Training Course of Design Compiler

• T. –W. Tseng, “ARES Lab 2008 Summer Training Course of Design Compiler”

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

Speaker: T. –J. Chen
Outline

- Basic Concept of the Synthesis
- Synthesis Using Design Compiler
Basic Concept of the Synthesis
Cell-Based Design Flow

System Level
MATLAB/ C/ C++/ System C/ ADS/ Covergen (MaxSim)

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
Conformal/ Formality

Design for Test
Physical Compiler/ Magma Blast Fusion

Gate Level

Layout Level

Post-Layout Verification

Advanced Reliable Systems (ARES) Lab.
What is Synthesis

- Synthesis = translation + optimization + mapping

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

HDL Source (RTL)

Translate (HDL Compiler)

No Timing Info.

Generic Boolean (GTECT)

Optimize + Mapping (Design Compiler)

Timing Info.

Target Technology

The synthesis is constraint driven and technology independent!!
Compile

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

Attributes & Constraints

Technology Library

Schematic

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

(Flatten Technology Library)

Logic Level Optimization

Structure

Gate Level Optimization

Map

Technology Library

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

- Verilog Basis
  - parameter declarations
  - wire, wand, wor declarations
  - reg declarations
  - input, output, inout declarations
  - continuous assignments
  - module instructions
  - gate instructions
  - always blocks
  - task statements
  - function definitions
  - for, while loop

- Synthesizable Verilog primitives cells
  - and, or, not, nand, nor, xor, xnor
  - bufif0, bufif1,notif0,notif1
Synthesizable Verilog (Cont’)

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

- Your RTL design
  - Functional verification by some high-level language
  - Also, the code coverage of your test benches should be verified (i.e. VN)
  - Coding style checking (i.e. n-Lint)
    - Good coding style will reduce most hazards while synthesis
    - Better optimization process results in better circuit performance
    - Easy debugging after synthesis

- Constraints
  - The area and timing of your circuit are mainly determined by your circuit architecture and coding style
  - There is always a trade-off between the circuit timing and area
  - In fact, a super tight timing constraint may be worked while synthesis, but failed in the Place & Route (P&R) procedure
Synthesis Using Design Compiler
## Related Files

<table>
<thead>
<tr>
<th>Folder</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTL</td>
<td>.synopsys_dc.setup</td>
<td>Design compiler setup file</td>
</tr>
<tr>
<td></td>
<td>my_script.tcl</td>
<td>Synthesis script file</td>
</tr>
<tr>
<td></td>
<td>my_design.v</td>
<td>Verilog files</td>
</tr>
<tr>
<td></td>
<td>tmy_design.v</td>
<td>Test bench</td>
</tr>
<tr>
<td></td>
<td>tsmc18.v</td>
<td>Verilog model of standard cells</td>
</tr>
</tbody>
</table>

**Ex:**

![File Manager - cae11:GTL]

Advanced Reliable Systems (ARES) Lab.
<.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 "/APP/cell_lib/CHUKU18_TSMC_Artisan/CIC/SynopsysUC/db $search_path"
set search_path "/usr/cad/synopsys/synthesis/cur/libraries/syn $search_path"
set search_path "/APP/cell_lib/CHUKU18_TSMC_Artisan/orle_lib/ac1/sy/symbols/synopsys $search_path"

set search_path "/usr2/grad97/tichen/tutorial orlv/NET $search_path"
set link_library "RAM_64B_fast3-40C_syn.db RAM_64B_fast60C_syn.db RAM_64B_slow_syn.db RAM_64B_typical_syn.db"
set target_library "RAM_64B Fast3-40C Syn db RAM_64B Fast60C Syn db RAM_64B Slow Syn db RAM_64B Typical Syn db"
set symbol_library tsmc18.db
set hdlin_translate_off_skip_text "TRUE"
set edifout_netlist_only "TRUE"
set verilogout_no_tr1 True
set plot_command Elpr -P1p3
```

Note that the MEM DB files are converted from the LIB files which are generated from the Artisan!!
## Settings for Using Memory

- **Convert *.lib to *.db**
  - Use `dc_shell` commands to convert a memory LIB file:
    - `%> dc_shell -t`
    - `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` to include the slow library:
    - `set link_library "* slow.db t13spsram512x32_slow.db dw_foundation.sldb"`
  - Set `target_library` to point to the slow library:
    - `set target_library "slow.db t13spsram512x32_slow.db"`
  - Add a "search path" to this file:
    - Add the path to the library in the .dc file.

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

```vhdl
module bisr_mem(clk,rst,ams,CS),bisl_mode,cmd_done,BGO,CS0,sh
    parameter WORD_LENGTH = 64;
    parameter ADR_LEN = 13;
```

---

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

Advanced Reliable Systems (ARES) Lab.
Getting Started

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

- `linux %> dv& (XG Mode)`

(GUI view of the Design Vision)
Read File

- 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**
PAD Parameters Extraction

- **Input PAD**
  - Input delay
  - Input driving

- **Output PAD**
  - Output delay
  - Output loading

```
set_driving_cell -lib_cell P010GZ -libary tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [ADDR_S01]]
set_driving_cell -lib_cell P010GZ -library tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [bira_en]]
set_driving_cell -lib_cell P010GZ -library tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [test_data]]
set_driving_cell -lib_cell P010GZ -library tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [disp_model]]
set_driving_cell -lib_cell P010GZ -library tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [chip_model]]
set_load -pin_load 0.00132 [get_ports [cmd_done]]
set_load -pin_load 0.00132 [get_ports [IC0]]
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 [bira_out_valid]]
set_load -pin_load 0.00132 [get_ports [addr_change]]
set_driving_cell -lib_cell P010GZ -library tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [s3]]
set_driving_cell -lib_cell P010GZ -library tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [s1]]
set_driving_cell -lib_cell P010GZ -library tpz973gbc -pin C -from_pin PAD 
-no_design_rule [get_ports [scantests]]
set_load -pin_load 0.00132 [get_ports [s0]]
```

```
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**

## Table

<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

<table>
<thead>
<tr>
<th>Hierarchy</th>
<th>Design</th>
<th>Schematic</th>
<th>Attributes</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ungroup...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniquify</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Logical Hierarchy View</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Log Window

```plaintext
$ 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'.
```
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**

- **Command Line**

  `set_operating_conditions -max "slow" -max_library "slow" -min "fast" -min_library "fast"`
Setting Drive Strength/Input Delay for PADs

- Assume that we use the input PAD “PDIDGZ”

```
set_drive [expr 0.288001] [all_inputs]
set_input_delay [expr 0.34] –clock clk [all_inputs]
```
Setting Load/Output Delay for PADs

- Assume that we use the output PAD “PDO24CDG”

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

Attributes/Operating Environment/Wire Load

- **Attributes**
  - Timing
  - Test
  - Window
  - Help

- **Operating Environment**
  - Specify Clock...
  - Operating Environment
  - Optimization Constraints
  - Optimization Directives

- **Wire Load**
  - Input Delay...
  - Output Delay...
  - Drive Strength...
  - Load...
  - Characterize...
  - Operating Conditions...

- **Timing Range**

### Command Line

- `set_wire_load_model -name "tsmc18_wl10" -library "slow"
- `set_wire_load_mode “top”`

(Worst Case)

Recommended:

- `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 }
creat_clock -period 10 [get_ports clk]
set_dont_touch_network [get_clocks clk]
set_fix_hold [get_clocks clk]
Setting Clock Skew

- Different clock arrival time
  - Ex:

  ![Clock Circuit Diagram]

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

(Timing Report)

```plaintext
{ Command Line }
set_clock_uncertainty 0.1 [get_clocks clk]
```
Setting Clock Latency

- **Source latency** is the propagation time from the actual clock origin to the clock definition point in the design.
- This setting can be 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.

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

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

```
set_input_transition -max 0.5 [all_inputs]
```

Advanced Reliable Systems (ARES) Lab.
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**

Advanced Reliable Systems (ARES) Lab.
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

```plaintext
set_max_area 0
set_max_fanout 50 [get_designs CORE]
```
Compile the Design

- Design/Compile Design

```
compile -map_effort high -boundary_optimization
```
Assign Problem

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

```plaintext
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];
```

```plaintext
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]) );
```

```plaintext
{ Command Line }
set_fix_multiple_port_nets --all --constants --buffer_constants [get_designs *]
```
Floating Port Removing

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

\{ Command Line \}

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

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

```
{ Command Line }
write_test_protocol -f stil -out "CHIP.spf"
write_sdc CHIP.sdc
write -format verilog -hierarchy -output "CHIP.vg"
write_sdf -version 1.0 CHIP.sdf
write -format db -hierarchy -output "CHIP.db"
```
Synthesis Report

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

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

Ex:

```
+---------------------------------------------------------------------+
| Report: 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)

```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
Report Area

- **Design/Report Area**

Ex:

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

Library(s) Used:
gtech (File: /usr/cad/synopsys/synthesis/cuR/libraries/syn/gtech.db)
USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/mem/DB/memory_8k_32_fast@-40C_syn.db)
USERLIB (File: /usr4/grad92/zwtseng/dv_training/RTL/MEM/DB/sc_memory_fsst@-40C_syn.db)
USERLIB (File: /usr4/grad92/zwtseng/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 *****
```
**List/Design View**

All the block area are listed!!

<table>
<thead>
<tr>
<th>array_or</th>
<th>undefined</th>
<th>undefined</th>
<th>undefined</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitmap</td>
<td>15501</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bsr_mem</td>
<td>3.34175e+06</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>memory_8k</td>
<td>3.29505e+06</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>aru</td>
<td>7717.25</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>remapping_</td>
<td>4440.74</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>address_6e_</td>
<td>8276.08</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>finj</td>
<td>1203.97</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bsr</td>
<td>46636</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bira_top</td>
<td>26953.8</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>remap</td>
<td>8016.62</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>remap_DW_</td>
<td>192.931</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>multi_bit</td>
<td>1593.35</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bitmap_DW_</td>
<td>83.16</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>fsm</td>
<td>1816.21</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>bist</td>
<td>17413.7</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>tpg</td>
<td>15151.8</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ADDR</td>
<td>2421.62</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ADDR_DW_</td>
<td>435.758</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>ADDR_DW_</td>
<td>472.349</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>CMP</td>
<td>9433.87</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>DATA</td>
<td>219.542</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>DECO</td>
<td>804.889</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ROM</td>
<td>1147.61</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>ctr</td>
<td>2251.95</td>
<td>undefined</td>
<td>undefined</td>
</tr>
</tbody>
</table>
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
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
  - **File/Execute Script**
  - Or use “source your_script.dc” in dc_shell command line

```bash
# set cycle 10
# set t_in 5
# set t_out 0.5
# set in_pad_delay 0.34
# set out_pad_delay 0.46
# set Current Design
# current_design blisr_mem
# uniquufy
# set operating conditions -max "slow" -max_library "slow" -min "fast"
# -min_library "fast"
# set wire_load_model -name "tsmc18_w110" -library "slow"
# set wire_load_mode "top"

clock

create_clock -period 10 [get_ports clk]
set_dont_touch_network [get_clocks clk]
set_fix_hold [get_clocks clk]
set_clock_uncertainty 0.1 [get_clocks clk];
set_clock_latency 1 [get_clocks clk];
set_ideal_network [get_ports clk]
set_input_transition -max 0.3 [all_inputs];

set_input_delay [expr 0.34] -clock clk [all_inputs]
set_input_delay [expr 0.34] -clock clk [all_inputs]
set_input_delay [expr 0.34] -clock clk [all_inputs]
set_output_delay [expr 0.34] -clock clk [all_outputs]
set_output_delay [expr 0.34] -clock clk [all_outputs]
set_load [expr 0.0613] [all_outputs]
set_drive [expr 0.288001] [all_inputs]

#set_max_fanout 50 [get_designs blisr_mem]
#set_max_area 0
```

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 "biss_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
  $fsvdbDumpfile("biss_mem.fsvdb");
  $fsvdbDumpfile("biss.fsvdb");
  $fsvdbDumpvars;
  end

  initial begin
  $sdf_anotate("biss_mem.sdf",biss_mem);
  end

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