Synthesis & Gate-Level Simulation

REF:
- TSMC 0.18um Process 1.8-Volt SAGE-X™ Stand Cell Library Databook, September, 2003
- TPZ973G TSMC 0.18um Standard I/O Library Databook, Version 240a, December 10, 2003
- Artisan User Manual
課程主題: Synthesis & Gate-level Netlist Simulation

學習目標

- 電路合成觀念
- Edit your synthesis script file
- Familiar with the Design Compiler
- Gate-level netlist simulation

LAB 2簡介 – 合成simple 8-bit microprocessor

由於合成的每個步驟，都可透過執行不同的script指令去完成，像是設定的各個constraints (ex: timing, area, etc…）。合成時，只需將script丟給Design Compiler即可完成合成的步驟。因此，學習編輯自己的script file將可大幅加速合成的步驟。

- 步驟一: 本實驗將會給定各個合成參數。只要依照參數去編輯script file並執行合成即可。
- 步驟二: 使用合成後的gate-level netlist跑simulation，並觀察波形。
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
What is Synthesis

- Synthesis = translation + optimization + mapping

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

The synthesis is constraint driven and technology independent!!

Advanced Reliable Systems (ARES) Lab.
Logic Synthesis Overview

RTL Design

HDL Compiler

Design Compiler

Design Library

Technology Library

Lib Compiler

DW Developer

Optimized Gate-Level Netlist

Architecture Optimization

Logic Optimization

Advanced Reliable Systems (ARES) Lab.
Compile

RTL code or netlist

Attributes & Constraints

Optimized Design (Gate-Level Netlist)

Schematic

Reports (Timing, Area, Power, ..., etc)

(Can be set by the GUI interface or user-defined Script File !!)

Technology Library

Flatten

Logic Level Optimization

Structure

Gate Level Optimization

Map

Technology Library

Advanced Reliable Systems (ARES) Lab.
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:

\[
\begin{align*}
f &= acd + bcd + e \\
g &= ae' + be' \\
h &= cde \\
(\text{Structure})
\end{align*}
\]

\[
\begin{align*}
f &= xy + e \\
g &= xe' \\
h &= ye \\
x &= a + b \\
y &= cd \\
(\text{Flatten})
\end{align*}
\]

\[
\begin{align*}
f0 &= at \\
f1 &= d + t \\
f2 &= t' e \\
t &= b + c
\end{align*}
\]

\[
\begin{align*}
f0 &= ab + ac \\
f1 &= b + c + d \\
f2 &= b' c' e
\end{align*}
\]
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

Advanced Reliable Systems (ARES) Lab.
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)
To meet the setup time requirement:

- \( T_{\text{require}} \geq T_{\text{arrival}} \)

Reg to Reg:

- \( T_{\text{arrival}} = T_{\text{clk1}} + T_{\text{DFF1}(\text{clk} \rightarrow \text{Q})} + T_{\text{PATH}} \)
- \( T_{\text{require}} = T_{\text{clk2}} - T_{\text{DFF2}(\text{setup})} \)
- \( T_{\text{slack}} = T_{\text{require}} - T_{\text{arrival}} \)

\( T_{\text{slack}} > 0 \) denotes “no timing violation”
Static Timing Analysis (Cont’)

- Setup Time

- PI to Reg

- $T_{\text{arrival}} = T_{\text{PI(delay)}} + T_{\text{PATH}}$

- $T_{\text{require}} = T_{\text{clk1}} - T_{\text{DFF1(setup)}}$

- $T_{\text{slack}} = T_{\text{require}} - T_{\text{arrival}}$
Reg to PO

- Setup Time

\[ T_{\text{arrival}} = T_{\text{clk}} + T_{\text{DFF1} (\text{clk} \rightarrow Q)} + T_{\text{PATH}} \]

\[ T_{\text{require}} = T_{\text{cycle}} - T_{\text{PO (output delay)}} \]

\[ T_{\text{slack}} = T_{\text{require}} - T_{\text{arrival}} \]
Static Timing Analysis (Cont’)

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

![Diagram showing PI, PATH, and PO with Tarrival, Trequire, Tslack, and Clk_source](image)
To meet the hold time requirement:

- \( T_{require} \leq T_{arrival} - \text{Hold Time} \)

**Reg to Reg**

- \( T_{arrival} = T_{clk1} + T_{DFF1}(\text{clk->Q}) + T_{PATH} \)
- \( T_{require} = T_{clk2} + T_{DFF2}(\text{hold}) \)
- \( T_{slack} = T_{arrival} - T_{require} \)
Static Timing Analysis (Cont’)

- Hold Time

- PI to Reg
  - $T_{arrival} = T_{PI\text{(delay)}} + T_{PATH}$
  - $T_{require} = T_{clk1} + T_{DFF\text{(hold)}}$
  - $T_{slack} = T_{arrival} - T_{require}$

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

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

- Create individual synopsys setup file for each folder

- **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
MEMs libraries are also included in this file

Ex:

```
set search_path "/usr1/teacher/jfl1/cell_lib/CBD018_TSMC_Artisan/CIC/SynopsysDC $search_path"
set search_path "/usr1/teacher/jfl1/cell_lib/CBD018_TSMC_Artisan/orig_lib/ac/sc/symbols/synopsys $search_path"
set search_path "/usr4/grad92/zvtseng/HOV/rom_base_bist/3_bit_bist/bist_256x8_with_ECC/0331_bist_FJU_ARTISAN/MEM $search_path"

RA1SHD256x8_fast0-40C_syn.db RA1SHD256x8_slow0-40C_syn.db RA1SHD256x8_ttypical_syn.db RA1SHD256x8_sslow_syn.db	
RA1SHD256x8_fast0-40C_syn.db RA1SHD256x8_fast00C_syn.db RA1SHD256x8_typical_syn.db RA1SHD256x8_sslow_syn.db	
RA1SHD256x8_fast0-40C_syn.db RA1SHD256x8_fast00C_syn.db RA1SHD256x8_typical_syn.db RA1SHD256x8_sslow_syn.db	

set link_library "typical.db fast.db slow.db tpz373gbc.db tpz373gwc.db dw_foundatioin.sldb dw01.slcb dw02.sldb dw03.sldb"
set target_library "typical.db fast.db slow.db"

set synthetic_library "dw_foundatioin.sldb dw01.slcb dw02.sldb dw03.sldb dw04.sldb dw05.sldb dw06.sldb dw07.sldb dw08.sldb"
```

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

- Memory block
  - Memory library files (synopsys model) are generated by memory compiler
  - Translate library files (.lib) to db files (.db) for synthesis
  - Four corners: fast@-40C, fast@0C, typical, and slow
Settings for Using Memory

- Convert *.lib to *.db
  - $dc\_shell$
  - $dc\_shell-t>\ read\_lib t13spsram512x32\_slow\_syn.lib$
  - $dc\_shell-t>\ write\_lib t13spsram512x32 -output t13spsram512x32\_slow\_syn.db$

- Modify <.synopsys\_dc.setup> File:
  - set link_library “* slow.db t13spsram512x32\_slow\_syn.db dw\_foundation.sldb”
  - set target_library “slow.db t13spsram512x32\_slow\_syn.db”

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

```vhdl
// include “sr\_memory\_1k.v”
module bisr\_mem(clk,rst,ams,CS,\_bisa\_mode,cmd\_done,BO,CS0,sh"
```

Advanced Reliable Systems (ARES) Lab.
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:

```markdown
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.

```
module CHIP(I_read_threshold, I_DI_S, I_CS_S, I_OE_S, I_bira_en, I_test done, O_bira_out_valid, O_addr_change, I_s0, 

input [3:0] I_read_threshold;
input [63:0] I_DI_S;
input [12:0] I_ADDR_S;
input [1:0] I_birr_mode;
input I_clk, I_rst, I_ams, ICSI, I_output O_cmd_done, O_BGO, O_CS0, O_S0, O_addr_change, O_s0;
wire [3:0] read_threshold;
wire [63:0] DI_S;
wire [12:0] ADDR_S;
wire [1:0] birr_mode;
wire clk, rst, amS, CSI, cmd_en, MEN;
wire cmd_done, BGO, CS0, shift_en, addr change, se;

brr_mem brr_mem( clk, rst, amS, CSI, O_E_S, D_L_S, ADDR_S, birr_en, shift_en, unrepair, done, bi

PDIDGZ PAD_CLK (.PAD(I_CLK), .CClk));
PDIDGZ PAD_rst (.PAD(I_rst), .CCrst));
PDIDGZ PAD_ams (.PAD(I_ams), .CCamS));
PDIDGZ PAD_CSI (.PAD(I_CSI), .CCCSI));
```

Ex: CORE.v

CHIP.v

"brr_mem.v"

Advanced Reliable Systems (ARES) Lab.
How To Choose the IO 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:

<table>
<thead>
<tr>
<th>Cell Name</th>
<th>Path</th>
<th>Parameter</th>
<th>Group1</th>
<th>Group2</th>
<th>Group3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD01CDG</td>
<td>I→E-PAD</td>
<td>hI</td>
<td>tSSN</td>
<td>2.127<em>0.2194</em>Closed</td>
<td>3.516<em>0.2170</em>Closed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tH</td>
<td></td>
<td>2.094<em>0.2170</em>Closed</td>
<td>2.094<em>0.2170</em>Closed</td>
</tr>
</tbody>
</table>

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

1. Design Import
2. Setting Design Environment
3. Setting Clock Constraints
4. Setting Design Rule Constraints
5. Compile the Design

6. DFT Insertion
7. Compile After DFT
8. Assign Violation Avoidance
9. Naming Rule Changing
10. Save Design
Getting Started

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

- 

```
linux %> dv
```

(GUI view of the Design Vision)
Read File

- 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

```bash
read_file -format verilog file name.v (/usr/LAB_DV/syn/cpu.v)
read_file -format ddc file name.ddc
```

Advanced Reliable Systems (ARES) Lab.
PAD Parameters Extraction

- Input PAD
  - Input delay
  - Input driving

- Output PAD
  - Output delay
  - Output loading

```
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD

set_load -pin_load 0.00132 [get_ports {cmd_done}]
set_load -pin_load 0.00132 [get_ports {GCO}]
set_load -pin_load 0.00132 [get_ports {CSO}]
set_load -pin_load 0.00132 [get_ports {shift_en}]
set_load -pin_load 0.00132 [get_ports {unspecified}]
set_load -pin_load 0.00132 [get_ports {done}]
set_load -pin_load 0.00132 [get_ports {bira_out_valid}]
set_load -pin_load 0.00132 [get_ports {addr_change}]
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_driving_cell -1ib_cell P010GZ -1library tpz973gbc -pin C -from_pin PAD
set_load -pin_load 0.00132 [get_ports {scantest}]
```

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

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

**Design View**

<table>
<thead>
<tr>
<th>Name</th>
<th>Design Area</th>
<th>Don't Touch</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>ROM</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SES_ID</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_0</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_1</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_2</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_3</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_4</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_5</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_6</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>SYN_DEC_8_7</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>addr_present</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>addr_previous1</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>addr_previous2</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>b_to_g_0</td>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>b_to_g_1</td>
<td>0</td>
<td>undefined</td>
</tr>
</tbody>
</table>

**Log Window**

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'.

uniquify

Advanced Reliable Systems (ARES) Lab.
Design Environment

- Setting Operating Environment
- Setting Don’t Used Cells
- Setting Input Driving Strength
- Setting Output Loading
- Setting Input/Output Delay
- Setting Wire Load Model
Setting Operating Condition

- Attributes/Operating Environment/Operating Conditions

```
set_operating_conditions -max "slow" -max_library "slow" -min "fast" -min_library "fast"
```

Setup/Hold time is evaluated
Setting Don’t Used Cells

- Some standard cells are for special purposes
  - Buffering clock
    - CLKBUF*
  - Inverting clock
    - CLKINV*
  - Creating delay (delay cell)
  - Tie cells
    - TIE*
  - DFT cells
    - SDF* and SEDF*

```
set_dont_use slow/CLKBUF*
set_dont_use typical/CLKBUF*
set_dont_use fast/CLKBUF*
```
Assume that we use the input PAD “PDIDGZ”

```
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”

```text
set_load [expr 0.06132] [all_outputs]
set_output_delay [expr 2] [all_outputs]
```
Setting Wire Load Model

- Attributes/Operating Environment/Wire Load

```bash
set_wire_load_model -name "tsmc18_wl10" -library "slow"
set_wire_lode_mode "top"
```

Worst Case

- Recommend

Advanced Reliable Systems (ARES) Lab.
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}
set sys_clk clk1
creat_clock -name $sys_clk -period 10 $sys_clk
set_dont_touch_network $sys_clk
set_fix_hold $sys_clk
```
Setting Clock Skew

- Different clock arrival time

Ex:

- experience
  - Small circuit: 0.3 ns
  - Large circuit: 0.5 ns

(Command Line)

- `set_clock_uncertainty -setup 0.6 $sys clk`
- `set_clock_uncertainty -hold 0.5 $sys clk`

(Timing Report)

Advanced Reliable Systems (ARES) Lab.
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 avoided if the design is without the clock generator.

Ex:

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

```
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

```bash
# Command Line
set_ideal_network [get_clocks clk]
```
Setting Don’t Touch Macro

- Modules have been synthesized/optimized
- Set dont_touch to avoid optimize the macro with other modules

{ Command Line }

set_dont_touch module_name
Setting Clock Transition

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

```bash
set_input_transition -max 0.1 $sys_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:
- Maximum Delay Constraint (5ns = 200 MHz)
- Minimum Delay Constraint
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

```
set_max_area 0
set_max_fanout 32 [get_designs CORE]
```
Compile the Design

Design/Compile Design

compile -map_effort high -boundary_optimization
compile ultra -no_autoungroup (time consuming)
Example for DFT Insertion

...

### 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 -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###
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

{ Command Line }

compile -scan
check_scan
report_test -scan_path
estimate_test_coverage

☐ The fault coverage will be shown as below:

<table>
<thead>
<tr>
<th>Uncollapsed Stuck Fault Summary Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>fault class</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Detected</td>
</tr>
<tr>
<td>Possibly detected</td>
</tr>
<tr>
<td>Undetectable</td>
</tr>
<tr>
<td>ATPG untestable</td>
</tr>
<tr>
<td>Not detected</td>
</tr>
</tbody>
</table>

| total faults      | 27630 |
| test coverage     | 92.37%|
| fault coverage    | 90.78%|

Pattern Summary Report

<table>
<thead>
<tr>
<th>#internal patterns</th>
<th>227</th>
</tr>
</thead>
<tbody>
<tr>
<td>#basic_scan patterns</td>
<td>227</td>
</tr>
</tbody>
</table>
The syntax of “assign” may cause problems in the LVS

```vhdl
assign \A[18] = A[18];
assign \A[16] = A[16];
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];
```

```vhdl
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]) );
```

```bash
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

- 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

```bash
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

- 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
  - *.ddc: binary file (i.e. all the constraints and synthesis results are recorded)

```
REPORT   WORK   syn   tbench   script

{ Command Line }

write -f ddc -o [format "%s%s" [format "%s%s" "./WORK/$TOP_BLOCK" ".ddc"] -no_implicit -hier
write -f verilog -o [format "%s%s" [format "%s%s" "./WORK/$TOP_BLOCK" "_syn.v"] -no_implicit -hier
write_sdc [format "%s%s" [format "%s%s" "./WORK/$TOP_BLOCK" ".sdc"]
write_sdf -version 2.1 -context verilog [format "%s%s" [format "%s%s" "./WORK/
"$TOP_BLOCK"] ".sdf"]
```

Advanced Reliable Systems (ARES) Lab.
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:

```
***********
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: hierarchy</td>
</tr>
<tr>
<td>Design: bisr_mem</td>
</tr>
<tr>
<td>Version: X-2005.09-SP4</td>
</tr>
<tr>
<td>Date: Fri Jul 27 14:53:58 2007</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Information: This design contains unmapped logic. (RPT-7)</td>
</tr>
<tr>
<td>bisr_mem</td>
</tr>
<tr>
<td>GTECH_OR2  gtech</td>
</tr>
<tr>
<td>bisr</td>
</tr>
<tr>
<td>GTECH_AND2  gtech</td>
</tr>
<tr>
<td>GTECH_BUF  gtech</td>
</tr>
<tr>
<td>GTECH_NOT  gtech</td>
</tr>
<tr>
<td>bisr_top</td>
</tr>
<tr>
<td>GTECH_AND2  gtech</td>
</tr>
<tr>
<td>GTECH_NOT  gtech</td>
</tr>
<tr>
<td>GTECH_OR2  gtech</td>
</tr>
<tr>
<td>bitmap</td>
</tr>
<tr>
<td>GTECH_AND2  gtech</td>
</tr>
<tr>
<td>GTECH_AND3  gtech</td>
</tr>
<tr>
<td>GTECH_AND4  gtech</td>
</tr>
<tr>
<td>GTECH_BUF  gtech</td>
</tr>
<tr>
<td>GTECH_NOT  gtech</td>
</tr>
<tr>
<td>GTECH_OR2  gtech</td>
</tr>
</tbody>
</table>
```

Advanced Reliable Systems (ARES) Lab.
### Design/Report Area

**Ex:**

<table>
<thead>
<tr>
<th>Library(s) Used:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gtech (File: /usr/cad/synopsys/synthesis/cmu/libraries/syn/gtech.db)</td>
<td></td>
</tr>
<tr>
<td>USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/MBM/MB/memory_8k_32_fast@-40C_syn.db)</td>
<td></td>
</tr>
<tr>
<td>USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/MBM/MB/sc_memory_fsst@-40C_syn.db)</td>
<td></td>
</tr>
<tr>
<td>USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/MBM/MB/sr_memory_fsst@-40C_syn.db)</td>
<td></td>
</tr>
</tbody>
</table>

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

**Combinational area:** 0.000000 (um²)

**Noncombinational area:** 3271507.000000 (um²)

**Net Interconnect area:** undefined (No wire load specified)

**Total cell area:** 3271507.000000

**Total area:** undefined

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

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

***** End Of Report *****
Design View

**List/Design View**

- All the block area are listed!!

```plaintext
Ex:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Type</th>
<th>Type</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>array_or</td>
<td>851.558</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bitmap</td>
<td>15501</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bisr_mem</td>
<td>3.34175e+06</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>memory_8k_</td>
<td>3.29505e+06</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>aru</td>
<td>7717.25</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>remapping_</td>
<td>4440.74</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>address_oe_</td>
<td>8276.08</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>finj</td>
<td>1203.97</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bsr</td>
<td>46636</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bira_top</td>
<td>26953.8</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>remap</td>
<td>8016.62</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>remap_DW_</td>
<td>132.331</td>
<td>undefined</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>multi_bit</td>
<td>1593.35</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bitmap_DW_</td>
<td>83.16</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>fsm</td>
<td>1816.21</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>blst</td>
<td>17413.7</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>tpg</td>
<td>15151.8</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ADDR</td>
<td>2421.62</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ADDR_DW_</td>
<td>435.758</td>
<td>undefined</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>ADDR_DW_</td>
<td>472.349</td>
<td>undefined</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>CMP</td>
<td>9433.87</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>DATA</td>
<td>219.542</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>DECO</td>
<td>804.989</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ROM</td>
<td>1147.61</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ctr</td>
<td>2261.95</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
</tbody>
</table>
```

*Advanced Reliable Systems (ARES) Lab.*
Report Timing

- **Timing/Report Timing**

Ex:

setup time

Critical Path

max: setup time

min: hold time

Slack = Data Require Time – Data Arrival Time

***** End Of Report *****
Critical Path Highlighting

- View/Highlight/Critical Path

Ex:

Advanced Reliable Systems (ARES) Lab.
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**

```bash
# This is an example of a script file
# It sets up a clock with a period of 10ns and a clock skew of 0.3ns
# It also sets the output delay and drive to specific values.

# setup
set clock_period 10
set clock_skew 0.3

# input delay
set_input_delay [expr 0.34] -clock clk [all_inputs]
set_input_delay [expr 0.34] -clock clk [get_ports clk]

# output delay
set_output_delay [expr 0.34] -clock clk [all_outputs]

# drive
set_drive [expr 0.2880001] [all_inputs]
```

Advanced Reliable Systems (ARES) Lab.
Gate-Level Simulation

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

```verilog
include "tmc18.v"
include "bistr_mem.vg"
include "memory_8k_32.v"
include "sc_memory.v"
include "sr_memory.v"
```

- Add the following Synopsys directives to the test bench

```verilog
initial begin
$fsdbDumpfile("bistr_mem.fsdb");
$fsdbDumpfile("bistr.fsdb");
$fsdbDumpVars;
end
```

```verilog
initial begin
$sdf_annotate("bistr_mem.sdf",bistr_mem);
end
```

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

- Initial

```verilog
Initial
```

```verilog
Clock
```

*Delay*

- Advanced Reliable Systems (ARES) Lab.
LAB