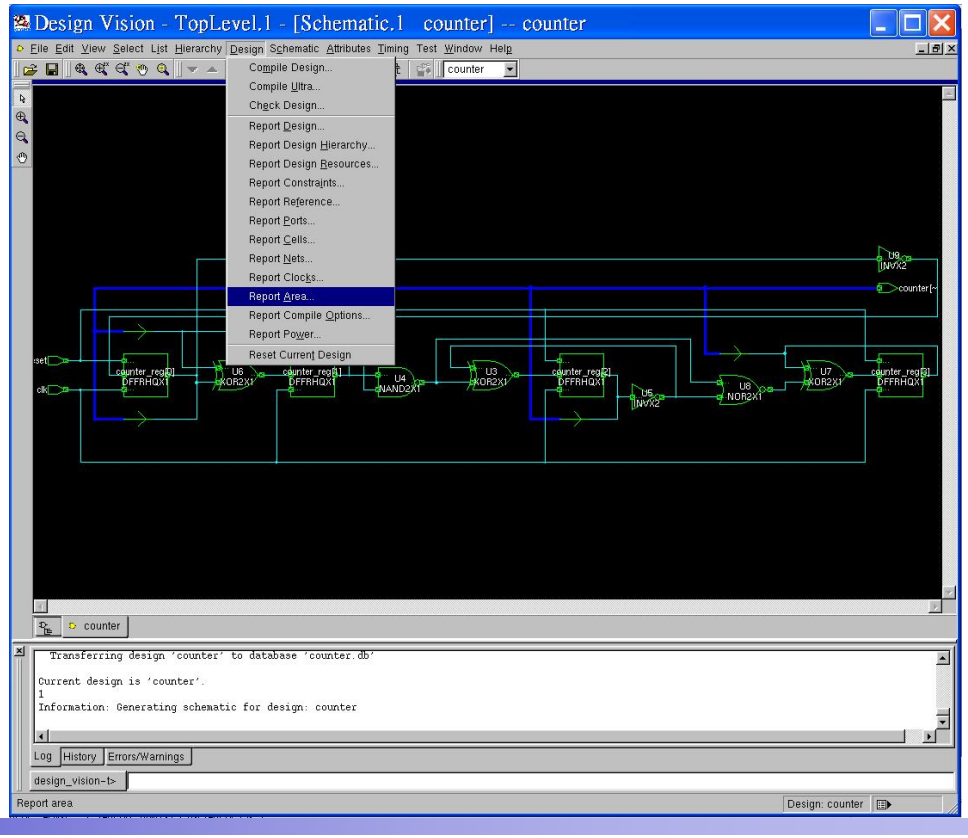
- In schematic view, we can see the Verilog file is translated with a *GTECH* library (the synopsys default)



Design Compiler

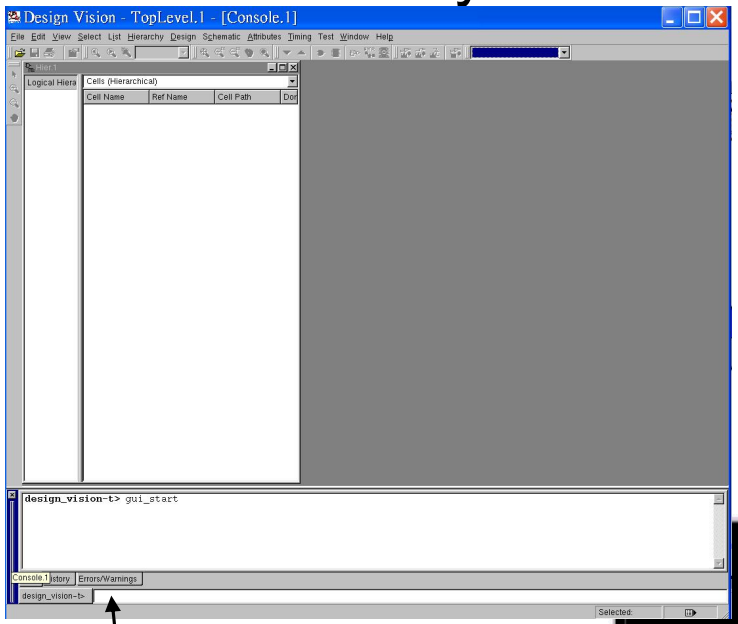
- Design Compiler maps Synopsys design block to gate level design with a user specified library



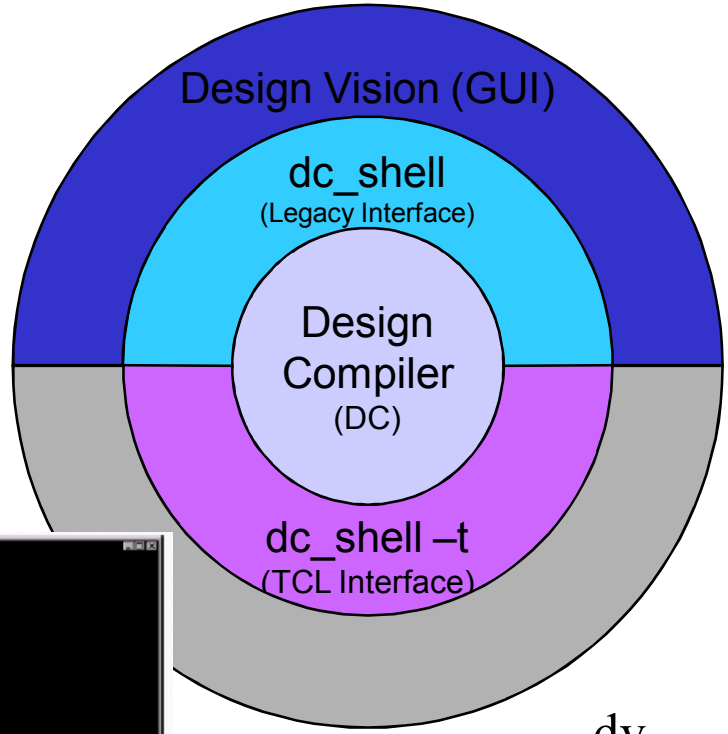


Design Compiler Interaction

- Three ways to interface

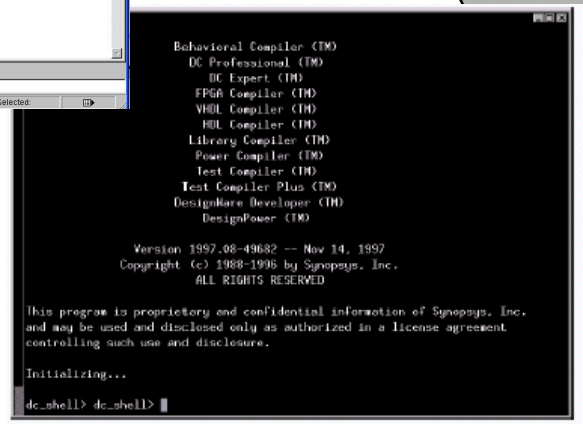


dv



dv

Command line



dc_shell

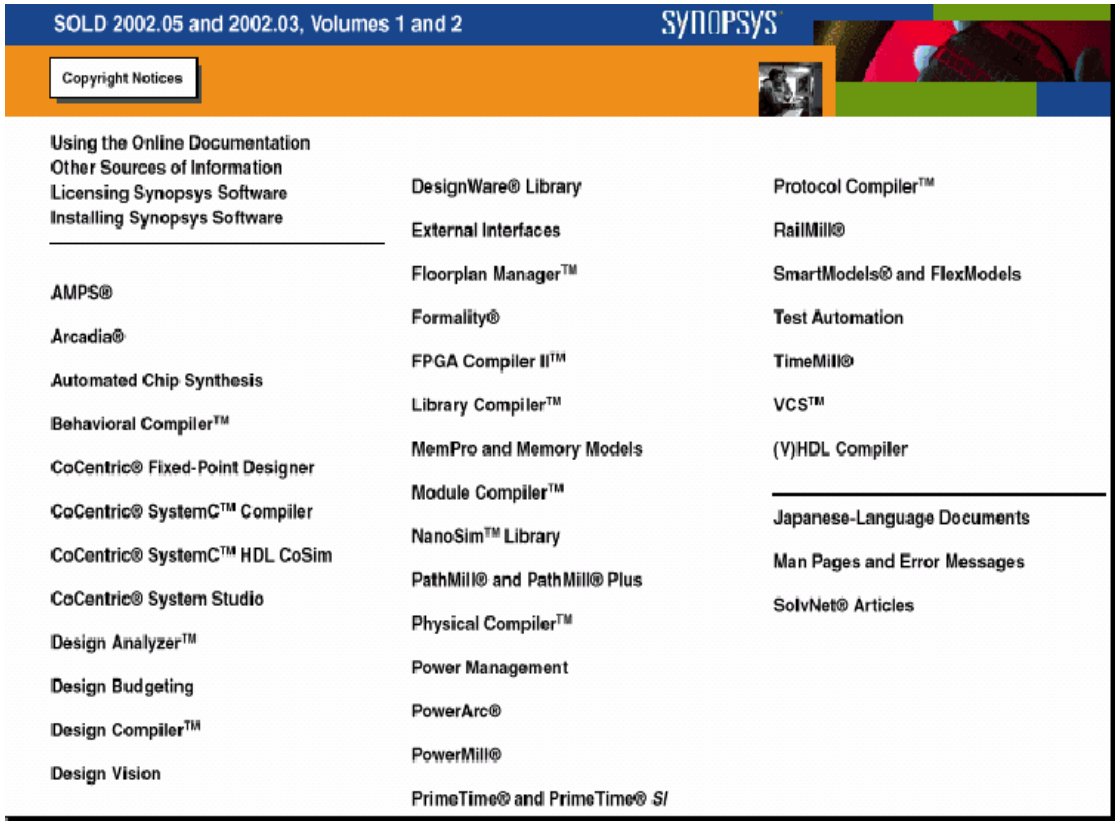
Synopsys Related Files

Files	Purpose
.cshrc	Set path and environment variables and license check
.Xdefault	Set X terminal display variables
.synopsys_dc.setup	Three distinct files are read and executed when DC is invoked 1. system-wide (do not modify): (e.g. $SYNOPSYS/admin/setup/) 2. User's home directory (e.g. ~think/) 3. User's current working directory (e.g. ~d95014/work/dsd/)

- Note
  - These 3 files are always read *in the same order*.
  - Any repeated command can *override* the previous one.

Synopsys On-Line Documentation (SOLD)

- Invoke Synopsys On-Line Document using the command
  - unix%> *acroread /usr/synopsys/sold/cur/top.pdf*
- Note: whenever you find a question, check *SOLD* first



What .synopsys_dc.setup defined

- link_library: the library used for interpreting input description
  - Any cells instantiated in your HDL code
  - Wire Load or Operating Condition models used during synthesis
- target_library: the ASIC technology that the design is mapped to
- symbol_library: used during schematic generation
- search_path: the path to search for unsolved reference library or design
- synthetic_library: designware library to be used

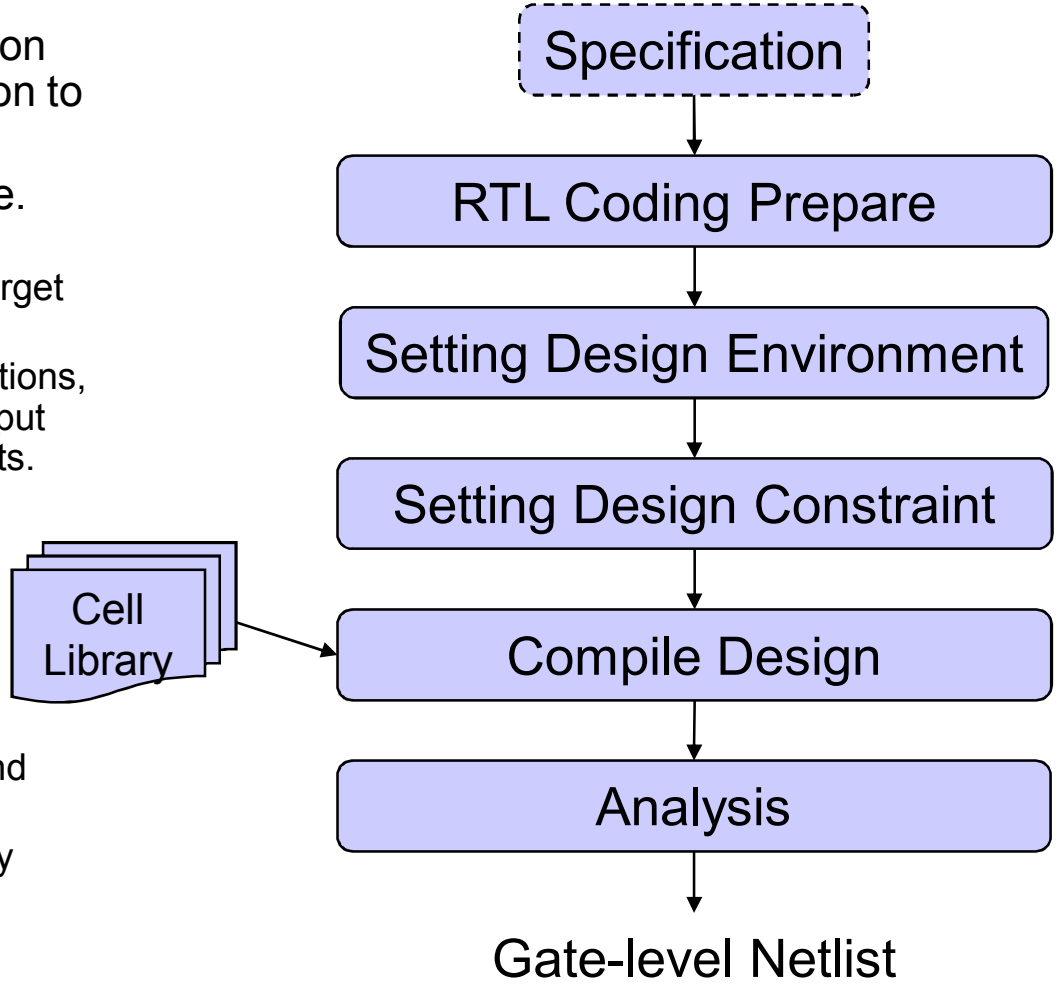
- Other variables

.synopsys_dc.setup file

```
set search_path [concat [list . /home/raid1_1/cic/CBDK018_UMC_Artisan/CIC/SynopsysDC/] $search_path]
set link_library [list "dw_foundation.sldb" "typical.db" "umc18io3v5v_typ.db"]
set target_library [list "typical.db" "umc18io3v5v_typ.db"]
set symbol_library [list "umc18.sdb" "umc183v5v.sdb"]
set default_schematic_options {-size infinite}
set synthetic_library [list "dw_foundation.sldb"]

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

Synthesis Design Flow

- Develop the HDL design description and simulate the design description to verify that it is correct.
- Set up the .synopsys_dc.setup file.
  - Set the appropriate technology, synthetic, and symbol libraries, target libraries, and link libraries.
  - Set the necessary compilation options, including options to read in the input files and specify the output formats.
- Read the HDL design description.
- Define the design.
  - Set design attributes
  - Define environmental conditions
  - Set design rules
  - Set realistic constraints (timing and area goals)
  - Determine a compile methodology

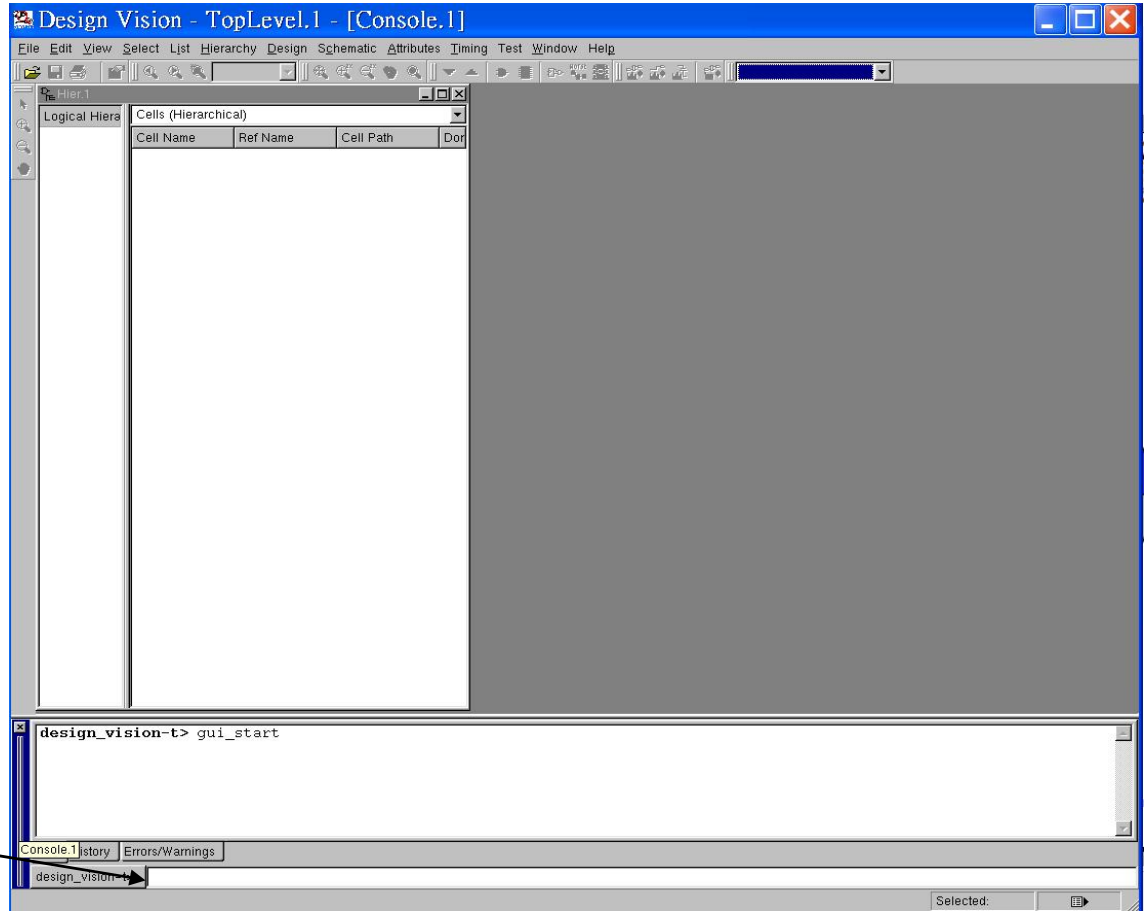


Digital System Design

Synopsys Graphical Environment

Invoke Design Vision

- Linux%> **dv** &



tcl command

Optimization Using the Design Vision

- File/Analyze & File/Elaborate - Verilog &VHDL, or File/Read - all other formats
- Attributes - set up Design Environment & Goals
- Analysis/Report - check if set up is OK
- Analysis/Check Design
- Tools/Design Optimization
- Analysis/Report
- File/Save

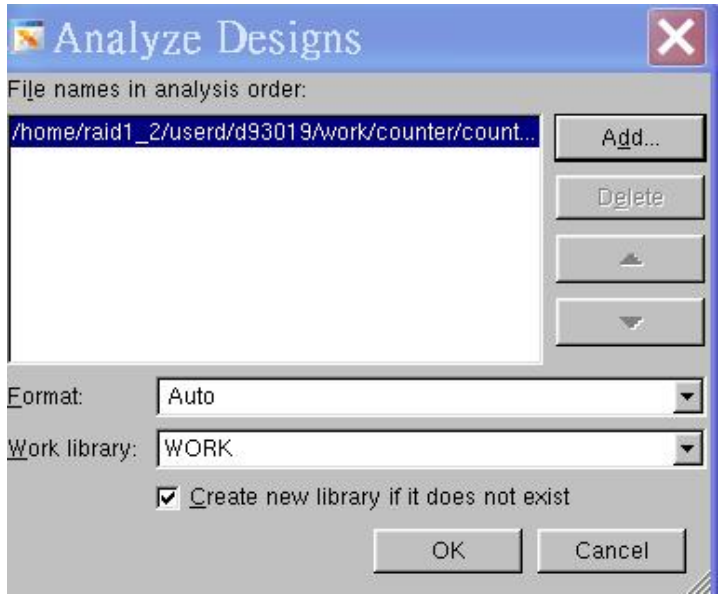
Analyze & Elaborate

- Use analyze and elaborate to bring Verilog or VHDL files into design compiler memory
- Analyze does syntax checking and produces an intermediate .syn .mra files to be stored in a design library
- Elaborate looks in the design library for the .syn file and builds the design up into design compiler memory (as design block)

Analyze

- Check VHDL & Verilog for syntax and synthesizability
- Create intermediate .syn and .mra files and places them in library specified – design library

File/Analyze



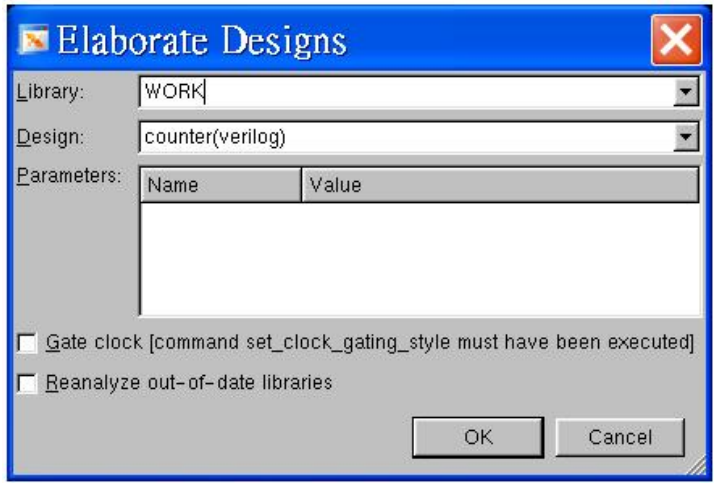
- Equivalent to dc_shell command

```
analyze -format verilog –library WORK counter.v
```

Elaborate

- Elaborate after analyze to bring design into Design Compiler memory using generic components (GTECH)
- Look in the design library for intermediate .syn file for design specified

File/Elaborate

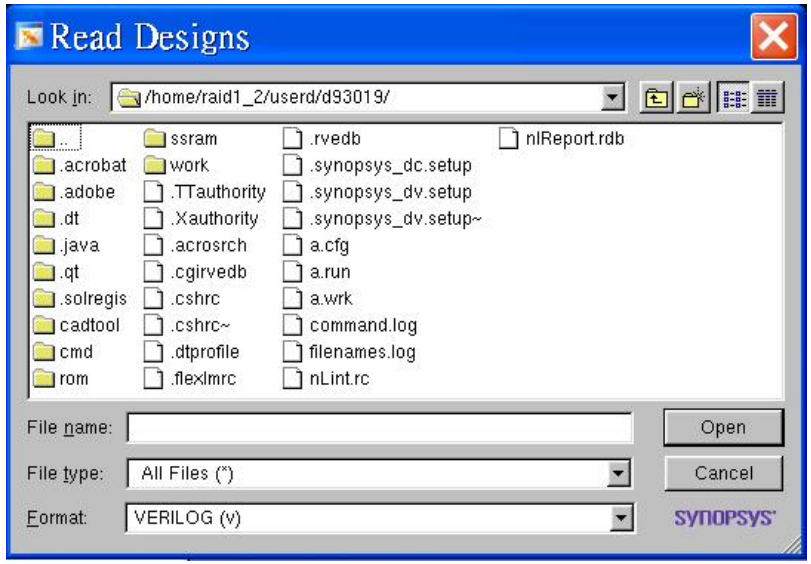


- Equivalent dc_shell

```
elaborate counter -architecture verilog -library WORK -update
```

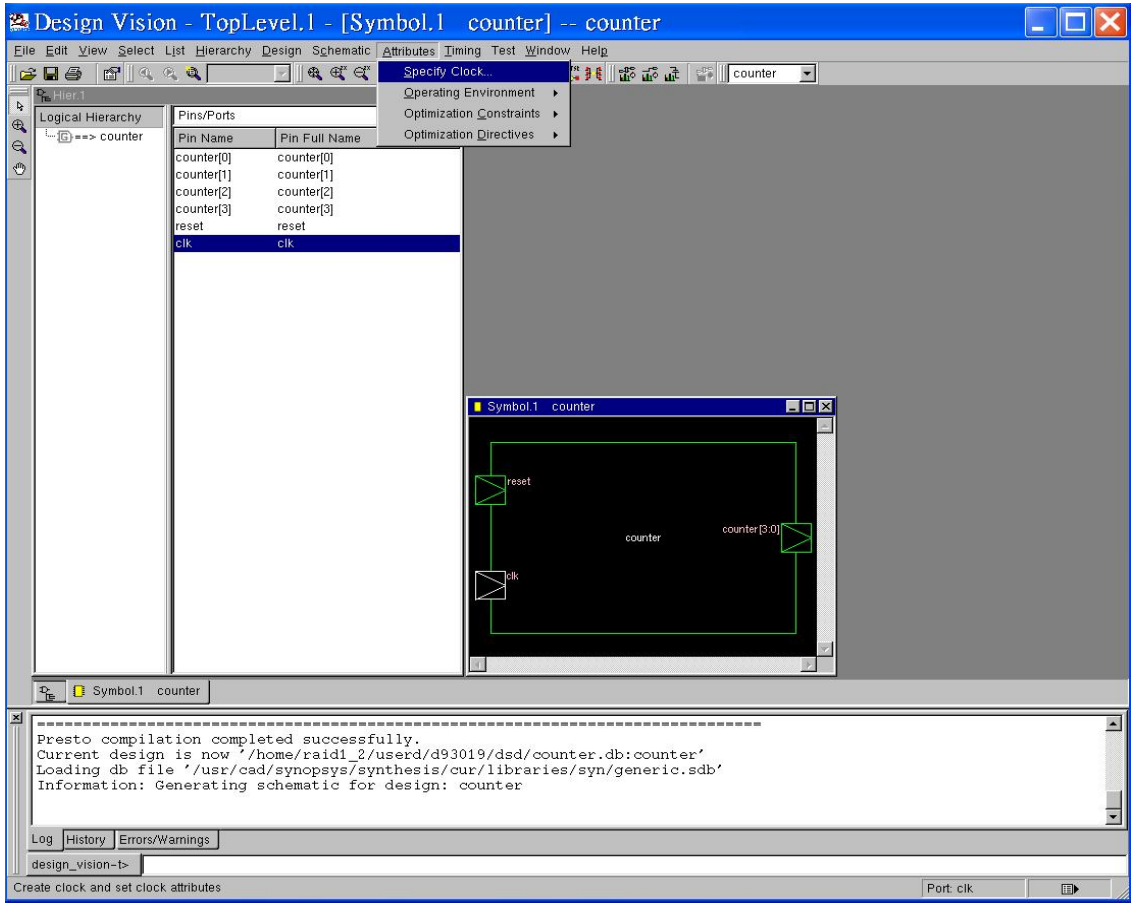
Read File

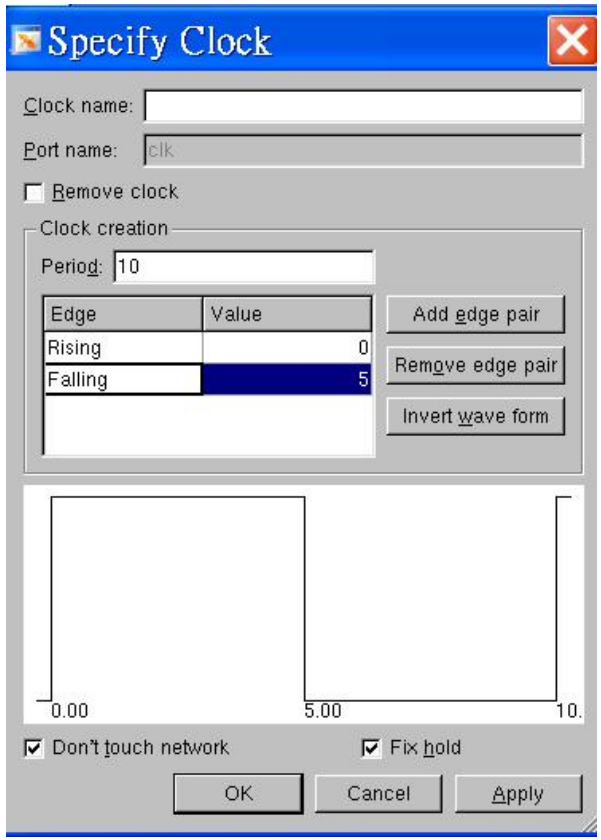
- Read netlists or other design descriptions into Design Compiler
- File/Read
- Support many different formats:
  - synopsys internal formats
    DB(binary): .db
    equation: .eqn
  - state table: .st
  - Verilog: .v
  - VHDL: .vhd
  - PLA(Berkeley Espresso): .pla
  - EDIF



Describe the Design Environment

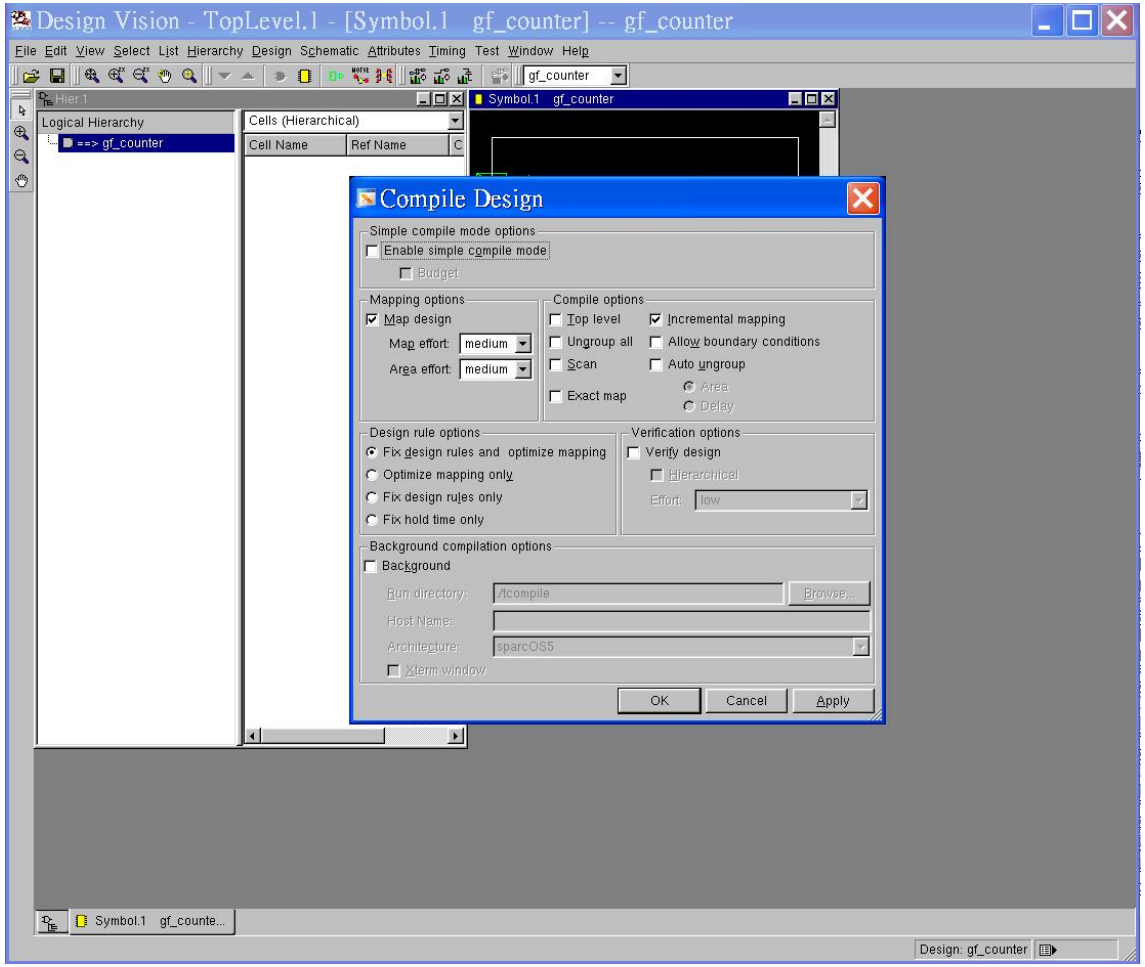
- You can use Design Vision to constrain your design





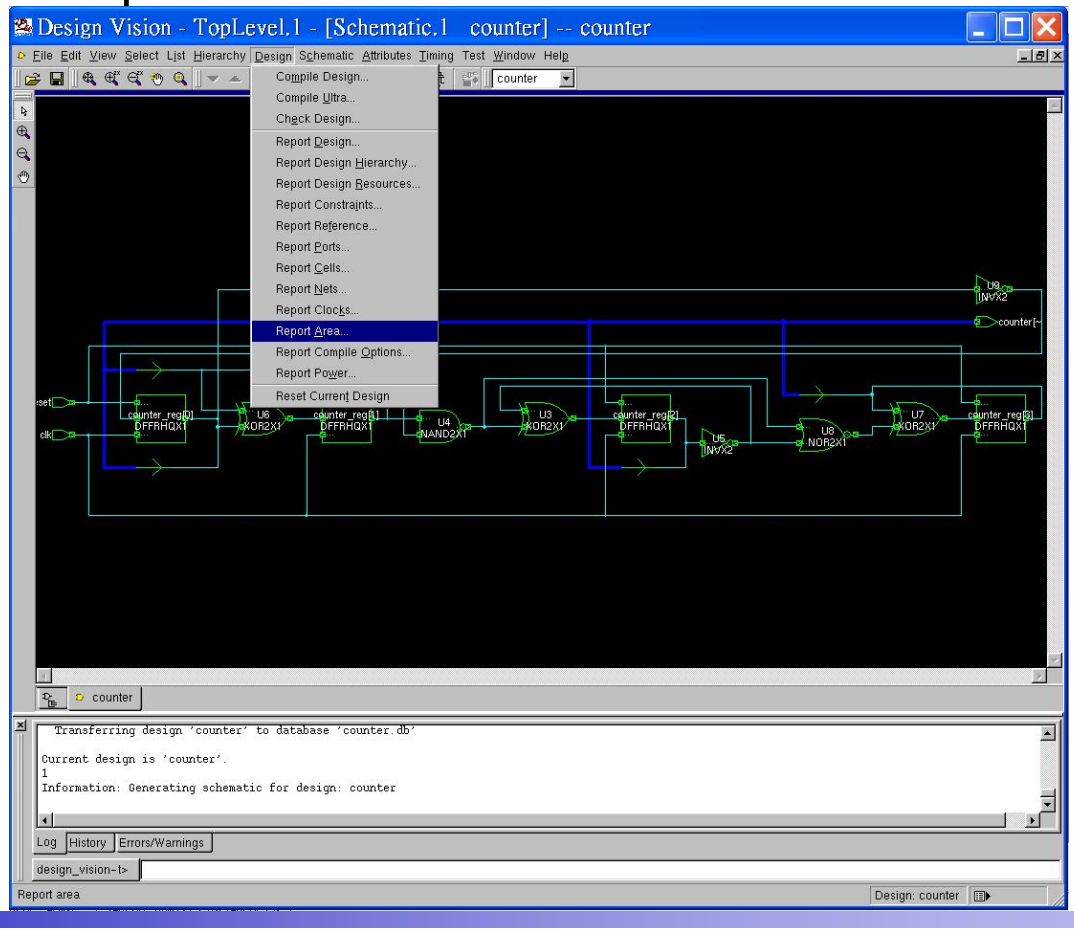
Compile the Design

- The compile command optimizes and maps the current_design



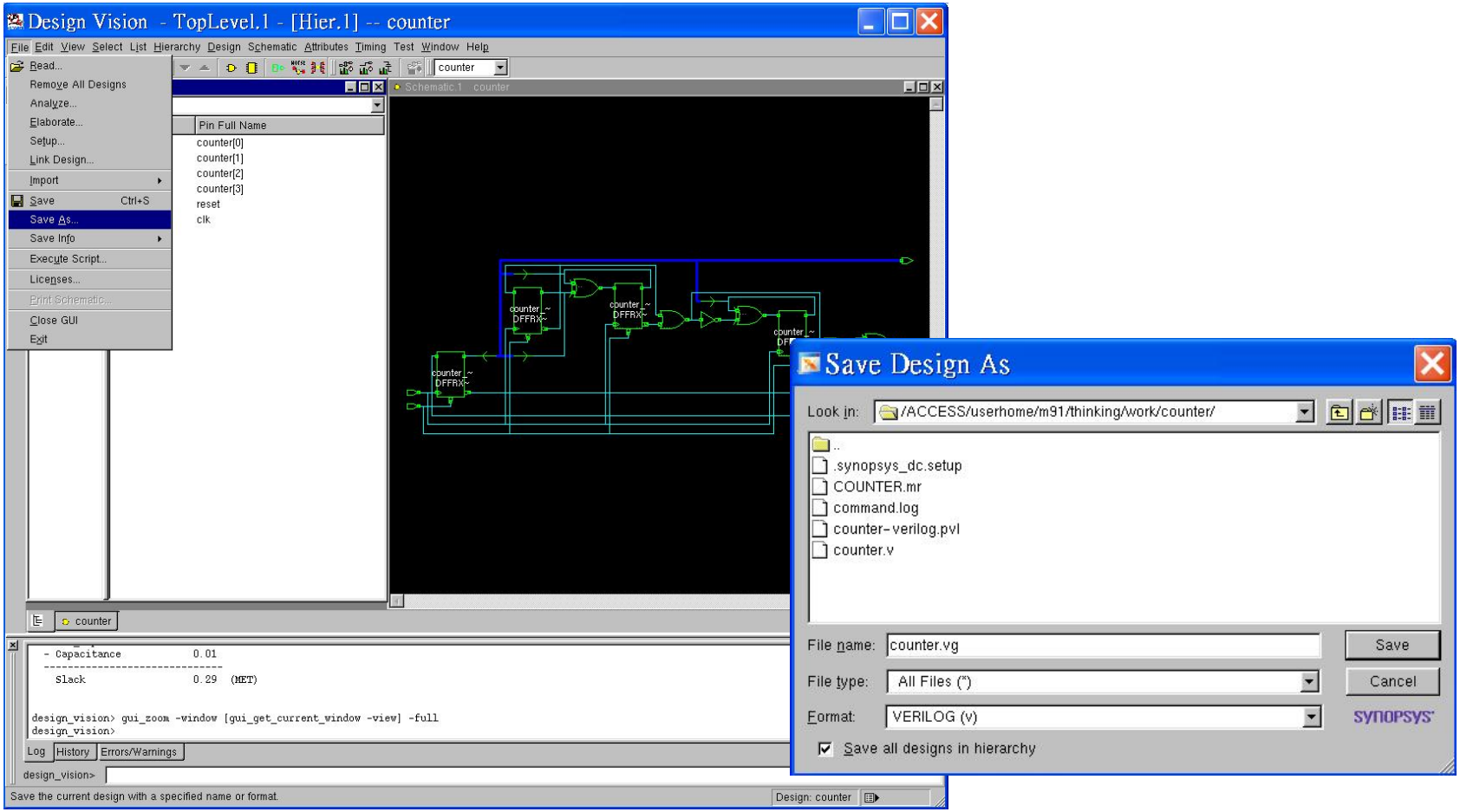
Report the Design

- From report and analysis, you can find the set attributes and the results after optimization



Save the Design

- Write out the design netlist after synthesis



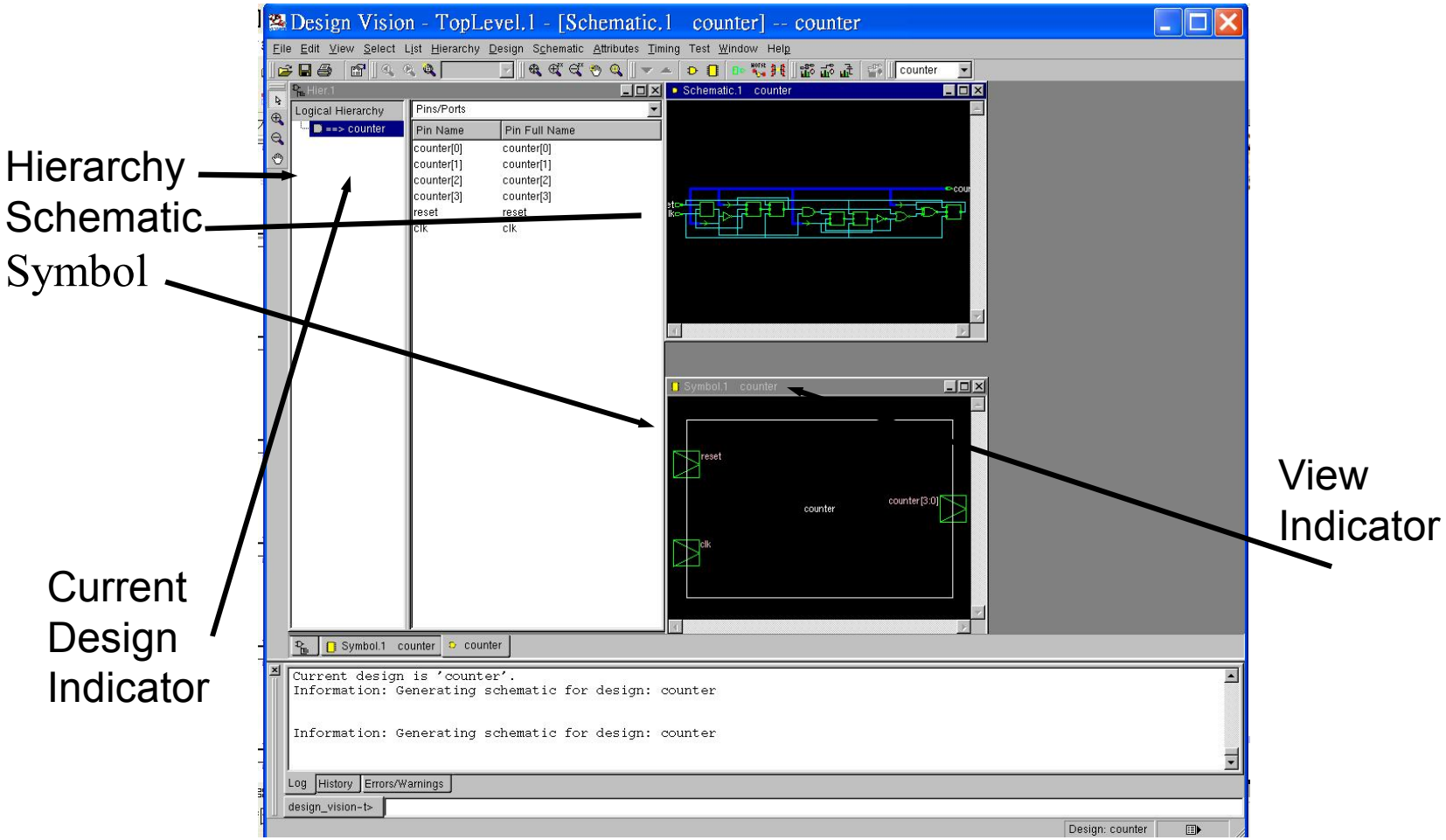
Link Design

- *Analysis/Link Design*
- Execute *link -all* before you optimize your design
- To ensure all sub-elements of your hierarchical design are available

Check Design

- *Analysis/Check Design*
- Execute check_design before you optimize your design
- Two types of messages are issued
- error
  - Error: In design ‘bcd7segs’, cell ‘decoder’ has more pins than it’s reference ‘d1’ has ports
- warnings
  - Warning: In design ‘converter’, port ‘A’ is not connected to any nets

Different View - Design View



Digital System Design

Design Constraints Settings

- Setting Design Environment
- Setting Design Constraint

Graduate Institute of Electronics Engineering, NTU

Digital System Design

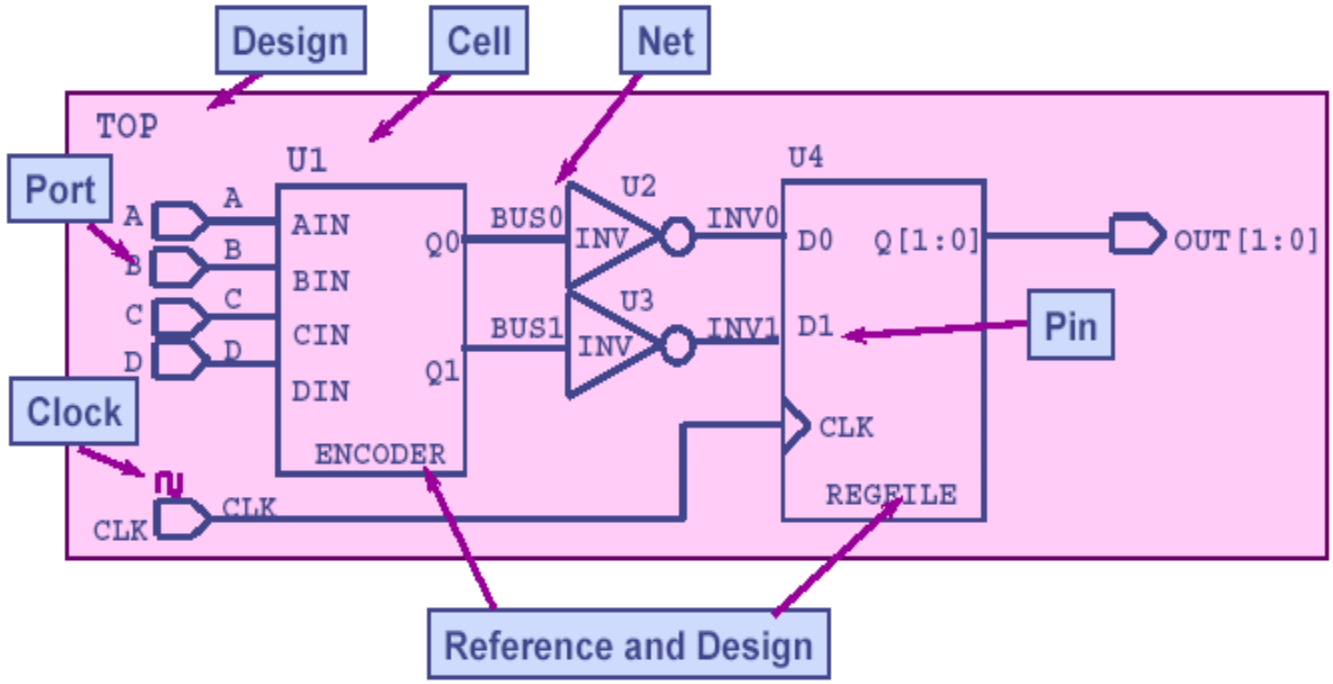
Setting Design Environment

Design Objects

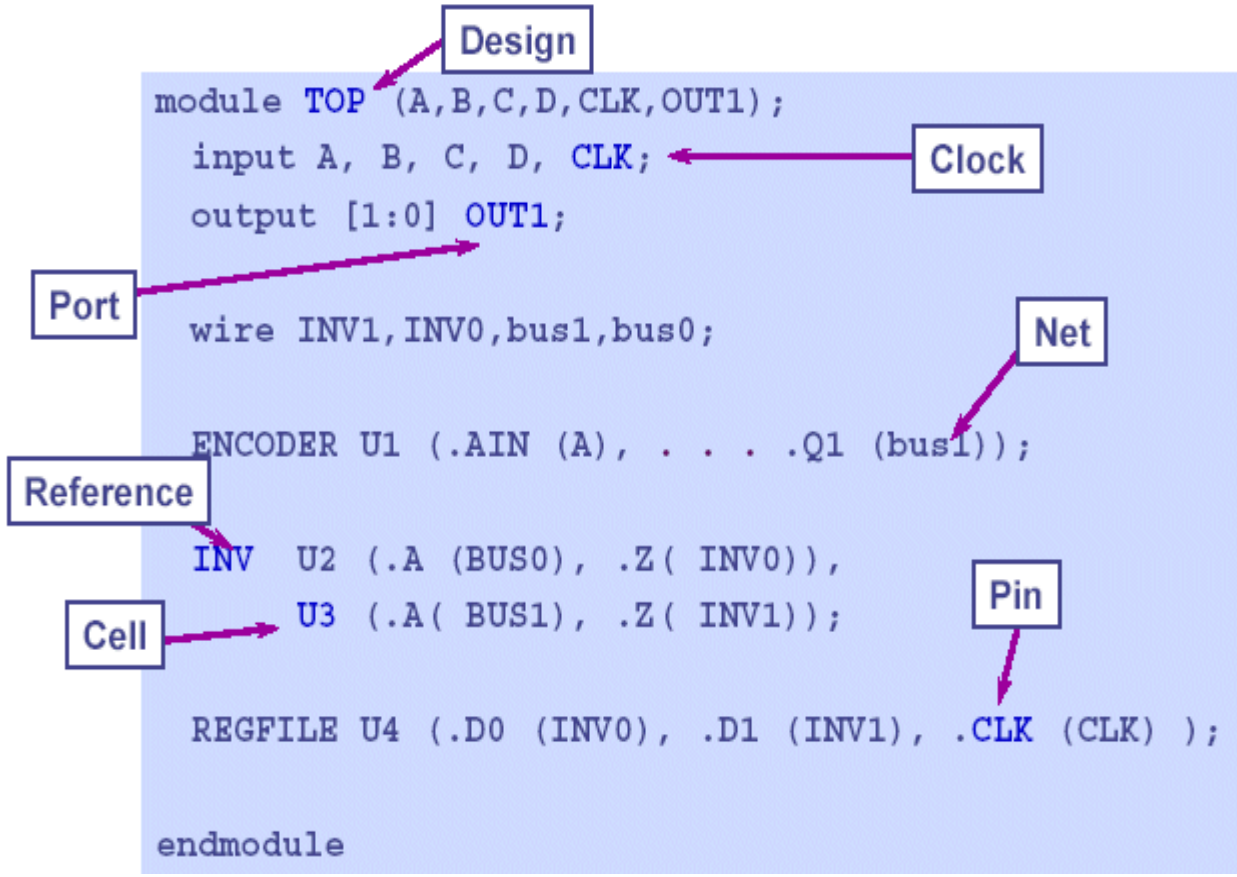
- Seven Types of Design Objects:

Design:	A circuit that performs one or more logical functions
Cell:	An instance of a design or library primitive within a design
Reference:	The name of the original design that a cell instance “points to”
Port:	The input or output of a design
Pin:	The input or output of a cell
Net:	The wire that connects ports to pins and/or pins to each other
Clock:	A timing reference object in DC memory which describes a waveform for timing analysis

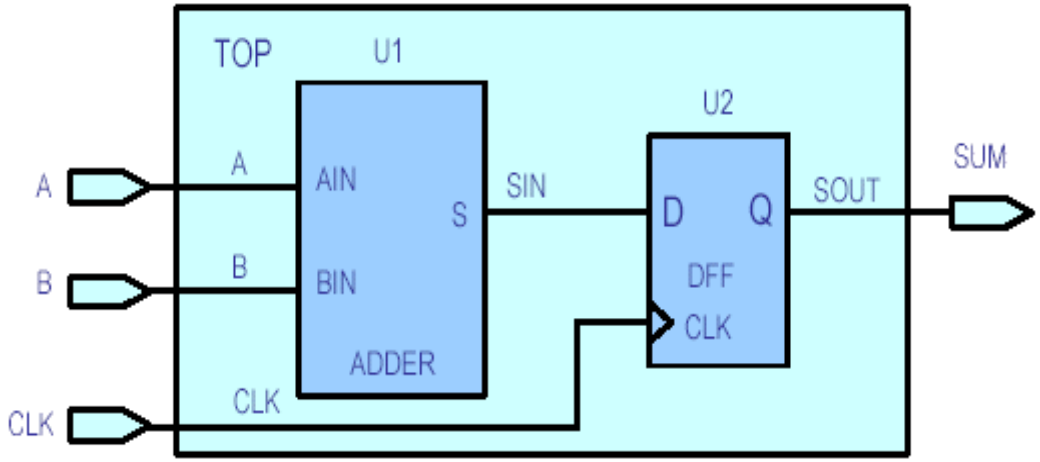
Design Objects (Schematic Perspective)



Design Objects (Verilog Perspective)



Design Objects Exercise



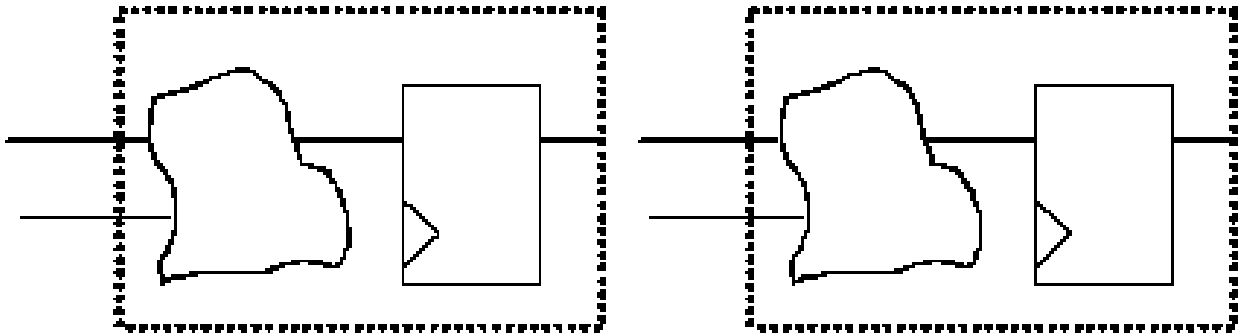
- Make a list of all the ports in the design?{ }
- Make a list of all the cells that have the letter “U” in their name? { }
- Make a list of all the nets ending with “CLK”? { }
- Make a list of all the “Q” pins in the design? { }
- Make a list of all the references? { }

Partitioning for Synthesis

- Register at hierarchical output, keep related combinational logic together at the same module
- Separate modules having different design goals
- Avoid asynchronous logic, false path, and multi-cycle path
- Avoid the glue logic
- Keep major blocks separate
- Additional Issues

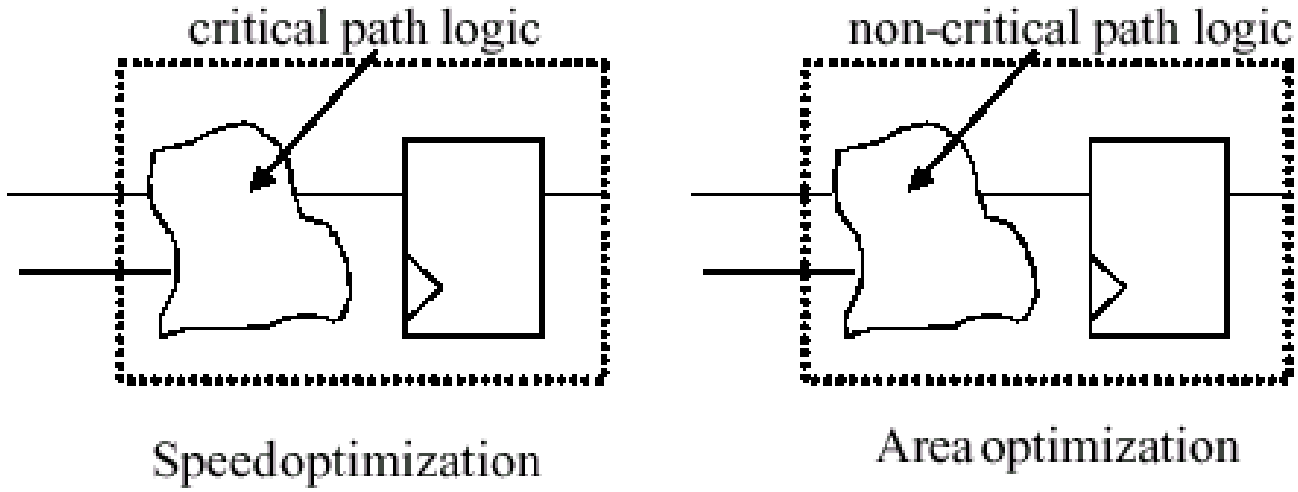
Register at Hierarchical Output

- Keep related combination logic in a single module.
- Register all at output make input data arrival time and output drive strength predictable.



Partition by Design Goals

- Optimize the critical path logic for speed
- Optimize non-critical path logic for area.



Avoid Non-normal Paths

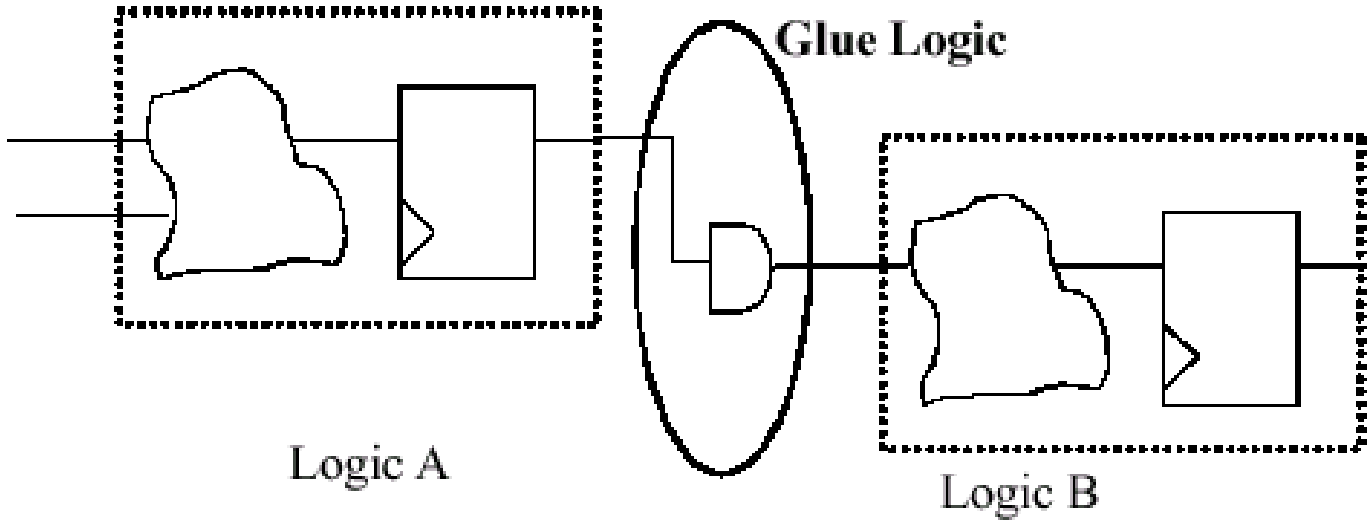
- Asynchronous logic is more difficult to design correctly and verify.
- Isolating the asynchronous logic in a separate module.
- False path and multi-cycle paths easily cause the human error.
- If false and multi-cycle paths required, use

  `set_false_path` and **`set_multicycle_path`**

  command to identify.

Avoid Glue Logic

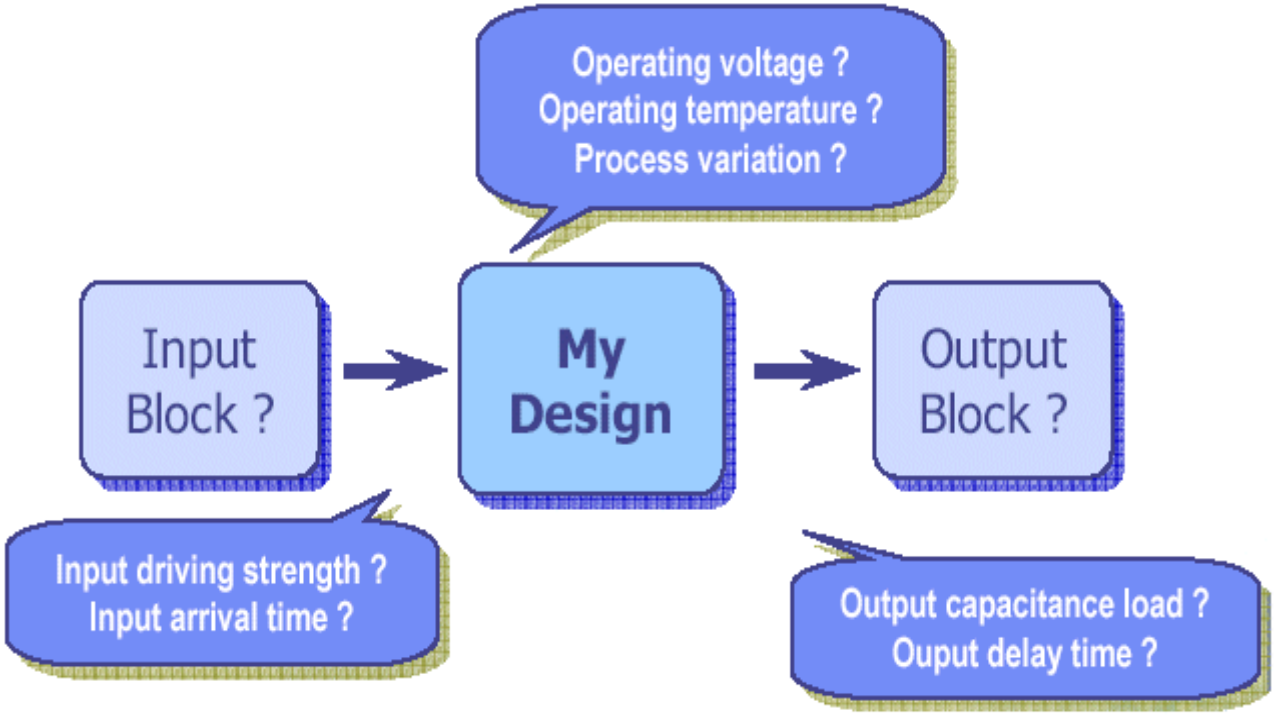
- No Cells except at leaf levels of hierarchy
- Any extra gates should be grouped into a sub-design



Why Describes the Real World Environment

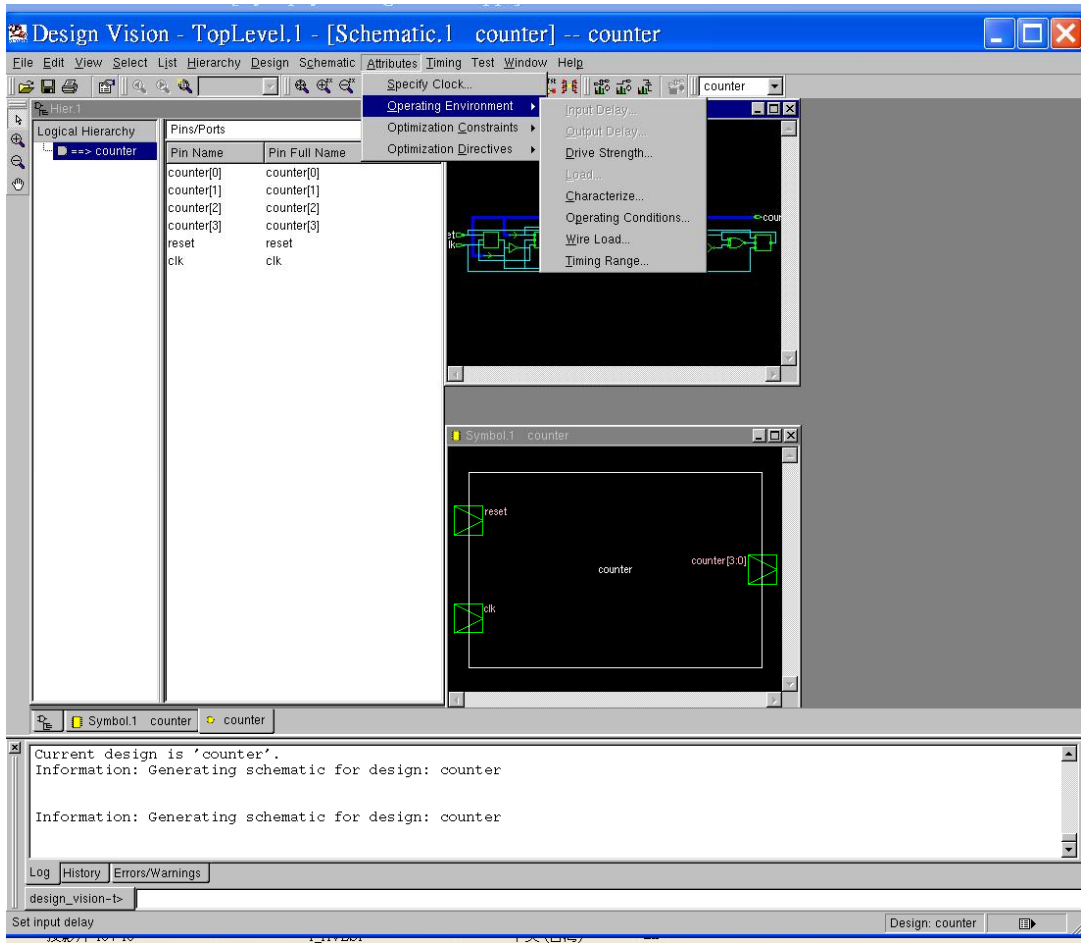
- Beware that the defaults are not realistic conditions.
  - Input drive is not infinite
  - Capacitive loading is usually not zero
  - Consider process, temperature, and voltage variation
- The operating environment affects the components selected from target library and timing through your design.
- The real world environment you define describes the conditions that the circuit will operate within.

Describing Design Environment



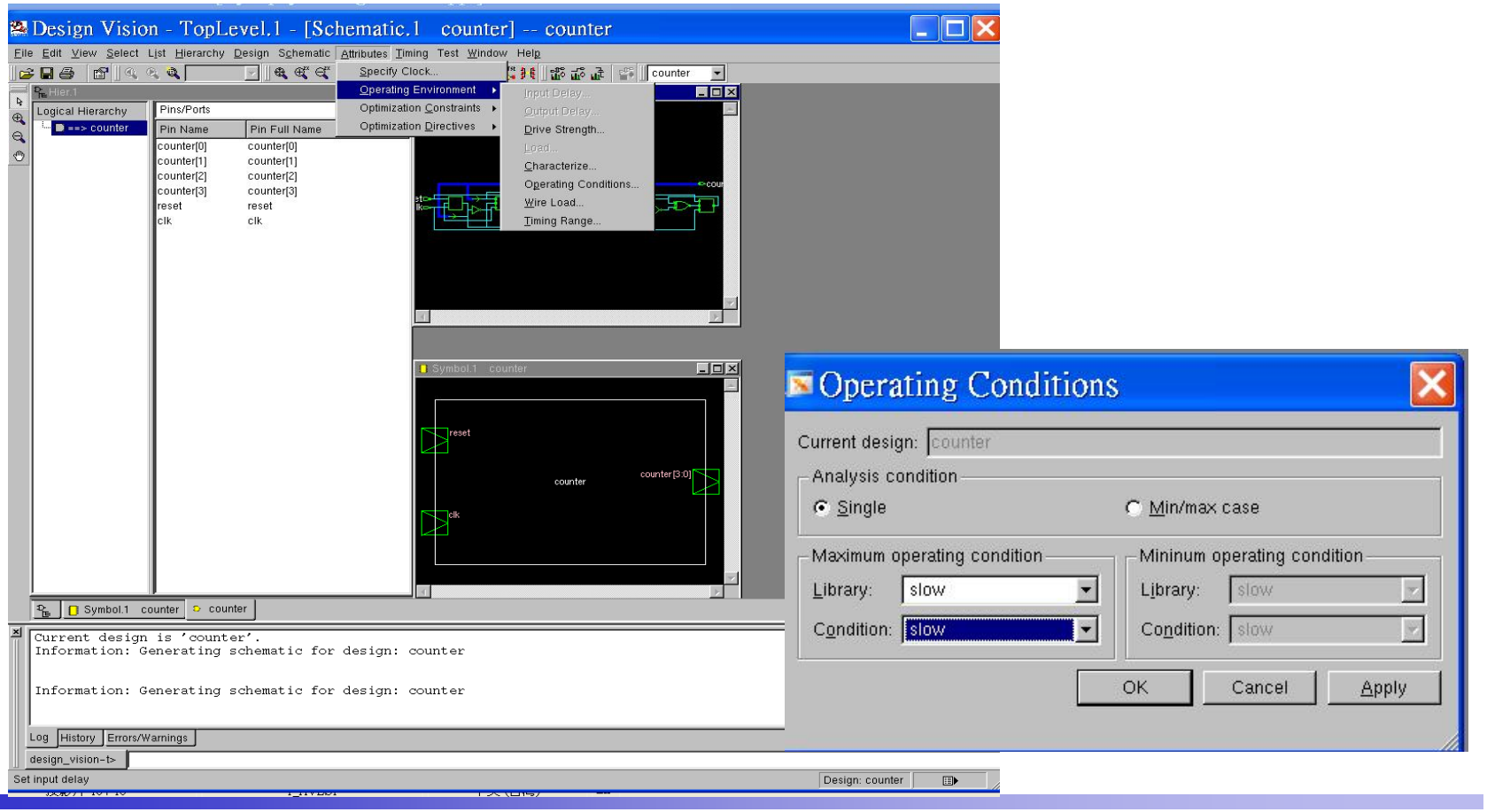
Setting Operating Environment

- Attributes/Operating Environment



Setting Operating Condition

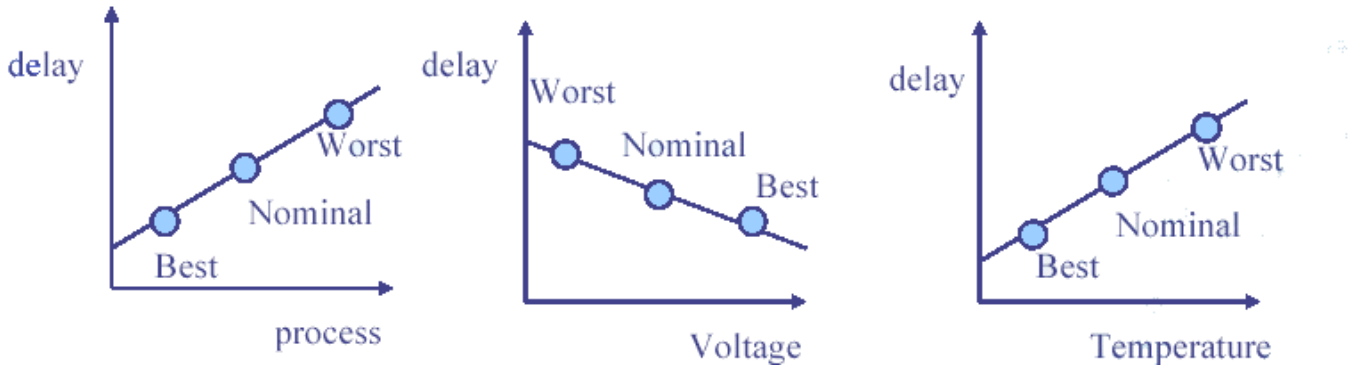
- Attributes/Operating Environment/Operating Condition
  - dc_shell> set_operating_conditions “slow”



Operating Condition

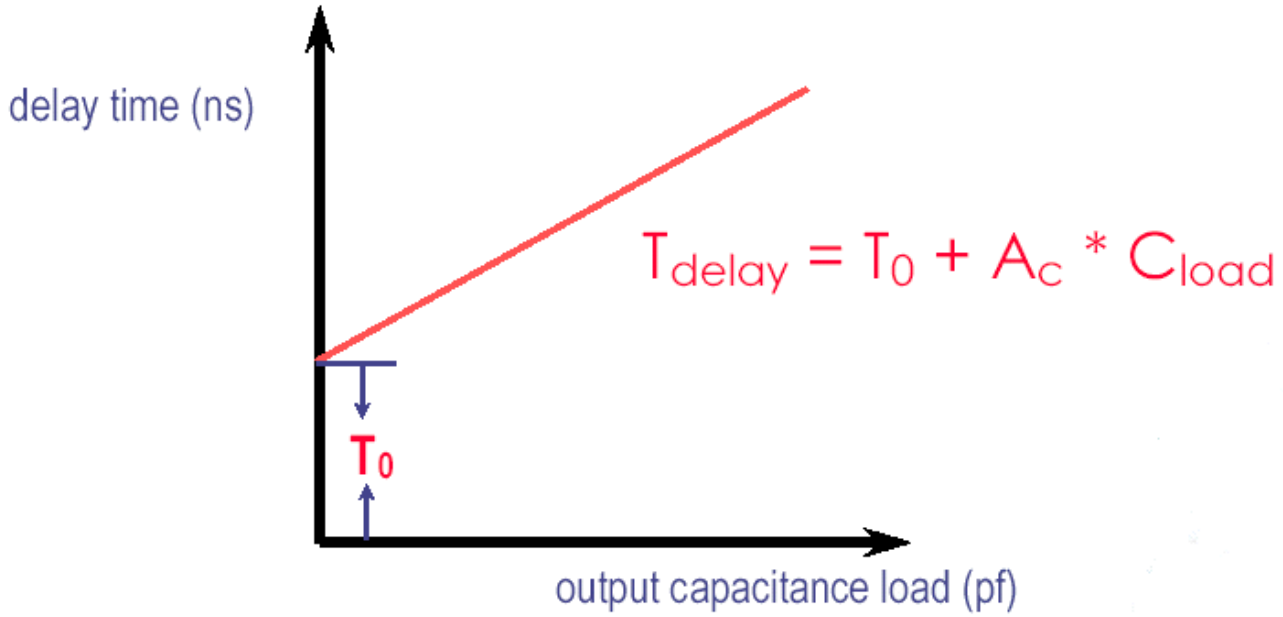
- Operating condition model scales components delay, directs the optimizer to simulate variations in process, temperature, and voltage.

Name	*Process*	*Temp*	*Volt*	*Interconnection Model*
WCCOM	1.32	100.00	2.7	worst_case_tree
BCCOM	0.73	0.00	3.6	best_case_tree
NCCOM	1.00	25.00	3.3	balance_tree



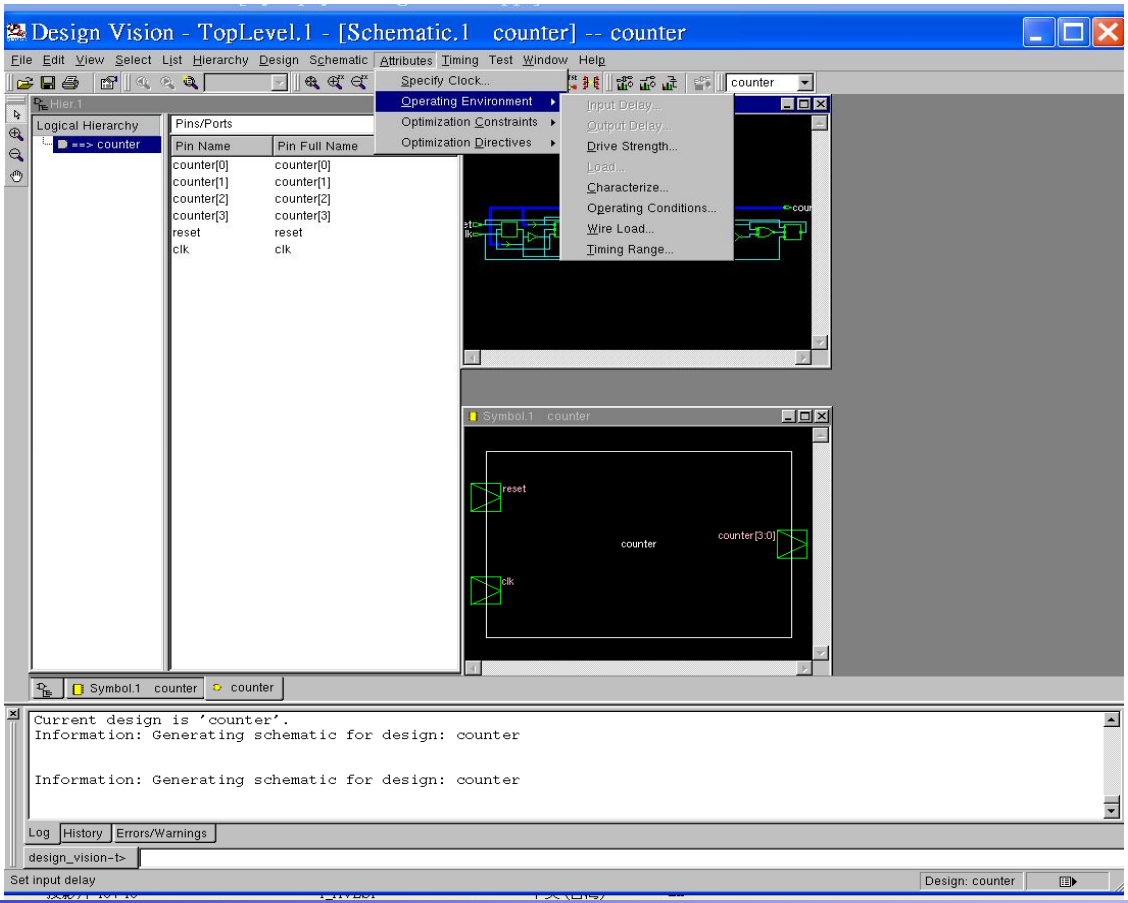
Input Drive Impedance

- $T_{delay} = T_0 + A_c * C_{load}$
  - T_0 : cell pin to pin intrinsic delay
  - A_c : drive impedance (unit: ns/pf)



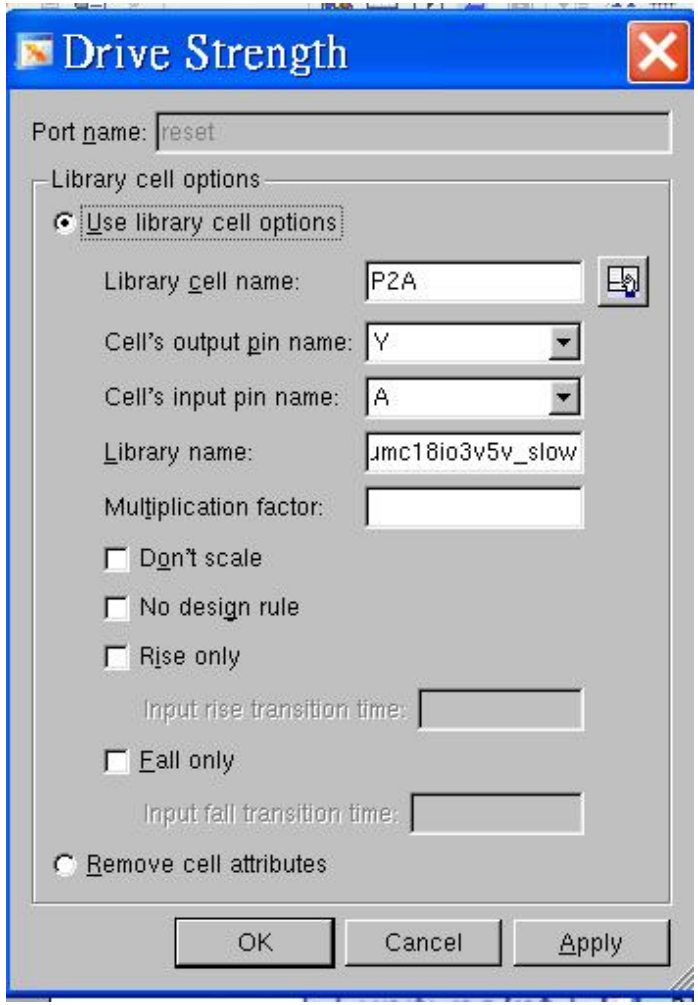
Setting Input Drive Impedance

- Attribute/Operating Environment/Drive Strength



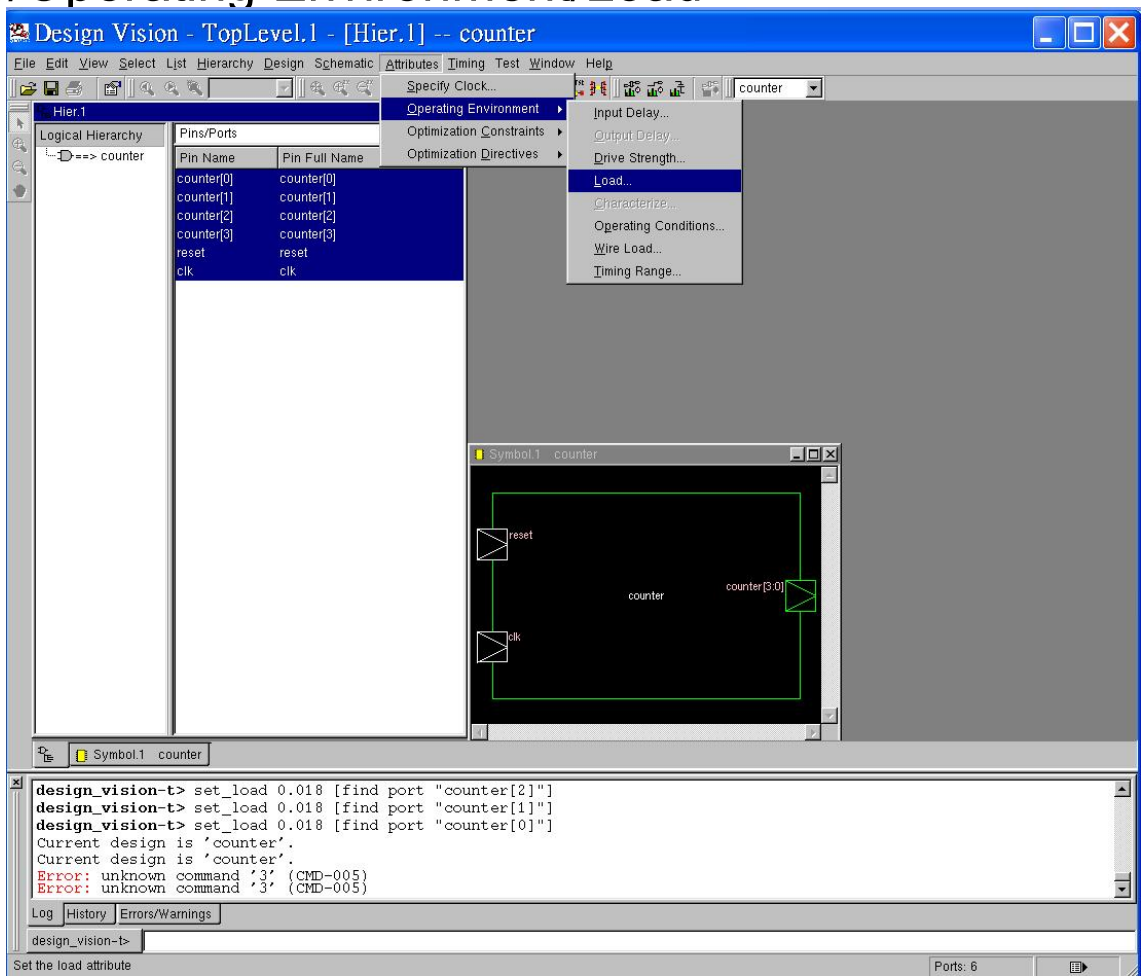
Setting Input Drive Impedance

- Also can be set using "drive_of" command
  - Example: (P2A cell output)



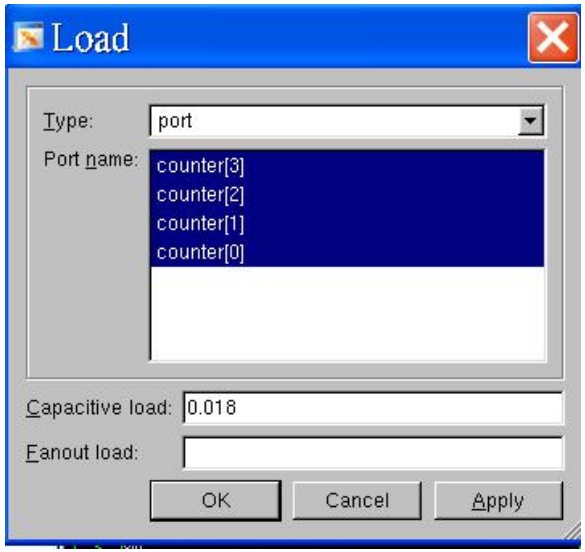
Setting Output Loading

- Attribute/Operating Environment/Load



Setting Output Loading

- Also can be set using load_of:
  - Example: (bufferd1 cell input)



Port Report

- dc_shell command
  - dc_shell> report_port -verbose {port_list }
- or in the option menu set verbose

```
****************************************
Report : port
Design : counter
Version: W-2004.12-SP1
Date   : Fri May  6 12:52:28 2005
****************************************


                        Pin      Wire     Max      Max      Connection
Port            Dir     Load     Load     Trans    Cap      Class       Attrs
--------------------------------------------------------------------------------
counter[3]      out     0.0180   0.0000   --       --       --
counter[1]      out     0.0180   0.0000   --       --       --
reset           in      0.0000   0.0000   --       --       --
counter[2]      out     0.0180   0.0000   --       --       --
clk             in      0.0000   0.0000   --       --       --
counter[0]      out     0.0180   0.0000   --       --       --


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

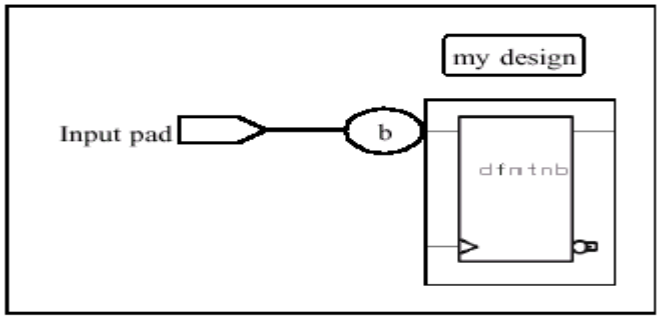
Drive Strength & Load for Pads

- How do you specify the Drive Strength & Output Load for the pins which connect to pads ?
- Example:
  - In CIC's cell_based design flow, we use the Avant! 0.35um cell library. In this library we can find a file named "ds_cb35io122.pdf", andin this file we can find the information for the pads you want to use.

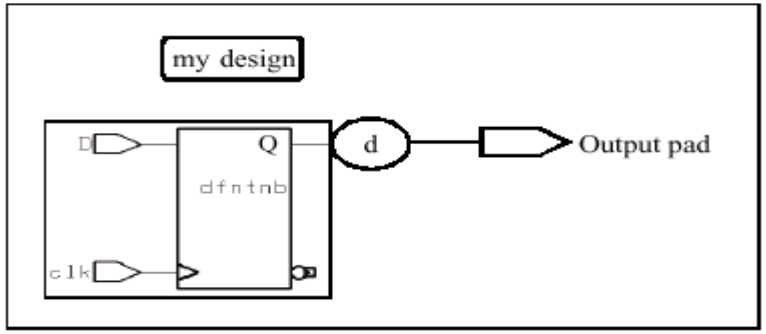
1. Based on the information, set the input drive strength & output load.
2. Or extract the pad boundary condition by using ***characterize*** command, we'll introduce it later.

Drive Strength & Load for Pads

- Input Drive Strength for Pads



- Output Load for Pads



3V CMOS Input Only Pads
PC3D01, PC3D11, PC3D21, and PC3D31

Performance Equations

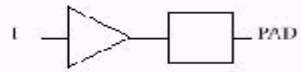
		RISE	FALL
PC3D01			
tCD	PAD -> CIN	0.3377 + 0.0378 * Cld	0.2596 + 0.0503 + 0.1978 * Cld
PC3D11			
tCD	PAD -> CIN	0.3261 + 0.0792 + 0.2265 * Cld	0.3782 + 0.0553 + 0.1960 * Cld
PC3D21			
tCD	PAD -> CIN	0.7315 + 0.0574 + 0.2538 * Cld	0.6064 + 0.0595 + 0.2165 * Cld
PC3D31			
tCD	PAD -> CIN	0.6528 + 0.0289 + 0.2236 * Cld	0.5911 + 0.0253 + 0.1976 * Cld

"ds_cb35io122.pdf"

3V CMOS Output Pads

PC3O01 through PC3O05

PC3O01 through PC3O05 cells are CMOS output pads with AC drive capabilities ranging from 1x to 5x.



Function Table

INPUT	OUTPUT
I	PAD
L	L
H	H

Cell Description

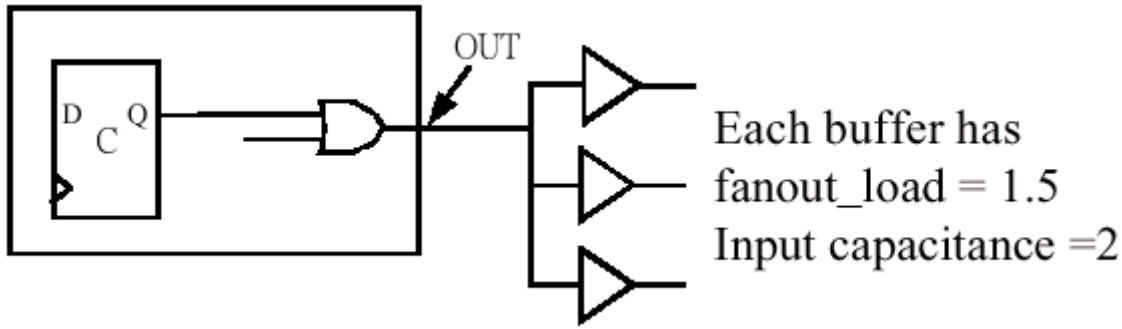
Macro Name:	PC3O01	PC3O02	PC3O03	PC3O04	PC3O05
Drive Capability	1x	2x	3x	4x	5x
Width (mils):	3.4	3.4	3.4	3.4	3.4
Power (µW/MHz):	198.55	198.59	198.75	198.61	197.58

Pin Description

name	Capacitance (pF)					Description
	PC3O01	PC3O02	PC3O03	PC3O04	PC3O05	
I	0.096	0.096	0.095	0.132	0.133	Input
PAD	8.577	8.577	8.577	8.576	8.575	Output

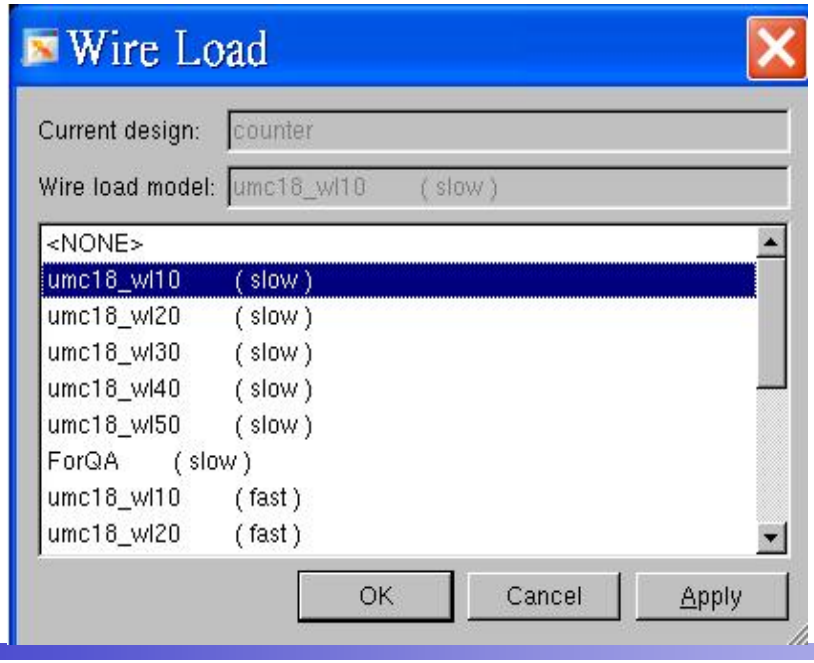
Setting the Fanout Load

- Use *fanout_load* to regulate *max_fanout*
- Syntax: *set_fanout_load fanout portlist* (for listed output ports)
- Design Rule: external fanout_load + internal fanout_load must be smaller than max_fanout_load
  - "OUT"has external fanout_load 4.5 (1.5 x 3)
  - "OUT"has external capacitance 6 (2 x 3)
  - "OUT"has the number of fanout 3 (3 buffers)

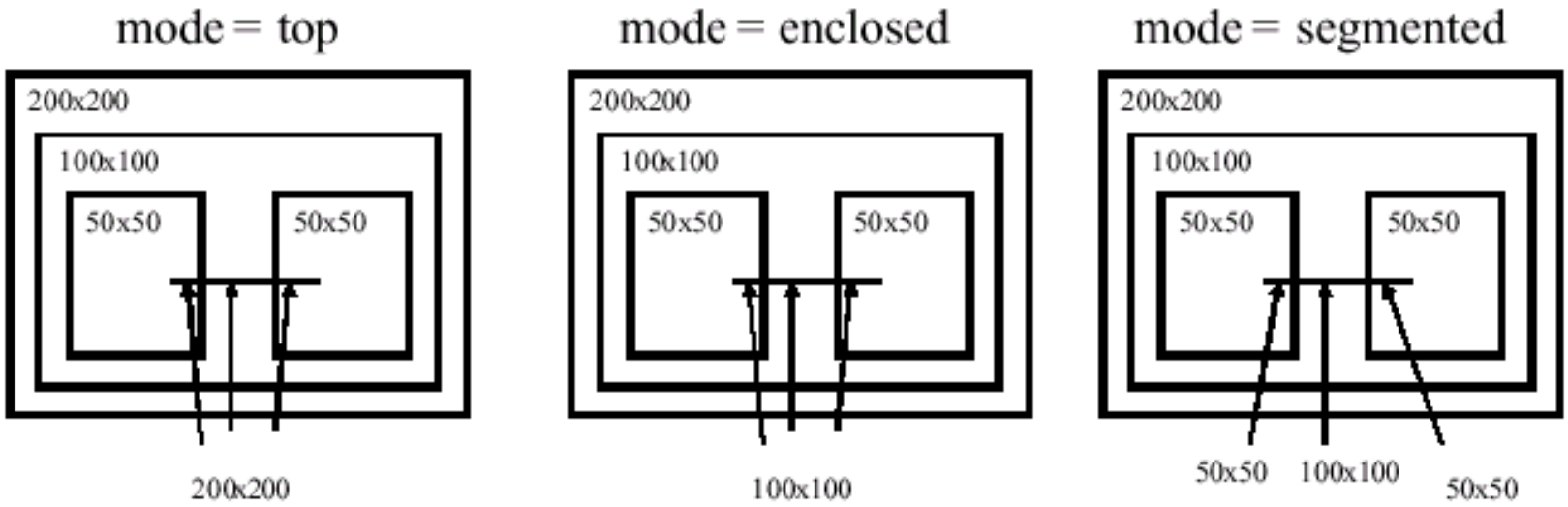


Setting Wire Load Model

- Wire load model estimates wire capacitance based on chip area & cell fanout
- Setting this information during compile in order to model the design more accurately
- *Attributes/Operating Environment/Wire Load*



Setting Wire Load Model (cont.)



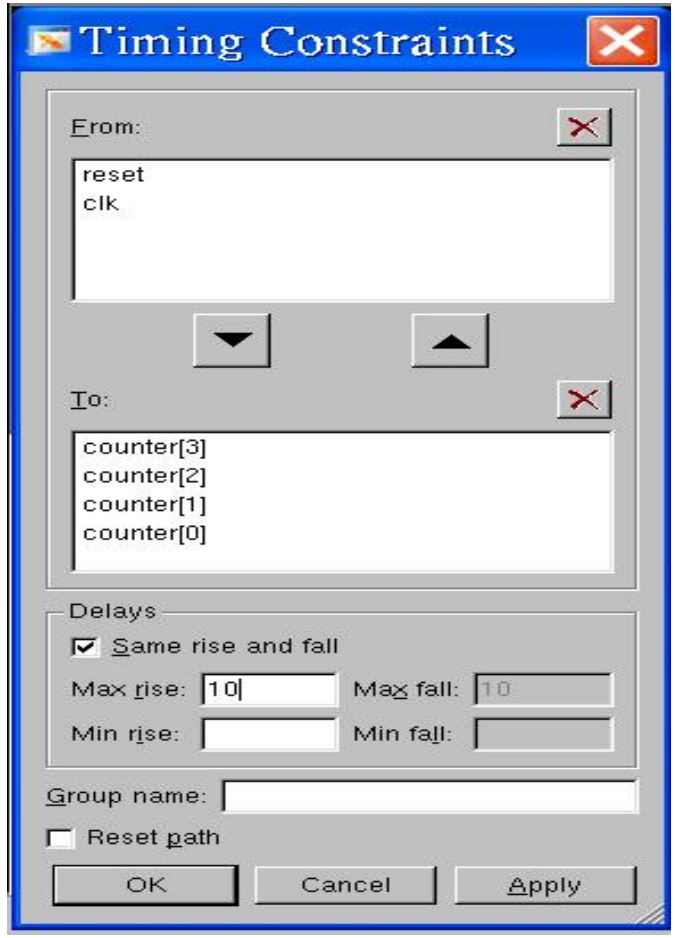
- Syntax: *set_wire_load wire_load_name –mode mode*
- Use –mode option, you can specify which wire load model to use for nets that cross hierarchical boundaries.

Digital System Design

Setting Design Constraints

Delay Constraints

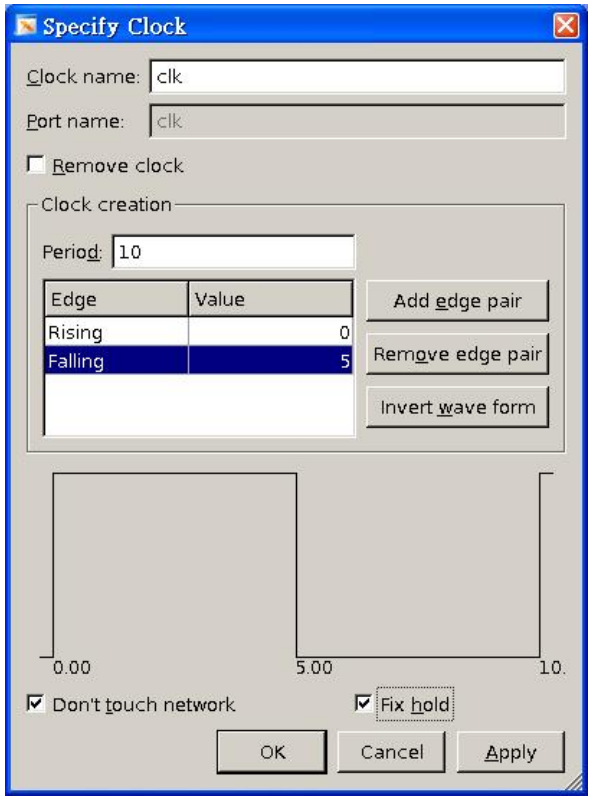
- For combinational circuits primarily
  - Select the start & end points of the timing path
  - *Attributes/Optimization Constraints/Timing Constraints*



set_max_delay 10 -from clock -to counter

Specify Clock Constrains

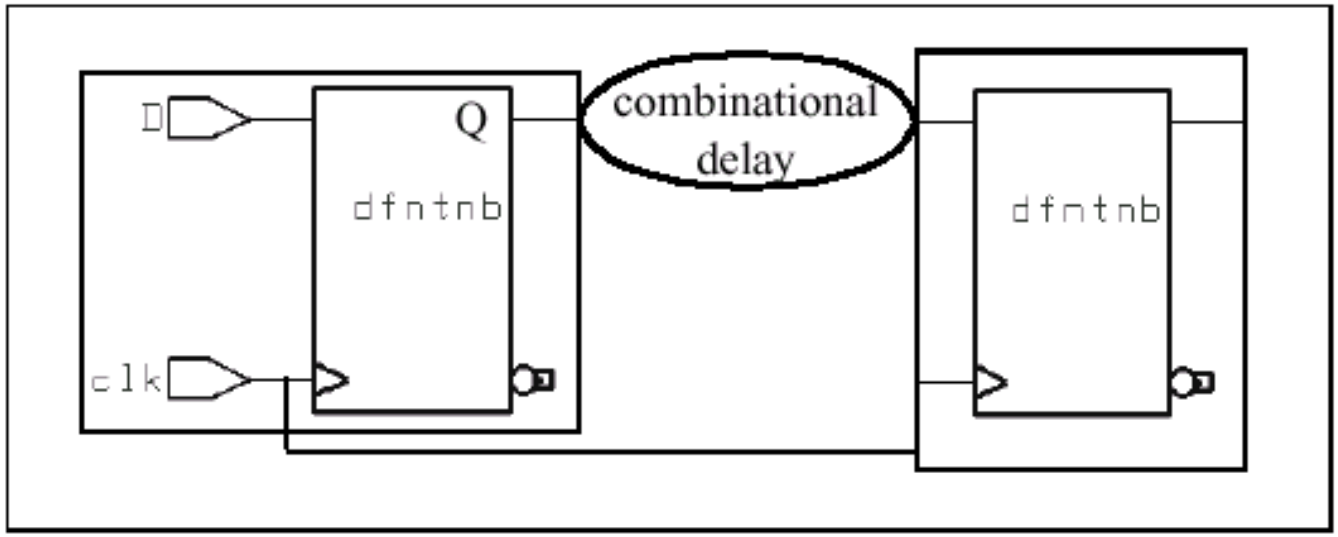
- Select clock port
- *Attributes/Specify Clock*



- create_clock : define your clock's waveform and the setup time /holdtime requirements of all clocked flip-flops
  - create_clock -name "clk" -period 10 -waveform { 0 5 } { clk }
- set_fix_hold : respect the hold time requirement of all clocked flip-flops
  - set_fix_hold [find clock clk]
- set_dont_touch_network : do not re-buffer the clock network
  - set_dont_touch_network [find clock clk]

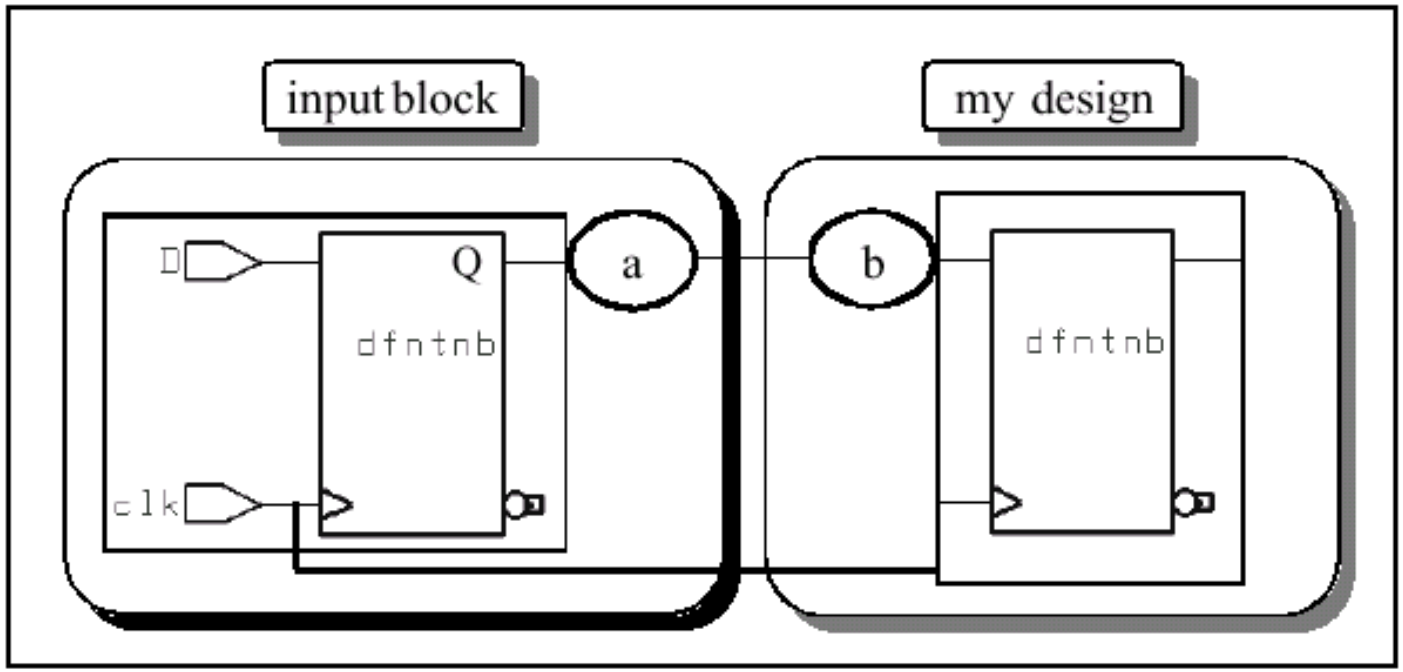
Sequential Circuit

- Sequential circuits are usually constrained by clock specify
- clock cycle >= $DFF_{clk\text{-}Qdelay}$ + combinational delay + DFF_{setup}



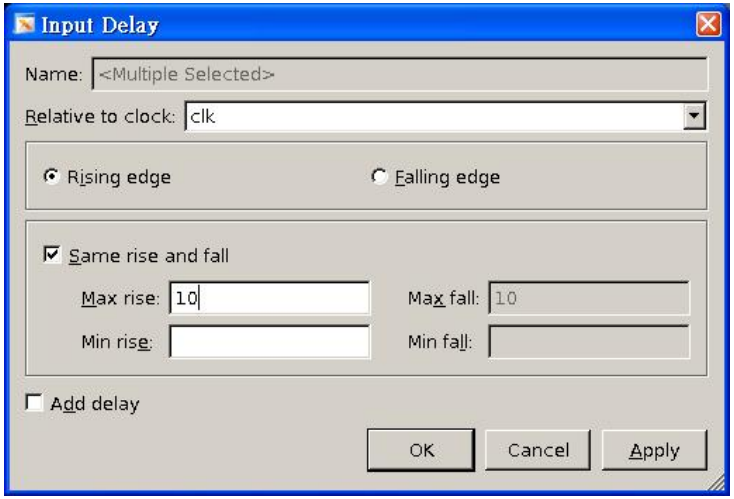
Input Delay Model

- Clock cycle >= DFFclk-Qdelay + a + b + DFFsetup
- Input delay = DFFclk-Qdelay + a

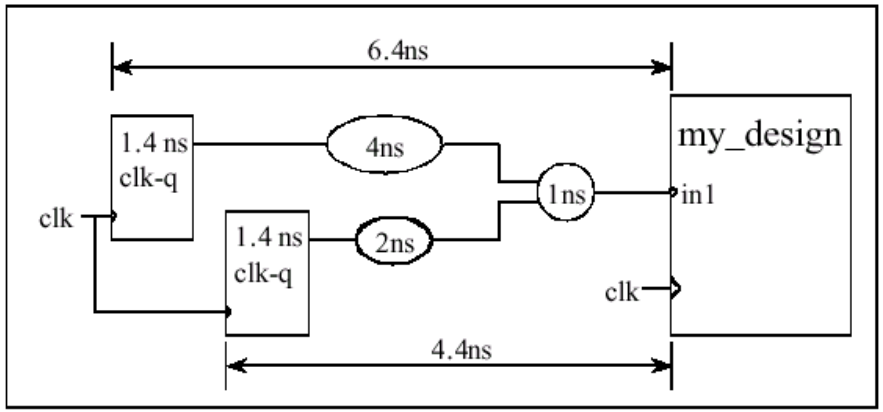


Setting Input Delay

- Select input ports
- Attributes/Operating Environment/Input Delay



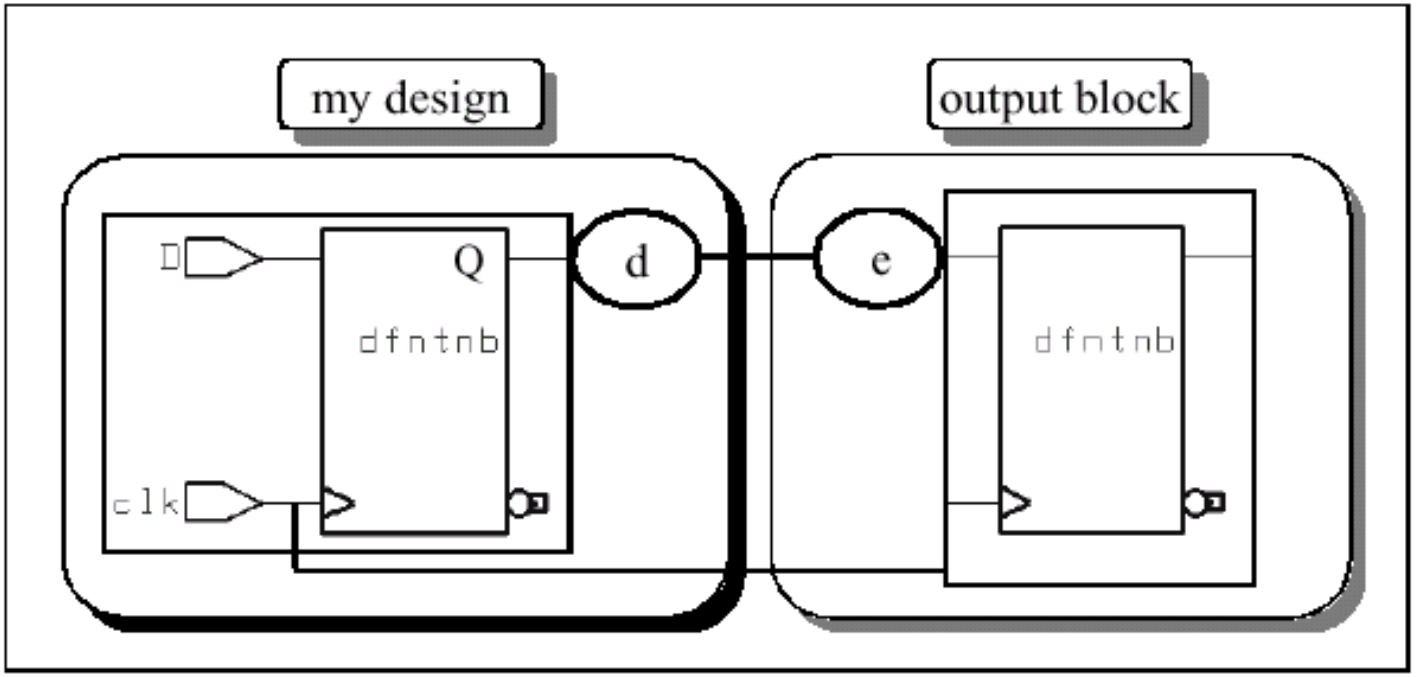
◆ Example




```
dc_shell>set_input_delay -clock clk -max 6.4 in1
dc_shell>set_input_delay -clock clk -min 4.4 in1
```

Output Delay Model

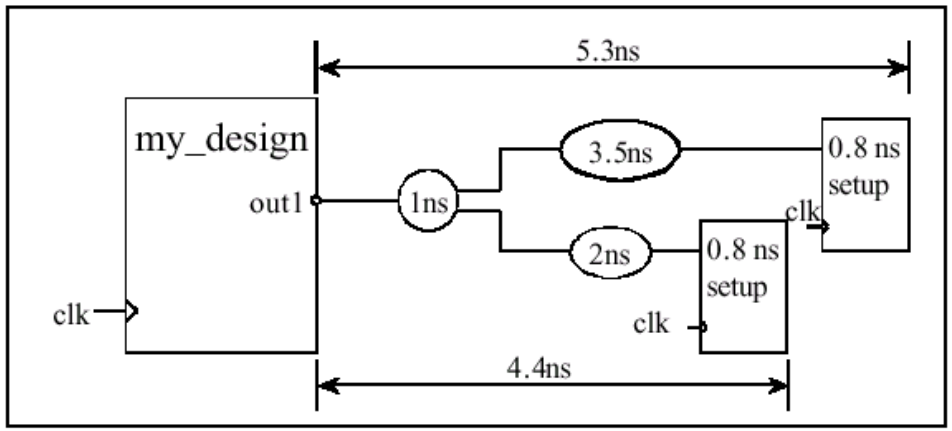
- clock cycle >= DFFclk-Qdelay + d + e + DFFsetup
- Output delay = e + DFFsetup



Setting Output Delay

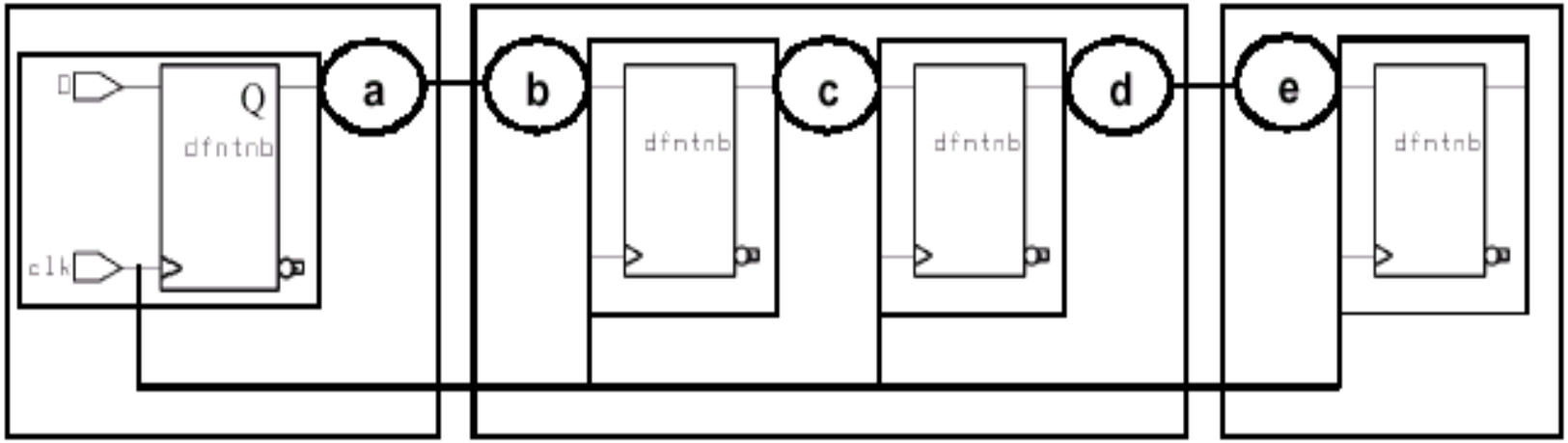
- Select output ports
- Attributes/Operating Environment/Output Delay

◆ Example




```
dc_shell>set_output_delay -clock clk -max 5.3 out1
```

What Have We Modeled ?



- Assume clock cycle = p
- Input delay $\tilde{=}$ a ; a + b < p
- Output delay = e ; d + e < p

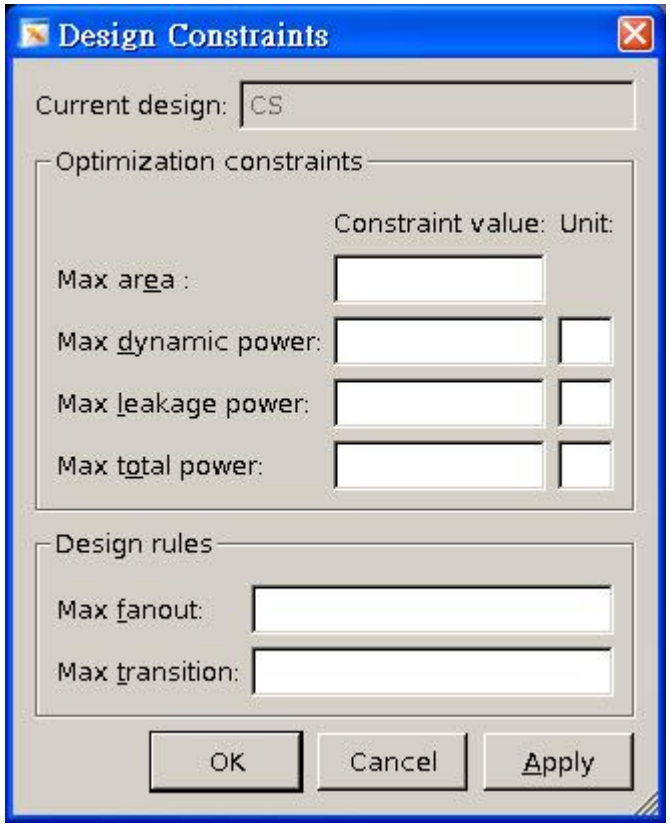
Design Rule Constraints

- 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:
  - *set_max_transition*
  - *set_max_fanout*
  - *set_max_capacitance*

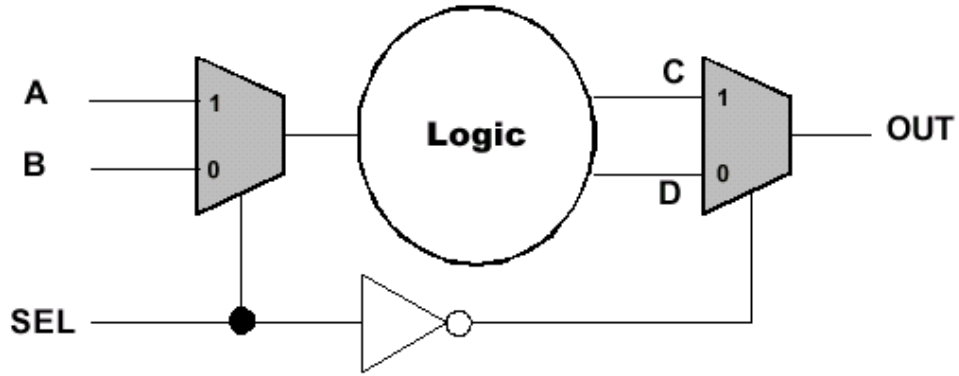
Setting Area Constraint

- *Attributes/OptimizationConstraints/Design Constraints*
- Area Unit : gate# or um^2 (follow unit defined in library)



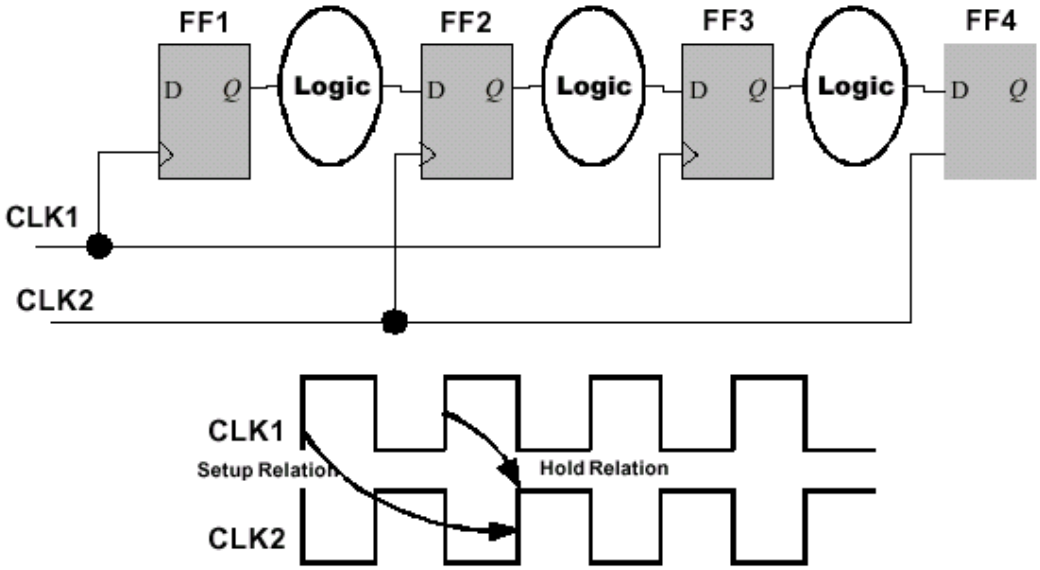
False Path

- A false path is a timing path that cannot propagate a signal, or a path we wish to ignore timing constraints.
- The *set_false_path* can be used to disable timing-based synthesis on a path-by-path basis
- 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}
```

Multicycle Path



- To define setup multiplier 2 for the path from FF1 to FF2 but maintain a hold multiplier 0, use command:

set_multicycle_path 2 -from FF1 -to FF2

Constraints Priority

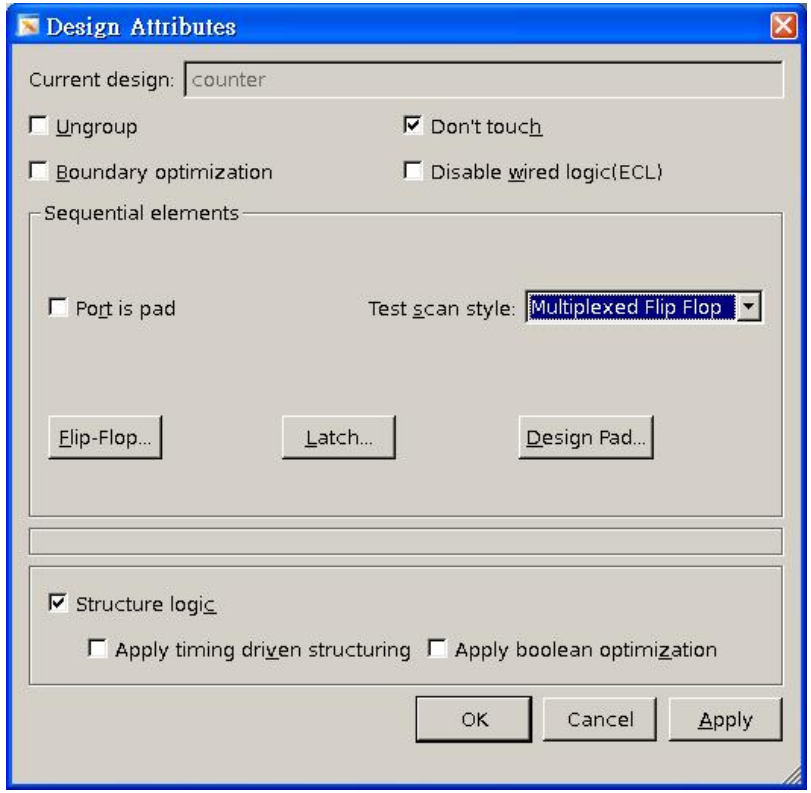
- During the optimization, there exists a constraint priority relationship.
  - Design Rule Constraint (max_transition, max_fanout, max_capacitance)
  - Timing constraint (max_delay, min_delay)
  - Power constraint
  - Area constraint
- Use set_cost_priority command to modify the order
  - set_cost_priority [-default] [-delay] [cost_list]

Check Design

- After you set up the deign attributes & design constraints, we recommend the next step is to check design
- *Analysis/Check Design*
- How to handle ?
  - dont_touch
  - ungroup
  - uniquify

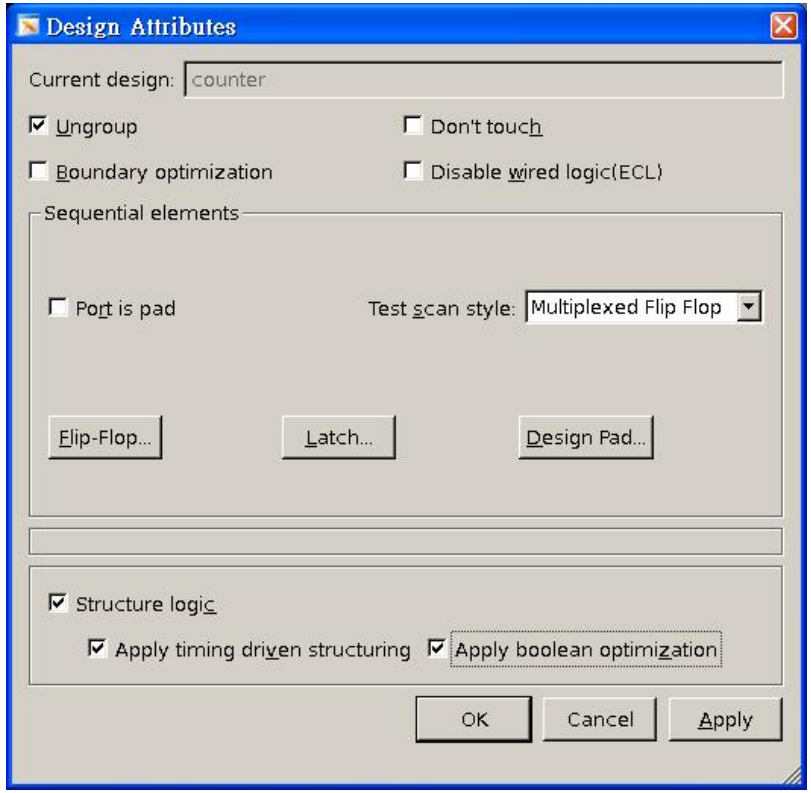
dont_touch

- Procedures
  - Constrain the block
  - Compile the block
  - Select the multiple design instances block
  - *Attributes/Optimization Directives/Design* & set the Don't Touch button
  - Compile the whole design using hierarchy compile



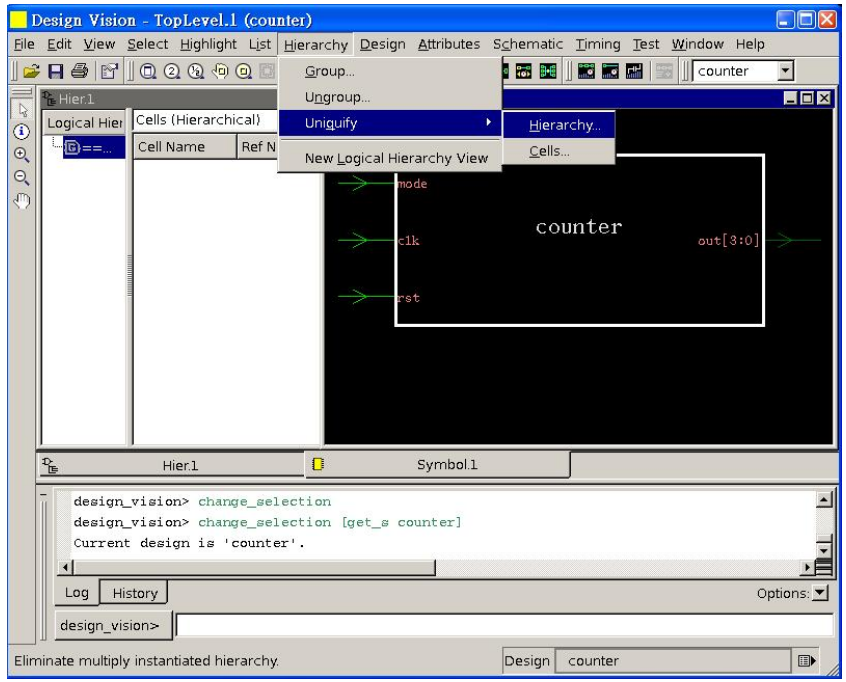
Ungroup

- Procedures
  - Select the multiple design instances block
  - *Attributes/Optimization Directives/Design* & set the Ungroup button
  - Compile whole design using hierarchy compile
- Remove a single level of hierarchy
- Does not preserve the hierarchy
- Take more memory
- Take more compile time



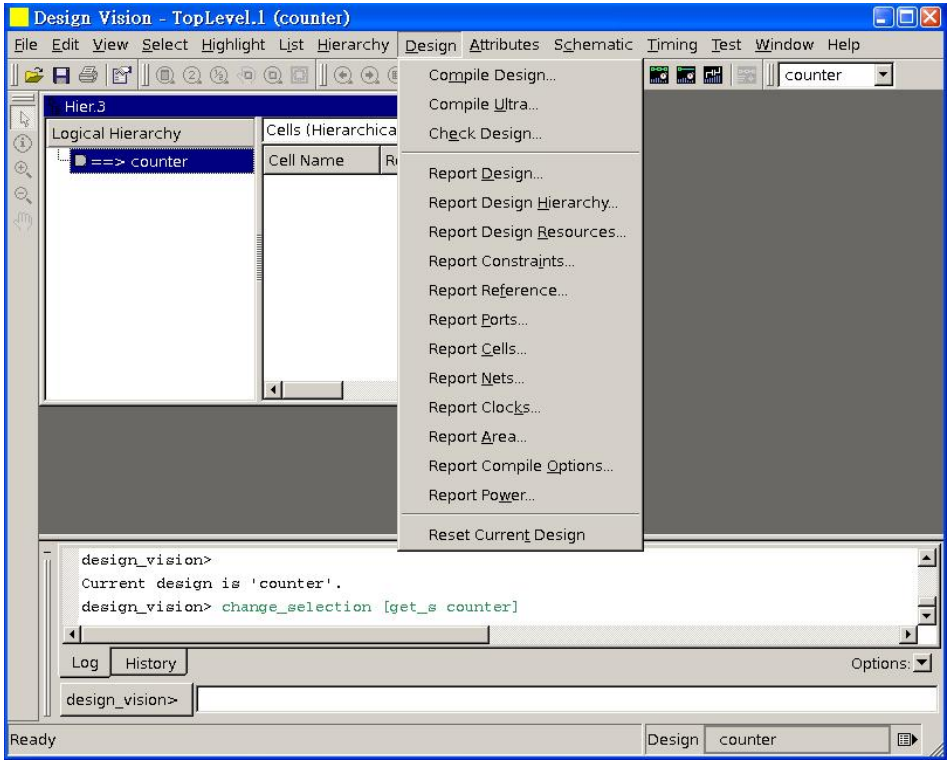
Uniquify

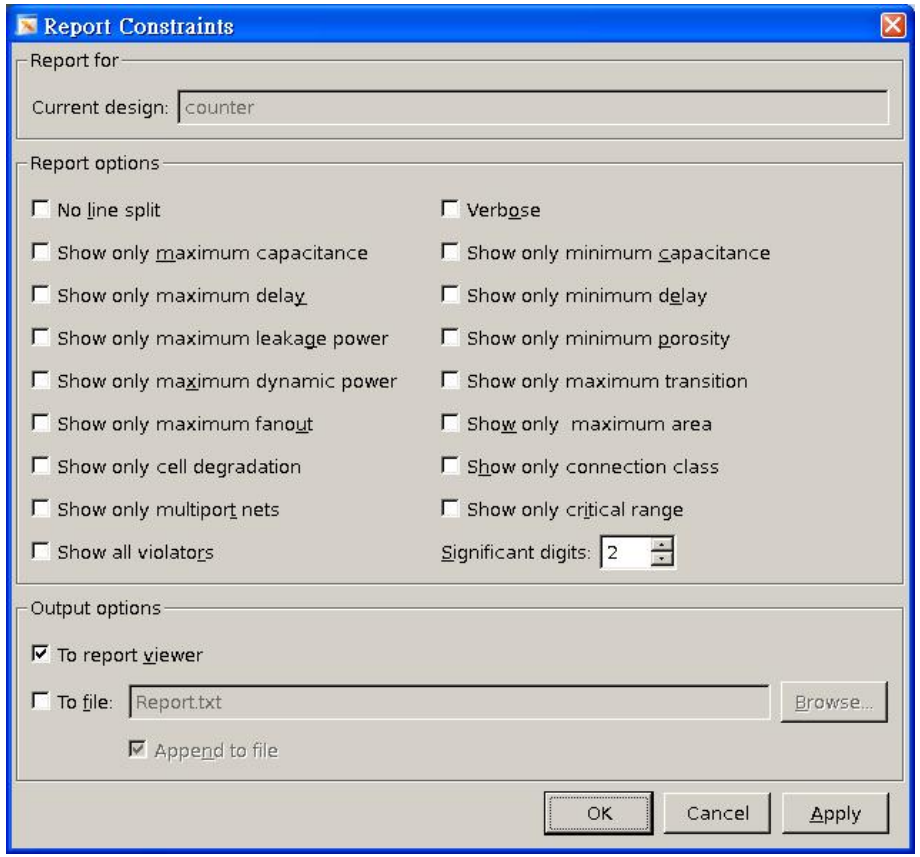
- Create a unique design file for each instance
- May select one cell or entire design hierarchy to be uniquify
- Allow design to be customized to its interface
- If the environment varies significantly, use *uniquify* rather than *compile+dont_touch*
- *Uniquify* uses more memory and cause longer compile time than *compile+dont_touch*
- Select the most top design of the hierarchy
- *Hierarchy/Uniquify/Hierarchy*



Check Constraints & Attributes

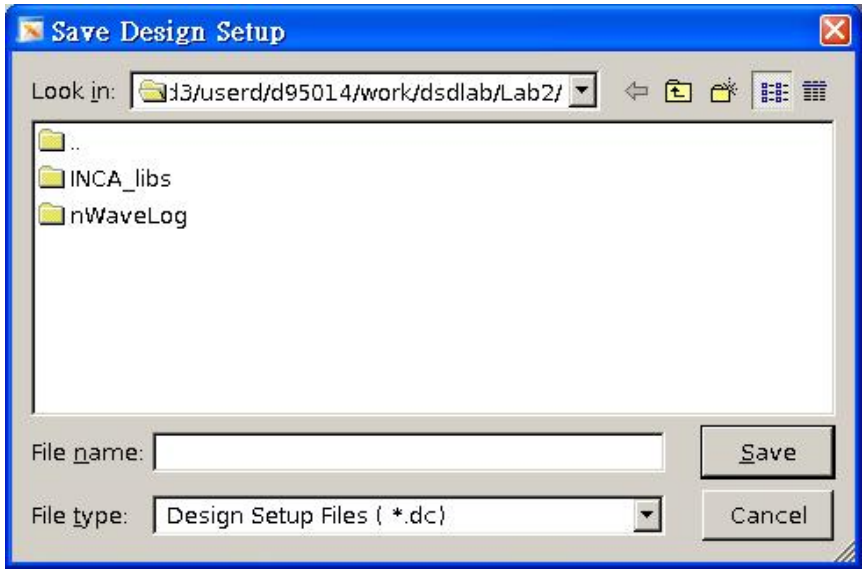
- Use the following reports to check constraints & attributes before compiling
- Analysis/Report





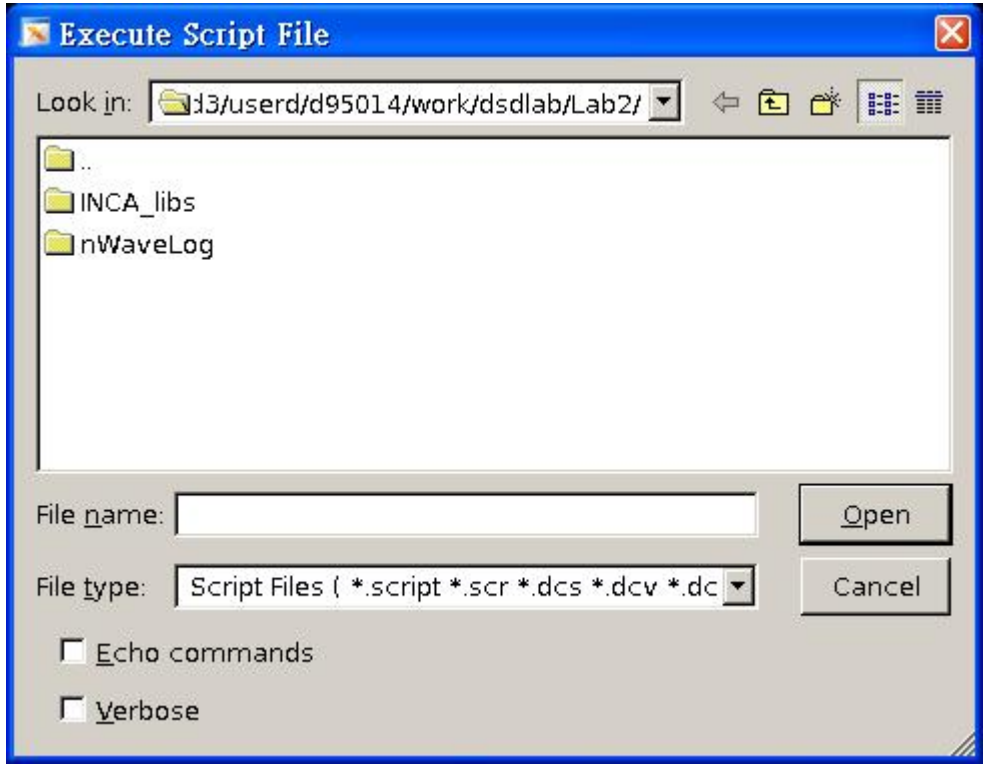
Save Constraints & Attributes

- Save attributes & constraints setting as the design setup file in TCL command format, use File/Save Info/Design Setup



Execute Script File

- Execute TCL command script file, use ***File/Execute Script***



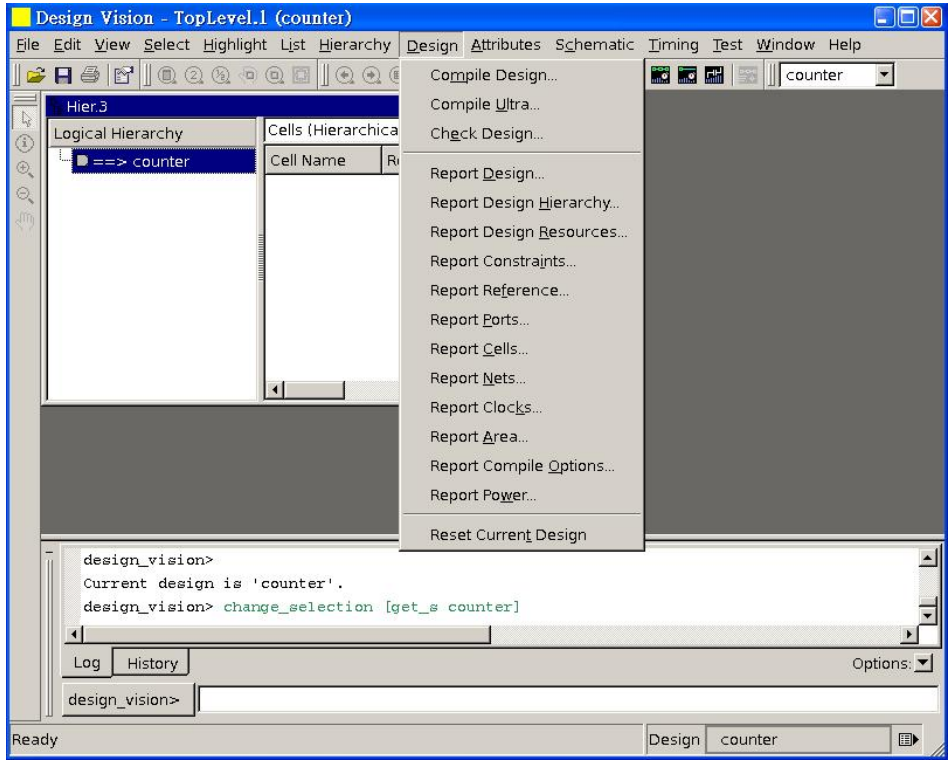
Graduate Institute of Electronics Engineering, NTU

Digital System Design

Synthesis Report and Analysis

Report

- *Analysis/Report*
  - From report and analysis, you can find the set attributes and the results after optimization
- Attribute reports
  - All attributes, clock, port, design, net
- Analysis reports
  - Area, hierarchy, constraints, timing, point timing



Timing Report

Report.2 - Timing

```
Startpoint: out_reg[0] (rising edge-triggered flip-flop clocked by clk)
Endpoint: out_reg[3] (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.00       0.00
out_reg[0]/CK (DFFRHQX1)                 0.00       0.00 r
out_reg[0]/Q (DFFRHQX1)                  0.34       0.34 r
U66/Y (INVX2)                            0.06       0.39 f
U64/Y (NAND2X1)                          0.12       0.52 r
U57/Y (OAI21X1)                          0.08       0.60 f
U70/Y (XOR2X1)                           0.19       0.79 r
U69/Y (AND3X2)                           0.12       0.91 r
U67/Y (OAI32X1)                          0.05       0.96 f
out_reg[3]/D (DFFRHQX1)                  0.00       0.96 f
data arrival time                                   0.96

clock clk (rise edge)                   10.00      10.00
clock network delay (ideal)              0.00      10.00
out_reg[3]/CK (DFFRHQX1)                 0.00      10.00 r
library setup time                      -0.24       9.76
data required time                                  9.76
-----------------------------------------------------------
data required time                                  9.76
data arrival time                                  -0.96
-----------------------------------------------------------
slack (MET)                                         8.80
```

Area Report

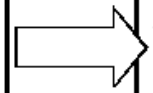
Report.3 - Area

```
****************************************
Report : area
Design : counter
Version: A-2007.12-SP4
Date   : Wed Apr  1 03:57:11 2009
****************************************

Library(s) Used:

    typical (File: /home/raid1_1/cic/CBDK018_UMC_Artisan/CIC/SynopsysDC/typical.db)

Number of ports:                7
Number of nets:                31
Number of cells:               28
Number of references:          18

Combinational area:        402.494410
Noncombinational area:     279.417603
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:           681.912012
Total area:                 undefined

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

Save Design

- Save your design in verilog format, run Verilog gate-level simulation, and we will use Verilog In interface to translate it into OPUS database for place & route
- If you can't read Verilog In, please check assign problem
- if there is any assignment problem, choose the block & use the dc_shell command as follow to fix it
  - set_fix_multiple_port_nets -all -buffer_constants
  - compile -map_effort medium

```
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];
```



```
buffda X37X ( .I(A[19]), .Z(ABSVAL[19]) );
buffd1 X38X ( .I(A[18]), .Z(ABSVAL[18]) );
buffd1 X39X ( .I(A[17]), .Z(ABSVAL[17]) );
buffd1 X40X ( .I(A[16]), .Z(ABSVAL[16]) );
buffd1 X41X ( .I(A[15]), .Z(ABSVAL[15]) );
```

Gate-Level Simulation (Verilog)

- Write out gate-level netlist

  File/Save As ➔ Verilog

  > write -hierarchy -format verilog -output design.vg

- Write out timing delay file

  File/Save Info ➔ Design timing **(not in current GUI)**

  > write_sdf –version 2.1 design.sdf

- Modify your testbench file to include timing delay

  $sdf_annotate ("SDF_FILE_NAME", top_module_instance_name);

- Gate level simulation with timing information:

  >> ncverilog design.vg testbench.v –v cell_model.v +access+r