Training Course of Design Compiler

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

Speaker: T. –W. Tseng

Advanced Reliable Systems (ARES) 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

**System Level**
- MATLAB/ C/ C++/ System C/ ADS/ Covergen (MaxSim)
- Memory Generator

**RTL Level**
- Verilog/ VHDL
- NC-Verilog/ ModelSim
  - Debussy (Verdi)/ VCS
- Design/ Power Compiler
  - DFT Compiler/ TetraMAX
  - NC-Verilog/ ModelSim
    - Debussy (Verdi)/ VCS
- SOC Encounter/ Astro
- GDS II
- DRC/ LVS (Calibre)
- PVS: Calibre xRC/ NanoSim
  - (Time/ Power Mill)

**Logic Synthesis**

**Design for Test**

**Gate Level**

**Layout Level**

**Post-Layout Verification**

**Advanced Reliable Systems (ARES) Lab.**
What is Synthesis

- Synthesis = translation + optimization + mapping

```python
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

 Architecture Optimization

 HDL Compiler

 Design Compiler

 Lib Compiler

 Logic Optimization

 DesignWare Library

 Technology Library

 Optimized Gate-Level Netlist

 Advanced Reliable Systems (ARES) Lab.
Compile

RTL code or netlist → Compile → Optimized Design (Gate-Level Netlist)

Attributes & Constraints → Compile → Schematic

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

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

Technology Library

Flatten Technology Library

Structure Technology Library

Logic Level Optimization

Gate Level Optimization

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 \\

  f &= xy + e \\
  g &= xe' \\
  h &= ye \\
  x &= a + b \\
  y &= cd
\end{align*}
\]

(Structure)

\[
\begin{align*}
  f_0 &= at \\
  f_1 &= d + t \\
  f_2 &= t'e \\
  t &= b + c
\end{align*}
\]

(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**

- **a** → **c**
- **a** → **c**
- **a** → **c**
- **a** → **c**

**Sequential Mapping**

- **AND_FF**
- **Loop_FF**

(assume g loading high)
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:

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

(T\(_{\text{slack}}\) > 0 denotes “no timing violation”)

Advanced Reliable Systems (ARES) Lab.
Static Timing Analysis (Cont’)

**PI to Reg**

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

Advanced Reliable Systems (ARES) Lab.
Reg to PO

- \( T_{\text{arrival}} = T_{\text{clk1}} + T_{\text{DFF1}(\text{clk} \rightarrow \text{Q})} + T_{\text{PATH}} \)
- \( T_{\text{require}} = T_{\text{cycle}} - T_{\text{PO}(\text{output delay})} \)
- \( T_{\text{slack}} = T_{\text{require}} - T_{\text{arrival}} \)
Static Timing Analysis (Cont’)

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

- Setup Time

Advanced Reliable Systems (ARES) Lab.
Static Timing Analysis (Cont’)

- Hold Time

To meet the hold time requirement:

- $T_{require} \leq T_{arrival}$

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

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

- PI to PO
  - $T_{\text{arrival}} = T_{\text{PI(delay)}} + T_{\text{PATH}}$
  - $T_{\text{require}} = -T_{\text{PO(output delay)}}$
  - $T_{\text{slack}} = T_{\text{arrival}} - T_{\text{require}}$
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
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 "/usr/cad/synopsys/synthesis/cur/libraries/syn $search_path"
set search_path "/usr1/teacher/jfl1/cell_lib/CBD018_TSMC_Artisan/orig_lib/ac/FL/sc/symbols/synopsys $search_path"
set search_path "/usr1/grad92/zvtseng/H0V/rom_base_bist/3_bit_bist/bist_256x8_with_ECC/0331_bist_FJ_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"
set target_library "RA1SHD256x8 Fast@-40C.syn.db RA1SHD256x8 Fast@0C.syn.db RA1SHD256x8 Typical.syn.db RA1SHD256x8 Slow.syn.db"

### without MEM ####
set link_library "typical.db fast.db slow.db tpz973qgc.db tpz973qtc.db dw_base.sld db slcb dw20.sld db dw03.sld"
set target_library "typical.db fast.db slow.db"

set symbol_library "tscm18.sdb"
set synthetic_library "dw_base.sld dw01.sld db dw02.sld dw03.sld dw04.sld dw05.sld dw06.sld dw07.sld dw08.sld"
```

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_slow_syn.db`

- Modify <.synopsys_dc.setup> File:
  - `set link_library "* slow.db t13spsram512x32_slow.db dw_foundation.sldb"`
  - `set target_library "slow.db t13spsram512x32_slow.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,CSI,bisr_mode,cmd_done,BGO,CSO,sh)
  ...
  parameter WORD_LENGTH = 64;
  parameter ADR_LEN = 16;
  ...
```
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:

```vhdl
input [WORD_LENGTH-1:0] DI_S;
input [ADDR_LEN-1:0] ADDR_S;
input bira_en;
input test_done;
input [1:0] bissr_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

---

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

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

Design Import
- Setting Design Environment
- Setting Clock Constraints
- Setting Design Rule Constraints
- Compile the Design

DFT Insertion
- Compile After DFT
- Assign Violation Avoidance
- Naming Rule Changing
- Save Design
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)
```
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

```
read_file -format verilog file name
```

Advanced Reliable Systems (ARES) Lab.
### PAD Parameters Extraction

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

```
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports [ADDR_SDO]]
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports drive_en]
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports test_done]
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports { ISR_model[13]}]
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports { ISR_model[0]}]
set_load -pin_load 0.00132 [get_ports {cmd_done}]
set_load -pin_load 0.00132 [get_ports { RDIS}]
set_load -pin_load 0.00132 [get_ports { CS0}]
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 {drive_out_valid}]
set_load -pin_load 0.00132 [get_ports {ADDR_CHANGE}]
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports {i3}]
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports { rst}]
set_driving_cell -lib_cell F010GZ -library tpz973gbc -pin C -from_pin PAD
-no_design_rule [get_ports {scantest}]
set_load -pin_load 0.00132 [get_ports {so}]
```

---

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

---

Advanced Reliable Systems (ARES) Lab.
Uniquify

- Select the most top design of the hierarchy

Hierarchy/Uniquify/Hierarchy

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

(Design View) (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

{ Command Line }
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**

Setup/Hold time is evaluated

- **set_operating_conditions** –max “slow” –max_library “slow” –min “fast” 
  -min_library “fast”
Setting Drive Strength/Input Delay for PADs

- Assume that we use the input PAD “PDIDGZ”

```command
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”

```bash
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_load_mode "top"
  ```

(Worst Case) Recommend
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 }

```sh
create_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_6k_64_2r_2c/sru/U488/Y (MX14X1)       0.40      10.69 r
memory_6k_64_2r_2c/sru/U568/Y (N0E2X1)       0.09      10.77 f
memory_6k_64_2r_2c/sru/del4_44_ent0 (szu)     0.00      10.77 f
memory_6k_64_2r_2c/sc_memory/D15 (sc_memory)  0.00      10.77 f
```

- Set clock uncertainty 0.1
  - `set_clock_uncertainty 0.1 [get_clocks clk]`

Experience

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

(Timing Report)
**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.

**Example:**

<table>
<thead>
<tr>
<th>Your Design</th>
<th>Origin of Clock</th>
<th>Source Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3ns</td>
<td></td>
</tr>
</tbody>
</table>

- Experience:
  - 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.

```
set_ideal_network [get_clocks clk]
```
Setting Clock Transition

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

```bash
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:

- 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 50 [get_designs CORE]
```
Compile the Design

Design/Compile Design

{ Command Line }

```
compile -map_effort high -boundary_optimization
```
### DB mode ###
```bash
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

```bash
{ 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
```

Advanced Reliable Systems (ARES) Lab.
Assign Problem

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

```vhdl
assign \(\text{ABSVAL}[19] = A[19];\)
assign \(\text{ABSVAL}[18] = A[18];\)
assign \(\text{ABSVAL}[17] = A[17];\)
assign \(\text{ABSVAL}[16] = A[16];\)
assign \(\text{ABSVAL}[15] = A[15];\)

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

```bash
remove_unconnected_ports -blast_buses [get_cells -hierarchical *]
```
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

```
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
```
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 : 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
      └─ bisr
          └─ GTECH_AND2
              └─ GTECH_BUF
                  └─ GTECH_NOT
                      └─ GTECH_AND2
                          └─ GTECH_AND3
                              └─ GTECH_AND4
                                  └─ GTECH_BUF
                                      └─ GTECH_NOT
                                          └─ GTECH_OR2
```

---

Advanced Reliable Systems (ARES) Lab.
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:
qtech (File: /usr/cad/synopsys/synthesis/cur/libraries/syn/qtech.db)
USERLIB (File: /usr4/grad92/zwtseg/dv_training/RTL/MEM/DB/memory_8k_32_fast-40C_syn.db)
USERLIB (File: /usr4/grad92/zwtseg/dv_training/RTL/MEM/DB/sc_memory_fsst-40C_syn.db)
USERLIB (File: /usr4/grad92/zwtseg/dv_training/RTL/MEM/DB/sr_memory_fsst-40C_syn.db)

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

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

Advanced Reliable Systems (ARES) Lab.
### List/Design View

**Ex:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>type1</th>
<th>type2</th>
<th>type3</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>bsr_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_6e</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>46836</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bira_top</td>
<td>26955.8</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>192.331</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>83.18</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>bist</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>9433.67</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>804.889</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>2251.95</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
</tbody>
</table>
**Report Timing**

- **Timing/Report Timing**

  - **Ex:**
    - **Path Type:** max
    - **Setup Time:**
    - **Critical Path:**

  - Slack = Data Require Time – Data Arrival Time

---

Advanced Reliable Systems (ARES) Lab.

60
Critical Path Highlighting

- View/Highlight/Critical Path

Advanced Reliable Systems (ARES) Lab.
Timing Slack Histogram

Timing/Endpoint Slack

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

Resolution

Advanced Reliable Systems (ARES) Lab.
Edit Your Own Script File

- For convenient, you should edit your own synthesis script file. Whenever you want to synthesize 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:
```
#Setup/Execute Script
set cycle 10
set t_in 5
set t_out 0.5
set in_pad_delay 0.14
set out_pad_delay 0.86
set_current_design tsmc18wl10 -library slow
set_wire_load_model -name "tsmc18_w10" -library slow
set_wire_load_mode "top"
```

---

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 "birs_mem.vg"
#include "memory_8k_32.v"
#include "sc_memory.v"
#include "sr_memory.v"
```

- Add the following Synopsys directives to the test bench

```bash
initial begin
$FsdbDumpfile("birs_mem.Fsdb");
$FsdbDumpfile("birs.fsdb");
$FsdbDumpvars;
end

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

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

*.sdf File

Instance Name

Delay

Advanced Reliable Systems (ARES) Lab.
Simulation-Based Power Estimation Using PrimePower
Modify the `<synopsys_dc.setup>` file, and then save as `<synopsys_pp.setup>`.

---

Add the path where you put this file:

```
set search_path "<user>/teacher/jfl1/cell_lib/CBDK018_TSMC_Artisan/CIC/SynopsysDC $search_path"
set search_path "<user>/teacher/jfl1/cell_lib/CBDK018_TSMC_Artisan/orig_lib/aci/sc/symbols/synop...
set search_path "<user>/grad92/zwtseng/H0Y/rom_base_bist/8_bit_bist/bist_256x8_with_ECC/0331_b...
```

Delete the directives:

```
set link_library "HTM_BISD.db RA1SHD256x8Fast&40C_syn.db RA1SHD256x8Fast&40C_syn.db RA1SHD256...
set target_library "RA1SHD256x8Fast&40C_syn.db RA1SHD256x8Fast&40C_syn.db RA1SHD256x8Typical_syn...
```

```
set hdlin_translate_off_skip_text "TRUE"
set cdifout_netlist_only "TRUE"
set verilogout_no_trf true
set plot_command [plot_all] ...
```
VCD File Generation

- Add the following Synopsys directives to the test bench

```verilog
initial
begin
  $dumpfile("bisl_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`
- `pp_shell> read_vcd --strip_path test/CORE CORE.vcd`
- `pp_shell> set_waveform_options --interval 1 --format fsdb --file pwr1`
- `pp_shell> calculate_power --waveform --statistics`
- `pp_shell> report_power --file pwr_rpt --hier 2`
**Power Report**

**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

<table>
<thead>
<tr>
<th>Level</th>
<th>instance_Name (Cell_Name)</th>
<th>#_of_States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Power</td>
<td>Dynamic</td>
<td>Leakage</td>
</tr>
<tr>
<td>Power</td>
<td>Power</td>
<td>Power</td>
</tr>
<tr>
<td>in Watt</td>
<td>in Watt</td>
<td>in Watt</td>
</tr>
</tbody>
</table>

*--------- 0 pp_root () 71517932*

2.596e-04 2.593e-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_BISO (HTW_BISO) 71517932*

2.596e-04 2.593e-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_BISO/BISO (BISO) 53627079*

( 99.81%) ( 0.19%) ( 4.01%) (95.99%) ( 0.01%) ( 0.15%)

*--------- 3 HTW_BISO/BISO/BISO_top (BISO_top) 23228922*

8.707e-05 8.711e-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_BISO/BISO/BISO_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%) ( 1.35%) (98.62%) ( 0.03%) ( 0.06%)
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
Artisan SRAM Types:

<table>
<thead>
<tr>
<th>Generator</th>
<th>Product Name</th>
<th>Executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-Speed/Density Single-Port SRAM</td>
<td>SRAM-SP</td>
<td>ra1sh</td>
</tr>
<tr>
<td>High-Speed/Density Dual-Port SRAM</td>
<td>SRAM-DP</td>
<td>ra2sh</td>
</tr>
<tr>
<td>High-Density Single-Port SRAM</td>
<td>SRAM-SP-HD</td>
<td>ra1shd</td>
</tr>
<tr>
<td>High-Density Dual-Port SRAM</td>
<td>SRAM-DP-HD</td>
<td>ra2shd</td>
</tr>
<tr>
<td>Low-Power Single-Port SRAM</td>
<td>SRAM-SP-LP</td>
<td>ra1shl</td>
</tr>
</tbody>
</table>

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

[REF: Artisan User Manual]
Pin Descriptions for Single-Port SRAM

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

Advanced Reliable Systems (ARES) Lab.
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

1. WEN = 2'b11
   ![Diagram 1](image)

2. WEN = 2'b10
   ![Diagram 2](image)

3. WEN = 2'b01
   ![Diagram 3](image)

4. WEN = 2'b00
   ![Diagram 4](image)
Waveforms for Single-Port SRAM

- **Read Cycle**

![Waveform Diagram](image.png)
Waveforms for Single-Port SRAM (Cont’)

- Write Cycle

![Waveform Diagram for Single-Port SRAM Write Cycle]

- **CLK**: Oscillating signal indicating the clock cycle.
- **CEN**: Control signal for enabling or disabling the operation.
- **WEN**: Write enable signal, active during write operations.
- **A[j]**: Address bus for selecting the memory location.
- **D[i]**: Data bus for input and output data.
- **Q[i]**: Output data signals, Q1 and Q2, for status or output data.

Advanced Reliable Systems (ARES) Lab.
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

Generated Files
Selection
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:
  - $\sqrt{4}$ [✓], 8, 16
- Drive Strength
- Word-Write Mask:
  - [✓] 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**: 8
- **Drive Strength**: on
- **Word-Write Mask**: on
- **Word Partition Size**: 8
- **Top Metal Layer**: m5
- **Power Type**: m3, m4
- **Horizontal Ring Layer**: m2, m3
- **Vertical Ring Layer**: m2, m3, m4

---

**Ex: 64k RAM (with mask write)**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Q[31:0]</td>
</tr>
<tr>
<td>WEN[3:0]</td>
<td></td>
</tr>
<tr>
<td>CEN</td>
<td></td>
</tr>
<tr>
<td>OEN</td>
<td></td>
</tr>
<tr>
<td>A[9:0]</td>
<td></td>
</tr>
<tr>
<td>D[31:0]</td>
<td></td>
</tr>
</tbody>
</table>

---

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

![Menu with options]

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

![Memory spec. file generation interface]
LAB