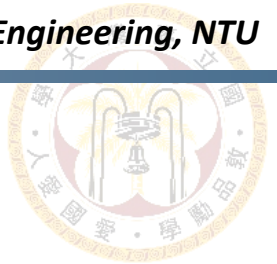
Computer-Aided VLSI System Design

Chapter 5. Synopsys Synthesis Design Compiler & Post-sim Part1

Lecturer: 王傑立

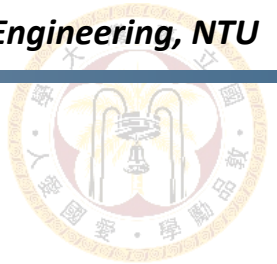
Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE



Outline

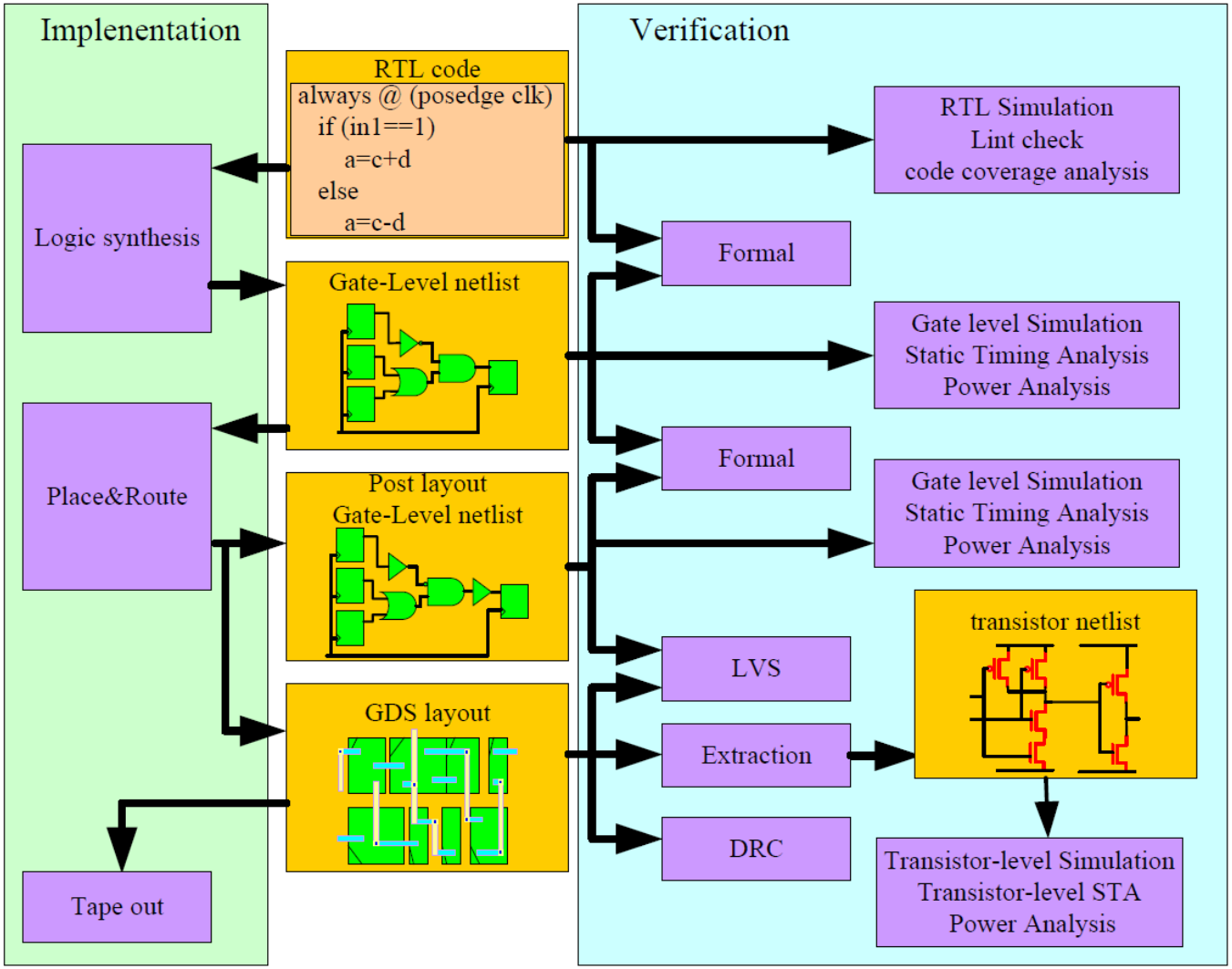
- Introduction
- RTL Coding Related to Synthesis
- Synopsys Environment
- Synthesis Design Flow
- Gate-Level Simulation
- Files Demo



Outline

- Introduction
- RTL Coding Related to Synthesis
- Synopsys Environment
- Synthesis Design Flow
- Gate-Level Simulation
- Files Demo

Cell-based IC Design Flow

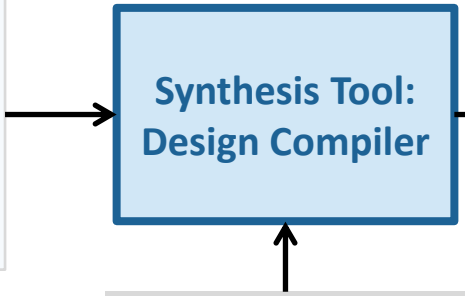


Logic Synthesis

- A process to convert RTL into a technology-specific gate-level netlist optimized with a set of design constraints
  - Library
    - Standard cell library
  - User data
    - RTL design
    - Design constraints

```
module counter (
    input i_clk, rstn, load,
    input [1:0] in,
    output reg [1:0] out);
    always@(posedge i_clk) begin
        if(!rstn) out <= 2'b0;
        else if (load) out <= in;
        else out <= out+1;
    end
endmodule
```

RTL Design
counter.v



Standard Cell Lib.

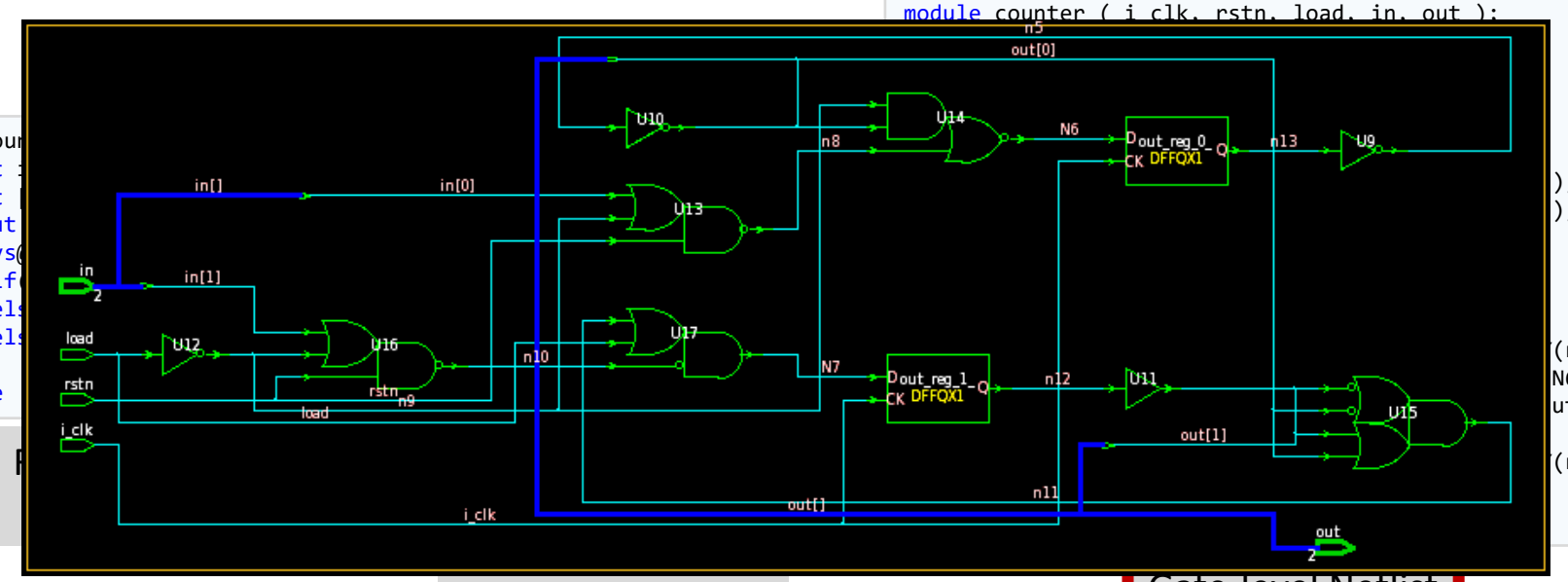
Design Constraints

```
module counter ( i_clk, rstn, load, in, out );
  input [1:0] in;
  output [1:0] out;
  input i_clk, rstn, load;
  wire   n12, n13, N6, N7, n5, n8, n9, n10, n11;

  DFFQX1 out_reg_1_ ( .D(N7), .CK(i_clk), .Q(n12) );
  DFFQX1 out_reg_0_ ( .D(N6), .CK(i_clk), .Q(n13) );
  CLKINVX1 U9 ( .A(n13), .Y(n5) );
  INVX16 U10 ( .A(n5), .Y(out[0]) );
  BUFX16 U11 ( .A(n12), .Y(out[1]) );
  INVXL U12 ( .A(load), .Y(n9) );
  OAI21XL U13 ( .A0(in[0]), .A1(n9), .B0(rstn), .Y(n8) );
  AOI21XL U14 ( .A0(n9), .A1(out[0]), .B0(n8), .Y(N6) );
  AOI2BB2X1 U15 ( .B0(out[1]), .B1(out[0]), .A0N(out[1]),
                  .A1N(out[0]), .Y(n11) );
  OAI21XL U16 ( .A0(in[1]), .A1(n9), .B0(rstn), .Y(n10) );
  AOI2BB1X1 U17 ( .A0N(n11), .A1N(load), .B0(n10), .Y(N7) );
endmodule
```

Gate-level Netlist
counter_syn.v

Logic Synthesis

- Gate-level Netlist
  - Describes the circuit with standard cells



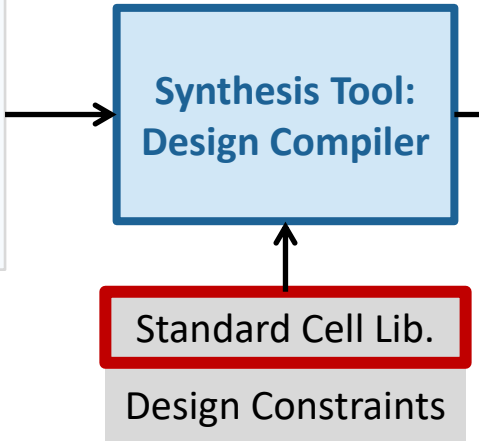
Gate-level Netlist
counter_syn.v

Logic Synthesis

- Standard cell library defines the components for gate-level netlist
- Standard cell library provided by foundry (e.g., TSMC)
  - Standard cell list, e.g., AND, XOR, NAND, D flip-flop, etc.
  - Area, delay, power, functionality of each cell

```
module counter (
    input i_clk, rstn, load,
    input [1:0] in,
    output reg [1:0] out);
    always@(posedge i_clk) begin
        if(!rstn) out <= 2'b0;
        else if (load) out <= in;
        else out <= out+1;
    end
endmodule
```

RTL Design
counter.v




```
module counter ( i_clk, rstn, load, in, out );
  input [1:0] in;
  output [1:0] out;
  input i_clk, rstn, load;
  wire   n12, n13, N6, N7, n5, n8, n9, n10, n11;

  DFFQX1 out_reg_1_ ( .D(N7), .CK(i_clk), .Q(n12) );
  DFFQX1 out_reg_0_ ( .D(N6), .CK(i_clk), .Q(n13) );
  CLKINVX1 U9 ( .A(n13), .Y(n5) );
  INVX16 U10 ( .A(n5), .Y(out[0]) );
  BUFX16 U11 ( .A(n12), .Y(out[1]) );
  INVXL U12 ( .A(load), .Y(n9) );
  OAI21XL U13 ( .A0(in[0]), .A1(n9), .B0(rstn), .Y(n8) );
  AOI21XL U14 ( .A0(n9), .A1(out[0]), .B0(n8), .Y(N6) );
  AOI2BB2X1 U15 ( .B0(out[1]), .B1(out[0]), .A0N(out[1]),
                  .A1N(out[0]), .Y(n11) );
  OAI21XL U16 ( .A0(in[1]), .A1(n9), .B0(rstn), .Y(n10) );
  AOI2BB1X1 U17 ( .A0N(n11), .A1N(load), .B0(n10), .Y(N7) );
endmodule
```

Gate-level Netlist
counter_syn.v

Logic Synthesis

- Design constraints
  - Conditions or requirements for the gate-level netlist
    - E.g., require the circuit to run at 100 MHz, minimize the area
  - Iteratively optimization is performed based on constraints

```
module counter (
    input i_clk, rstn, load,
    input [1:0] in,
    output reg [1:0] out);
    always@(posedge i_clk) begin
        if(!rstn) out <= 2'b0;
        else if (load) out <= in;
        else out <= out+1;
    end
endmodule
```

RTL Design
counter.v

Synthesis Tool:
Design Compiler

Standard Cell Lib.

Design Constraints

```
module counter ( i_clk, rstn, load, in, out );
  input [1:0] in;
  output [1:0] out;
  input i_clk, rstn, load;
  wire   n12, n13, N6, N7, n5, n8, n9, n10, n11;

  DFFQX1 out_reg_1_ ( .D(N7), .CK(i_clk), .Q(n12) );
  DFFQX1 out_reg_0_ ( .D(N6), .CK(i_clk), .Q(n13) );
  CLKINVX1 U9 ( .A(n13), .Y(n5) );
  INVX16 U10 ( .A(n5), .Y(out[0]) );
  BUFX16 U11 ( .A(n12), .Y(out[1]) );
  INVXL U12 ( .A(load), .Y(n9) );
  OAI21XL U13 ( .A0(in[0]), .A1(n9), .B0(rstn), .Y(n8) );
  AOI21XL U14 ( .A0(n9), .A1(out[0]), .B0(n8), .Y(N6) );
  AOI2BB2X1 U15 ( .B0(out[1]), .B1(out[0]), .A0N(out[1]),
                  .A1N(out[0]), .Y(n11) );
  OAI21XL U16 ( .A0(in[1]), .A1(n9), .B0(rstn), .Y(n10) );
  AOI2BB1X1 U17 ( .A0N(n11), .A1N(load), .B0(n10), .Y(N7) );
endmodule
```

Gate-level Netlist
counter_syn.v

1. Constraint-driven Optimization

- The gate-level netlist generated in synthesis is meant to meet several goals defined by designer
  - Design rule constraints
    - E.g., capacitance limit/fanout/transition time limit for each wire
  - Timing requirements for each path
  - Power constraint
  - Area constraint

- Optimization in synthesis is **constraint-driven**
  - Not exhaustive search
  - The gate-level netlist is iteratively optimized for unmet goals

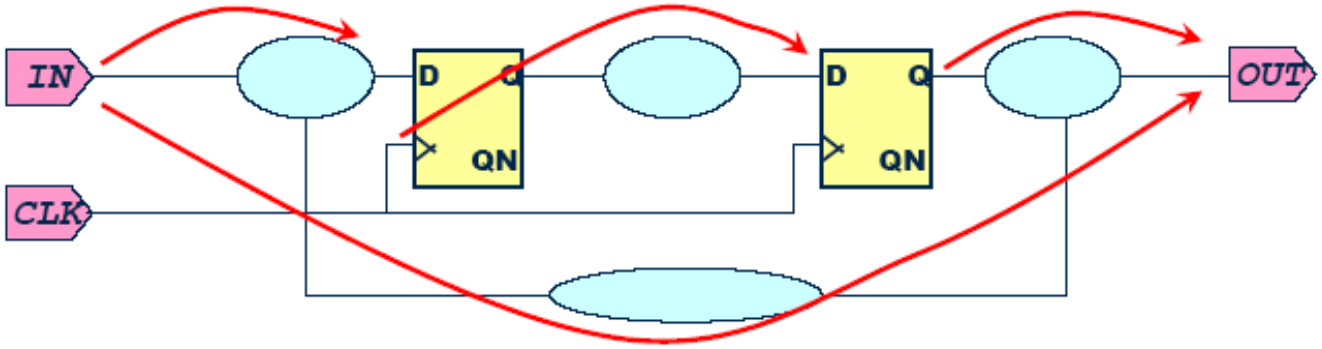
1. Constraint-driven Optimization

- Optimization process example in Design Compiler

```
                                  TOTAL
 ELAPSED            WORST NEG     SETUP    DESIGN                                LEAKAGE    MIN DELAY
  TIME      AREA      SLACK       COST    RULE COST        ENDPOINT              POWER      COST
--------- --------- --------- --------- --------- ------------------------- --------- -----------
  0:00:09   24208.3      0.07       7.2       0.0                          24438066.0000       0.00
  0:00:09   24208.3      0.07       7.2       0.0                          24438066.0000       0.00

Beginning Constant Register Removal
-----------------------------------

Beginning Global Optimizations
------------------------------
Numerical Synthesis (Phase 1)
Numerical Synthesis (Phase 2)
Global Optimization (Phase 1)
Global Optimization (Phase 2)
Global Optimization (Phase 3)
Global Optimization (Phase 4)

Beginning Isolate Ports
-----------------------

Beginning Delay Optimization
----------------------------
  0:00:10   15865.6      0.51      48.3      37.6                          15621728.0000       0.00
  0:00:12   16744.9      0.13      13.0      19.3                          16497935.0000       0.00
  0:00:12   16744.9      0.13      13.0      19.3                          16497935.0000       0.00
  0:00:12   16733.0      0.14      13.4       0.0                          16813510.0000       0.00
  0:00:12   16733.0      0.14      13.4       0.0                          16813510.0000       0.00
  0:00:12   16733.0      0.14      13.4       0.0                          16813510.0000       0.00

Beginning WLM Backend Optimization
----------------------------------
  0:00:13   16673.6      0.14      13.9       0.0                          16743739.0000       0.00
  0:00:13   16673.6      0.14      13.9       0.0                          16743739.0000       0.00
  0:00:13   16673.6      0.14      13.9       0.0                          16743739.0000       0.00
  0:00:13   16783.9      0.14      13.6       0.0                          16824478.0000       0.00
  0:00:13   16783.9      0.14      13.6       0.0                          16824478.0000       0.00
  0:00:13   17700.5      0.12      10.9       0.0                          17591000.0000       0.00
  0:00:14   17700.5      0.12      10.9       0.0                          17591000.0000       0.00
  0:00:14   17748.0      0.12      10.5       0.0                          17656992.0000       0.00
  0:00:14   17748.0      0.12      10.5       0.0                          17656992.0000       0.00
  0:00:14   17748.0      0.12      10.5       0.0                          17656992.0000       0.00
  0:00:14   17748.0      0.12      10.5       0.0                          17656992.0000       0.00
  0:00:16   18335.3      0.08       6.1       6.6                          18312380.0000       0.00
  0:00:16   18335.3      0.08       6.1       6.6                          18312380.0000       0.00
```

2. Timing Paths & Timing Requirements

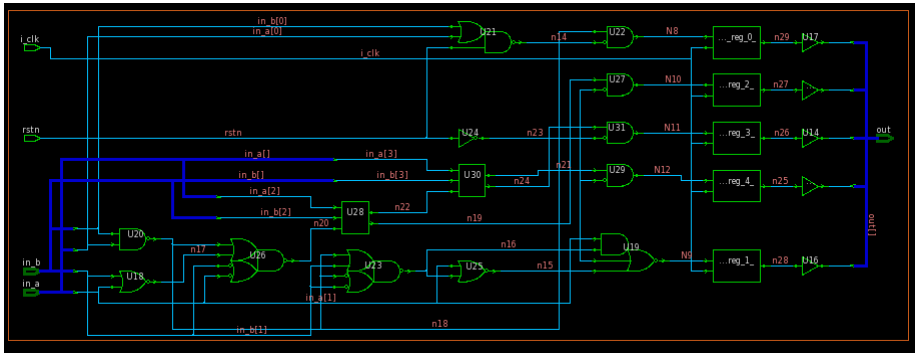
- Using inputs, outputs and registers as endpoints, the combinational part of design can be divided into 4 type of paths
  - Input to register
  - Register to register
  - Register to output
  - Input to output
- In the synthesized netlist, all paths should meet timing requirements
  - E.g., in a gate-level netlist synthesized for 200MHz, delay on all reg-to-reg paths should ≤ 5ns



3. Trade-off between Timing and Area

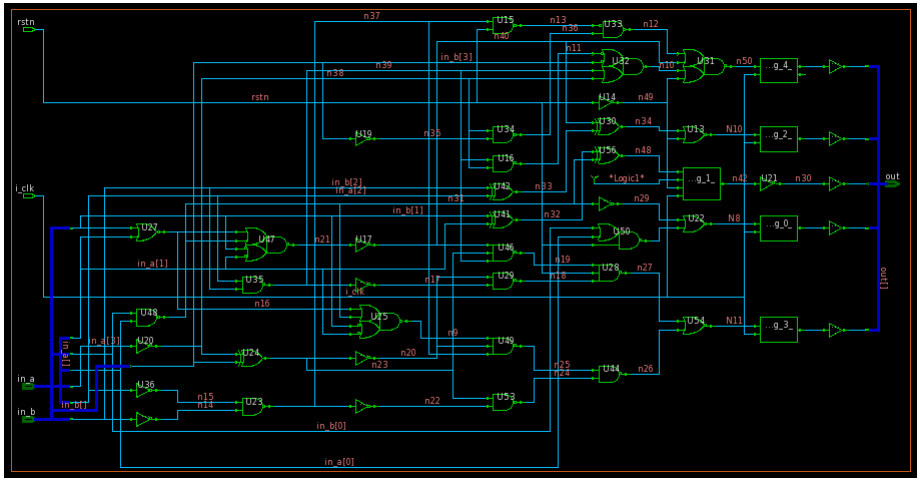
```
module adder (
    input i_clk, rstn,
    input [3:0] in_a,in_b,
    output reg [4:0] out);
    always@(posedge i_clk) begin
        if (!rstn) out <= 0;
        else out <= in_a+in_b;
    end
endmodule
```

Clock Cycle = 10 ns



Cell area = 361 um^2

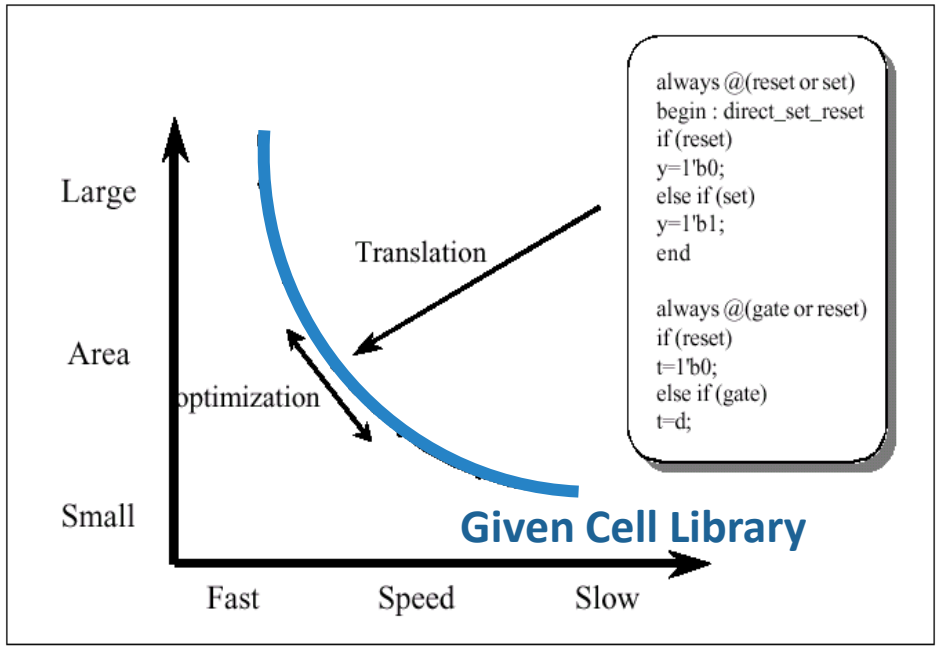
Clock Cycle = 5 ns

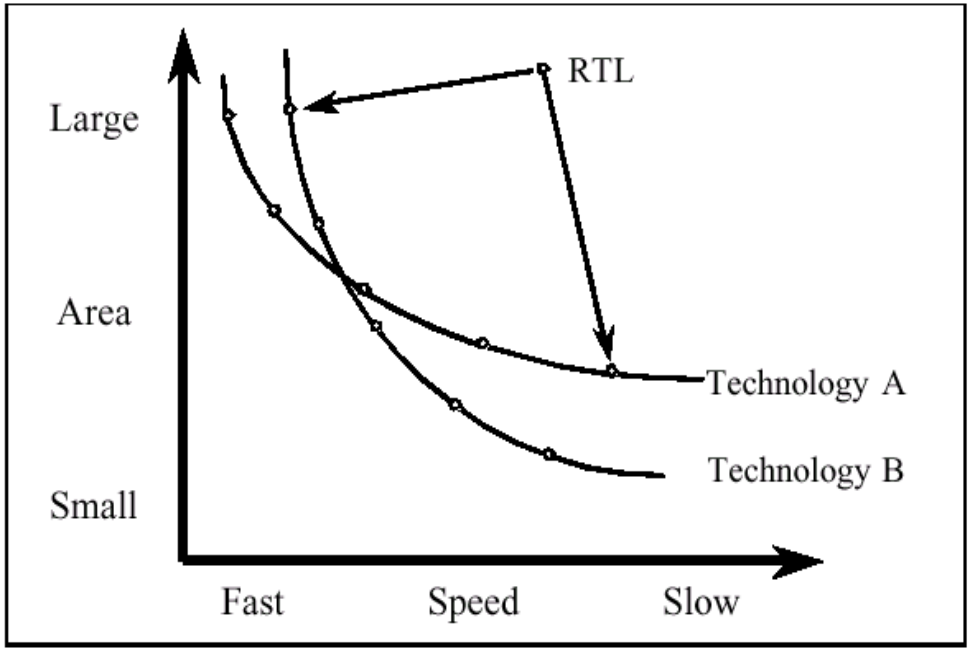


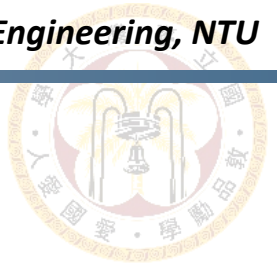
Cell area = 729 um^2

3. Trade-off between Timing and Area

- Given the same library and the same source code, you can
  - Sacrifice area for higher speed
  - Sacrifice speed for lower area







Outline

- Introduction
- **RTL Coding Related to Synthesis**
- Synopsys Environment
- Synthesis Design Flow
- Gate-Level Simulation
- Files Demo

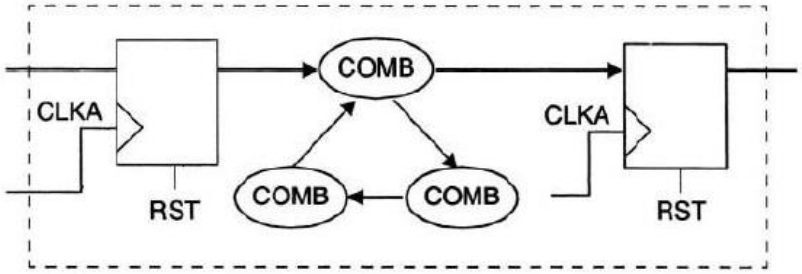
Unsupported Syntax for Synthesis

- These syntaxes are for simulation only and cannot be converted to gates
  - delay
  - initial
  - repeat
  - wait
  - fork … join
  - event
  - deassign
  - force
  - release
  - primitive
  - time
  - triand, trior, tri1, tri0, trireg
  - nmos, pmos, cmos, rnmos, rpmos, rcmos
  - pullup, pulldown
  - rtran, tranif0, tranif1, rtranif0, rtranif1
  - case identity (===)
  - not identity (!==)

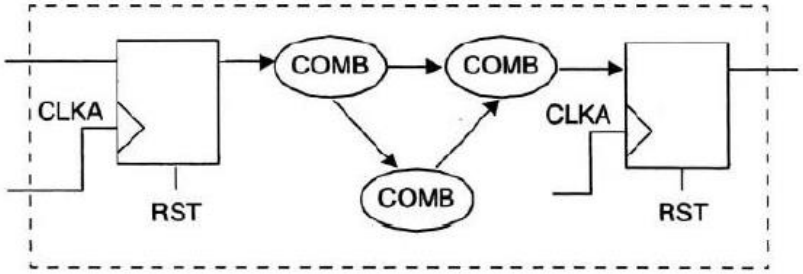
Avoid Combinational Loop

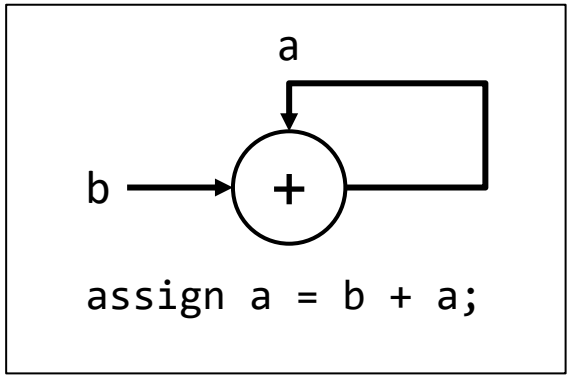
- An output of a combinational block feeds back to an input of the same block

Bad: Combinational processes are looped



Good: Combinational processes are not looped





An error example

Static timing analysis (STA) cannot be applied

Avoid Combinational Loop

- Check_timing
  - Checks for possible timing problems in the current design
  - Can detect combination loop, unconstrainted path, etc.

```
Information: Checking out the license 'DesignWare'. (SEC-104)
Information: Updating design information... (UID-85)

Information: Checking generated_clocks...

Information: Checking loops...

Information: Checking no_input_delay...

Information: Checking unconstrained_endpoints...

Information: Checking pulse_clock_cell_type...

Information: Checking no_driving_cell...

Information: Checking partial_input_delay...
1

```

Example of check_timing
(No problems detected)

```
Information: Updating design information... (UID-85)

Information: Checking generated_clocks...

Information: Checking loops...
Warning: timing loops detected. (TIM-209)
    #              loop breakpoint

                cell_arc                            lib cell
------------------------------------------------------------------------------
C38/B                   -> C38/Z                    gtech/GTECH_AND2        #
C54/A                   -> C54/Z                    gtech/GTECH_OR2

                cell_arc                            lib cell
------------------------------------------------------------------------------
C39/B                   -> C39/Z                    gtech/GTECH_AND2        #
C55/A                   -> C55/Z                    gtech/GTECH_OR2

                cell_arc                            lib cell
------------------------------------------------------------------------------
C40/B                   -> C40/Z                    gtech/GTECH_AND2        #
C56/A                   -> C56/Z                    gtech/GTECH_OR2

                cell_arc                            lib cell
------------------------------------------------------------------------------
C41/B                   -> C41/Z                    gtech/GTECH_AND2        #
C57/A                   -> C57/Z                    gtech/GTECH_OR2

                cell_arc                            lib cell
------------------------------------------------------------------------------
C42/B                   -> C42/Z                    gtech/GTECH_AND2        #
```

Loops detected in check_timing

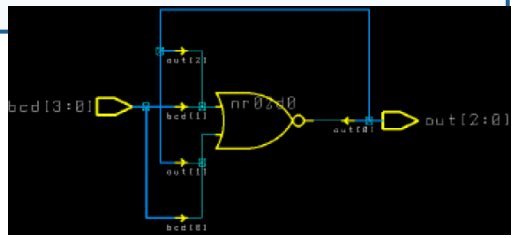
Supported Syntax for Synthesis

- Operators
  - Binary bit-wise (~, &, |, ^, ~^)
  - Unary reduction (&, ~&, |, ~|, ^, ~^)
  - Logical (~, &&, ||)
  - 2's complement arithmetic (+, -, *, /, %)
  - Relational (>, <, >=, <=)
  - Equality (==, !=)
  - Logical shift (>>, <<, >>>, <<<)
  - Conditional (?:)
- Continuous assignment (assign)
- Always block
  - Combinational always block (always@(*))
  - Register (always@(posedge clk))
  - if-else & case statements

Special Syntax: Compiler Directives

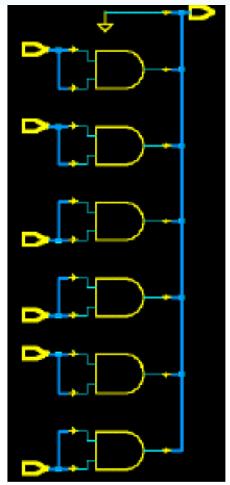
- synopsys full_case
  - Do not specify default case/all cases
  - Requirement: no other cases will appear
  - Other branches will be considered as don't-care conditions

```
always@ (bcd) begin
    case (bcd) //synopsys full_case
        4'd0: out = 3'b001;
        4'd1: out = 3'b010;
        4'd2: out = 3'b100;
    endcase
end
```



- synopsys parallel_case
  - Eliminate priority in case statements
  - Requirement: only 1 case matched at a time

```
always@ (u or v or w or x or y or z) begin
    case (2'b11) //synopsys parallel_case
        u: out = 6'b000001;
        v: out = 6'b000010;
        w: out = 6'b000100;
        x: out = 6'b001000;
        y: out = 6'b010000;
        z: out = 6'b100000;
    default:
            out = 6'b000000;
    endcase
end
```

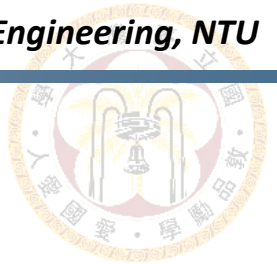


Special Syntax: Compiler Directives

- Example
  - synopsys full_case parallel_case for one-hot finite state machines (FSMs)

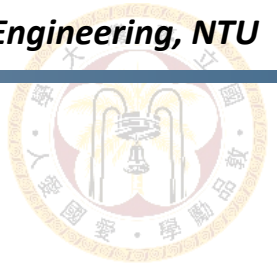
```
// Coding Style 1
case({sel3, sel2, sel1}) //synopsys full_case parallel_case
    3'b001: out = in1;
    3'b010: out = in2;
    3'b100: out = in3;
endcase
```

```
// Coding Style 2
case(1'b1) //synopsys full_case parallel_case
    sel1: out = in1;
    sel2: out = in2;
    sel3: out = in3;
endcase
```



RTL Coding Related to Synthesis

- Special-case needs constraint setting
  - Clock gating
  - Multi-cycle path
  - False path
  - Asynchronous logic
- General guidelines for RTL coding
  - Separate combinational and sequential part
  - Separate control and VLSI-design strategy
  - Register at input/output



Outline

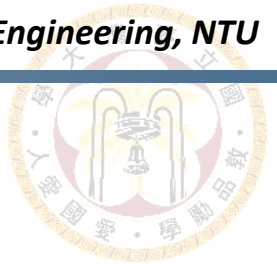
- Introduction
- RTL Coding Related to Synthesis
- **Synopsys Environment**
  - Tools Used in Logic Synthesis
  - Design Objects
  - DC-TCL: Introduction
- Synthesis Design Flow
- Gate-Level Simulation
- Files Demo

Tools We Will Use

Tool	Purpose
Design Compiler (DC)	Constraint driven logic optimizer
Design Vision	Graphical User Interface Version of DC
Presto HDL Compiler	Translate Verilog descriptions into GTECH
Design Ware	Provide building blocks & operators for synthesis
Design Time	Static Timing Analysis (STA) engine
Power Compiler	For power optimization, e.g., multi-V_t, clock gating

Design Compiler

```
                        Design Compiler Graphical
                              DC Ultra (TM)
                               DFTMAX (TM)
                            Power Compiler (TM)
                               DesignWare (R)
                               DC Expert (TM)
                            Design Vision (TM)
                            HDL Compiler (TM)
                            VHDL Compiler (TM)
                               DFT Compiler
                            Design Compiler(R)

                Version R-2020.09-SP5 for linux64 - Apr 23, 2021

                     Copyright (c) 1988 - 2021 Synopsys, Inc.
   This software and the associated documentation are proprietary to Synopsys,
 Inc. This software may only be used in accordance with the terms and conditions
 of a written license agreement with Synopsys, Inc. All other use, reproduction,
   or distribution of this software is strictly prohibited.  Licensed Products
     communicate with Synopsys servers for the purpose of providing software
    updates, detecting software piracy and verifying that customers are using
    Licensed Products in conformity with the applicable License Key for such
  Licensed Products. Synopsys will use information gathered in connection with
    this process to deliver software updates and pursue software pirates and
                                 infringers.

 Inclusivity & Diversity - Visit SolvNetPlus to read the "Synopsys Statement on
            Inclusivity and Diversity" (Refer to article 000036315 at
                          https://solvnetplus.synopsys.com)
Initializing...
Initializing gui preferences from file  /home/raid7_2/user09/r09176/.synopsys_dc_g
dc_shell>
```



Launching Design Compiler

- Launching design compiler

```
dc_shell
```

- Launching graphical user interface

```
design_vision
```

```
dc_shell -gui
```

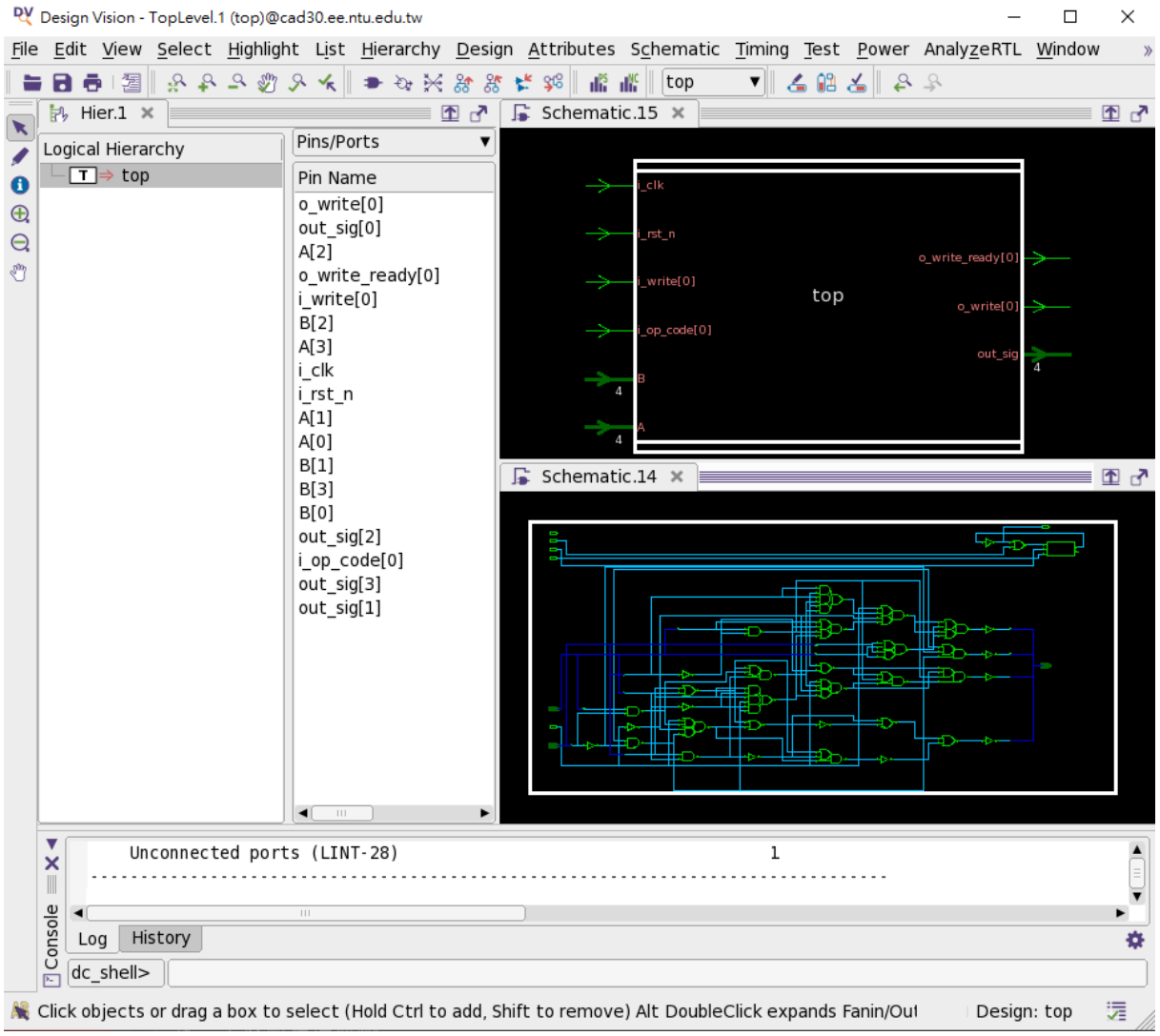
```
dc_shell
dc_shell> gui_start
```

- Running script

```
dc_shell -f script.tcl
```

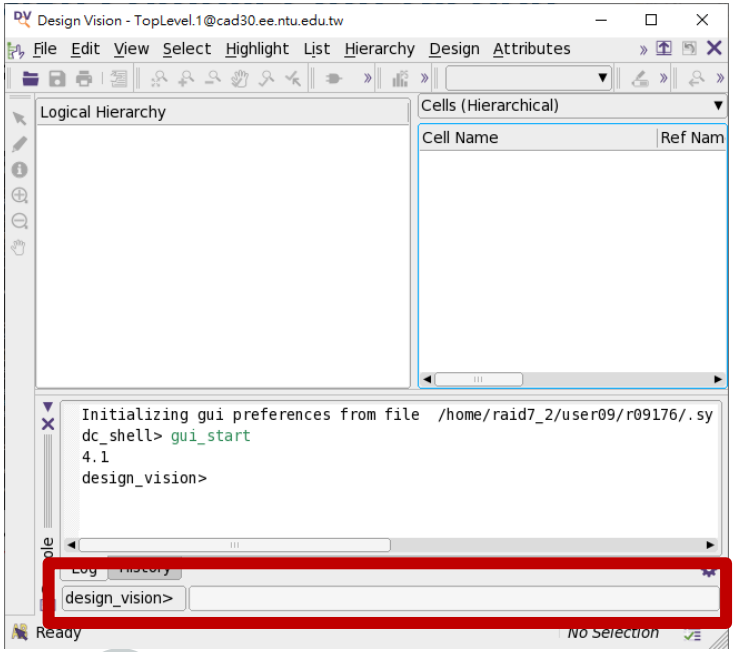
```
dc_shell
dc_shell> source script.tcl
```

Design Vision



Design Compiler Interaction

3 GUI



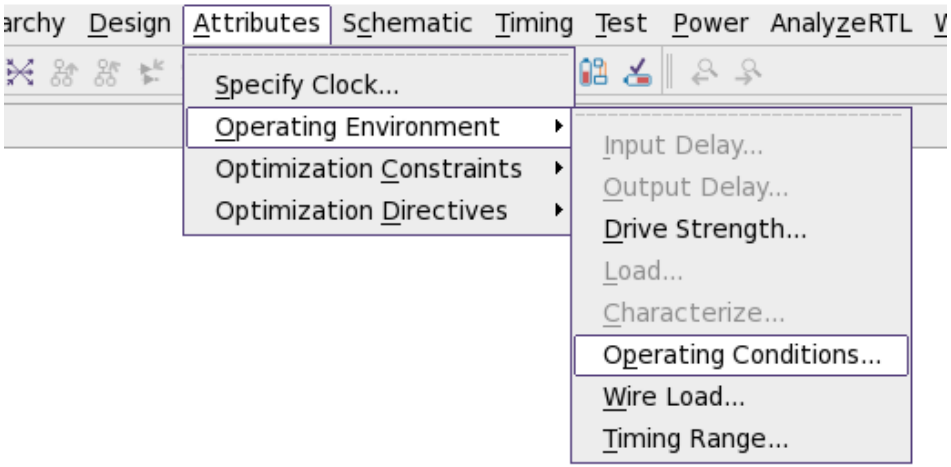
2 Command line

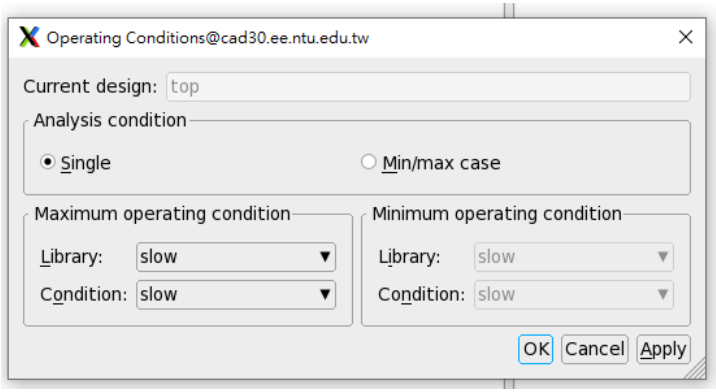
```
                        Design Compiler Graphical
                              DC Ultra (TM)
                              DFTMAX (TM)
                           Power Compiler (TM)
                              DesignWare (R)
                              DC Expert (TM)
                            Design Vision (TM)
                            HDL Compiler (TM)
                            VHDL Compiler (TM)
                               DFT Compiler
                            Design Compiler(R)

                    Version R-2020.09-SP5 for linux64 - Apr 23, 2021

                        Copyright (c) 1988 - 2021 Synopsys, Inc.
   This software and the associated documentation are proprietary to Synopsys,
 Inc. This software may only be used in accordance with the terms and conditions
 of a written license agreement with Synopsys, Inc. All other use, reproduction,
   or distribution of this software is strictly prohibited.  Licensed Products
     communicate with Synopsys servers for the purpose of providing software
    updates, detecting software piracy and verifying that customers are using
    Licensed Products in conformity with the applicable License Key for such
  Licensed Products. Synopsys will use information gathered in connection with
    this process to deliver software updates and pursue software pirates and
                                 infringers.

 Inclusivity & Diversity - Visit SolvNetPlus to read the "Synopsys Statement on
            Inclusivity and Diversity" (Refer to article 000036315 at
                          https://solvnetplus.synopsys.com)
Initializing...
Initializing gui preferences from file  /home/raid7_2/user09/r09176/.synopsys_dc_g
dc_shell>
```

1 dc_shell

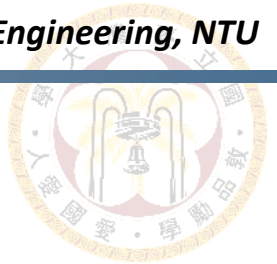
Design Compiler Interaction

- Each action in GUI is equivalent to a series of commands
  - But only a part of commands can be done by GUI actions





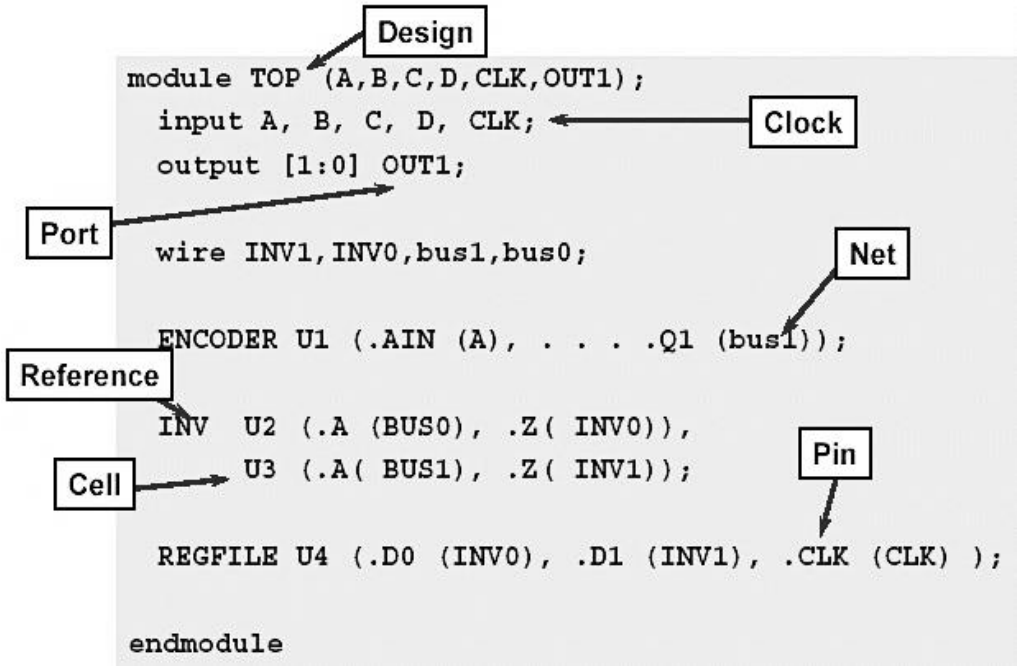

```
dc_shell> set_operating_conditions -library slow slow
Using operating conditions 'slow' found in library 'slow'.
dc_shell>
```



Outline

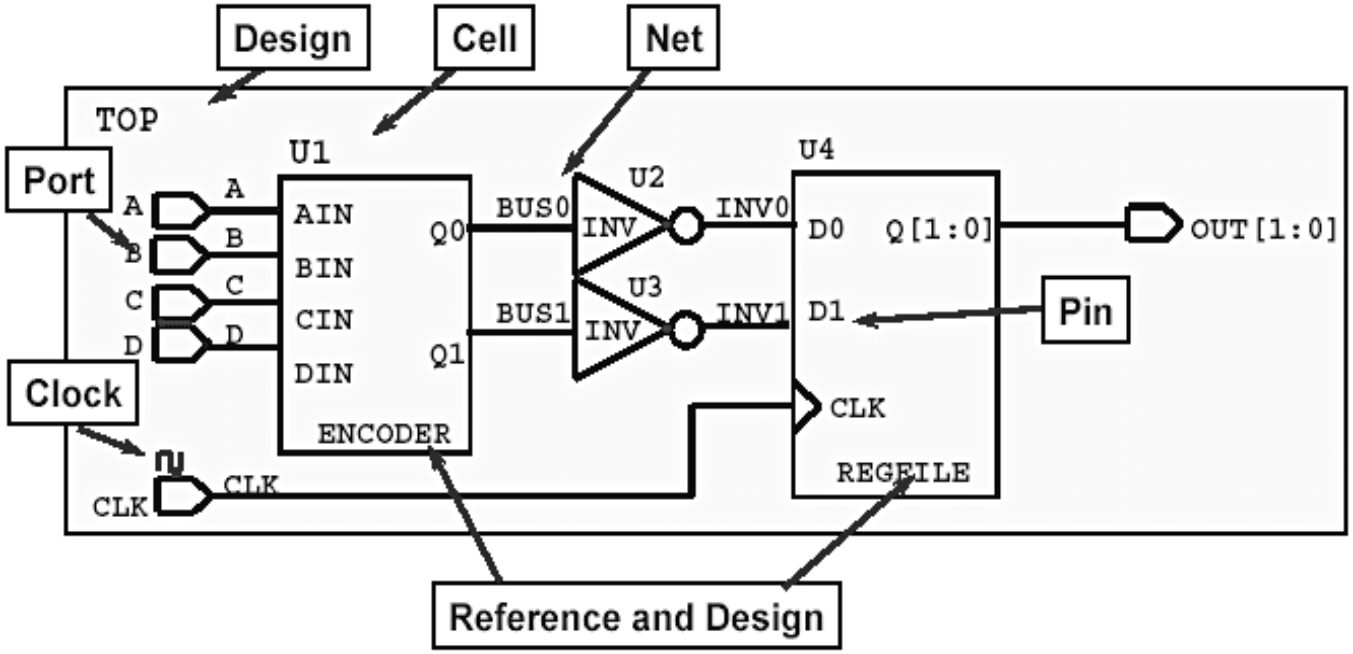
- Introduction
- RTL Coding Related to Synthesis
- **Synopsys Environment**
  - Tools Used in Logic Synthesis
  - Design Objects
  - DC-TCL: Introduction
- Synthesis Design Flow
- Gate-Level Simulation
- Files Demo

Design Objects (Verilog Perspective)



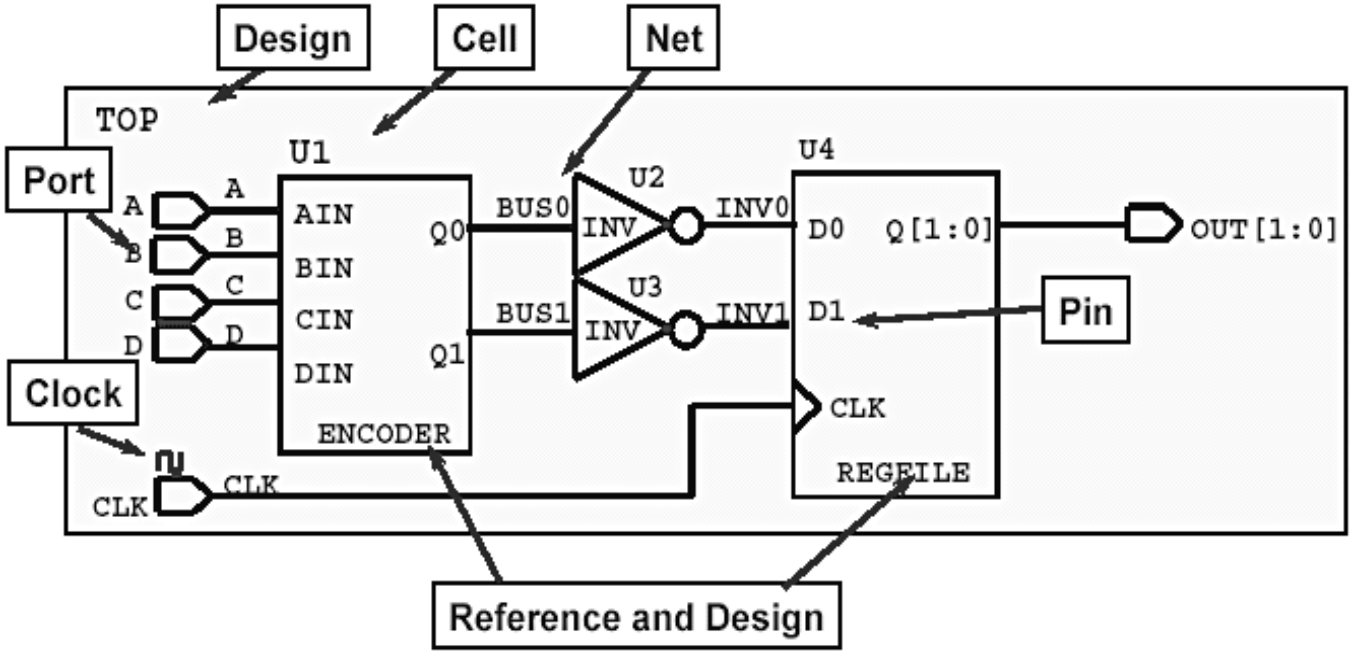
- Design: A circuit that performs one or more logical functions
- Cell: An instance of a design or library primitive within a design
- Reference: The name of the original design that a cell instance “points to”
- Port: The input or output of a design
- Pin: The input or output of a cell
- Net: The wire that connects ports to pins and/or pins to each other
- Clock: Waveform applied to a port or pin identified as a clock source

Design Objects (Schematic Perspective)

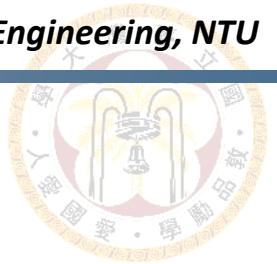


- Design: A circuit that performs one or more logical functions
- Cell: An instance of a design or library primitive within a design
- Reference: The name of the original design that a cell instance “points to”
- Port: The input or output of a design
- Pin: The input or output of a cell
- Net: The wire that connects ports to pins and/or pins to each other
- Clock: Waveform applied to a port or pin identified as a clock source

Design Objects (Schematic Perspective)



- Design: {TOP ENCODER REGFILE INV}
- Cell: {U1 U2 U3 U4}
- Reference: {ENCODER REGFILE INV}
- Port: {A B C D CLK OUT[1] OUT[0]}
- Pin: {U1/AIN U1/BIN U1/CIN U1/Q0 ...}
- Net: {A B C D CLK BUS0 BUS1 INV0 ...}
- Clock: {CLK}



Outline

- Introduction
- RTL Coding Related to Synthesis
- **Synopsys Environment**
  - Tools Used in Logic Synthesis
  - Design Objects
  - DC-TCL: Introduction
- Synthesis Design Flow
- Gate-Level Simulation
- Files Demo

Basic of TCL (Tool Command Language)

- Frequently used syntaxes in control flow and operations
  - TCL script = sequence of commands
  - Used in .synopsys_setup.dc / my_script.tcl / DUT_syn.sdc
- DC-TCL command (recommended*)

```
dc_shell -f script.tcl
```

```
dc_shell
dc_shell> source script.tcl
```

- Some references
  - Official: www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html
  - Chinese: www.nhu.edu.tw/~cmwu/Lab/TCL.doc
  - English: jan.newmarch.name/ProgrammingUnix/tcl/tcl_tut.html

General Syntax of Commands in DC

- General syntax

```
command [value] [options] [object_list]
```

- Examples

```
set_max_capacitance 0.1
```

```
set_ideal_network          [get_ports clk]
```

```
set_output_delay    0.5    [all_outputs]
```

```
compile -map_effort high -gate_clock -incremental_mapping
```

```
create_clock -name clk -period 10 -waveform {0 5} [get_ports clk]
```

Design Objects Calling

- Search the current design and return a list of given object types
- get_
  - get_cells, get_designs, get_libs, get_lib_cells
  - get_nets
  - get_pins, get_ports
  - get_clocks

Example

```
set_clock_transition 0.1 [get_clocks clk]
set_ideal_network    [get_ports clk]
```

- all_
  - all_instance
  - all_fanin, all_fanout, all_connected
  - all_inputs, all_outputs
  - all_registers

Example

```
set_load             0.05    [all_outputs]
set_drive            0.05    [all_inputs]
```

TCL Syntax in DC

- Termination by newline or " ; "
- Separation by " \ "

```
set_load \
0.05 \
[all_outputs]
```

- Comments by #
  - Only valid in the beginning of a line
  - Error Example

```
read_verilog top.v #read file
```

- Wildcard characters
  - "*" any character(s), e.g. u*e = use or usage …
  - "?" any single character, e.g. u?e = use…

- Help: list all commands with given name

```
dc_shell> help *clock*
```

- Get help/manual about command

```
dc_shell> compile -help
```

```
dc_shell> man compile
```

TCL: Special Characters

Character	Meaning
$	Used to access the value of a variable.
( )	Used to group expressions.
[ ]	Denotes a nested command.
\	Used for escape quoting and as a line continuation character.
" "	Denotes weak quoting. Nested commands and variable substitutions still occur.
{ }	Denotes rigid quoting. There are no substitutions.
;	Ends a command.
#	Begins a comment.

Variable Setting: set Command

- set
- Syntax: $variable_name
- Variable name is letters, digits, underscores

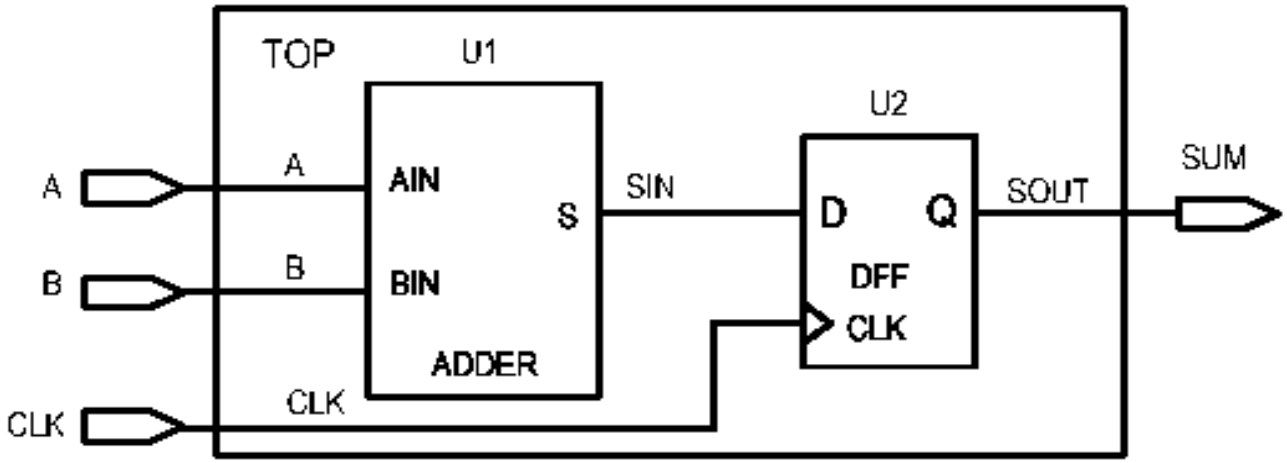
Command	Result
set b 66	66
set a b	b
set a $b	66

Example

```
dc_shell> set DESIGN "top"
dc_shell> current_design [get_designs ${DESIGN}]
Current design is 'top'.
dc_shell> check_design > "./${DESIGN}_check_design.txt"
```

top_check_design.txt 1 2022-

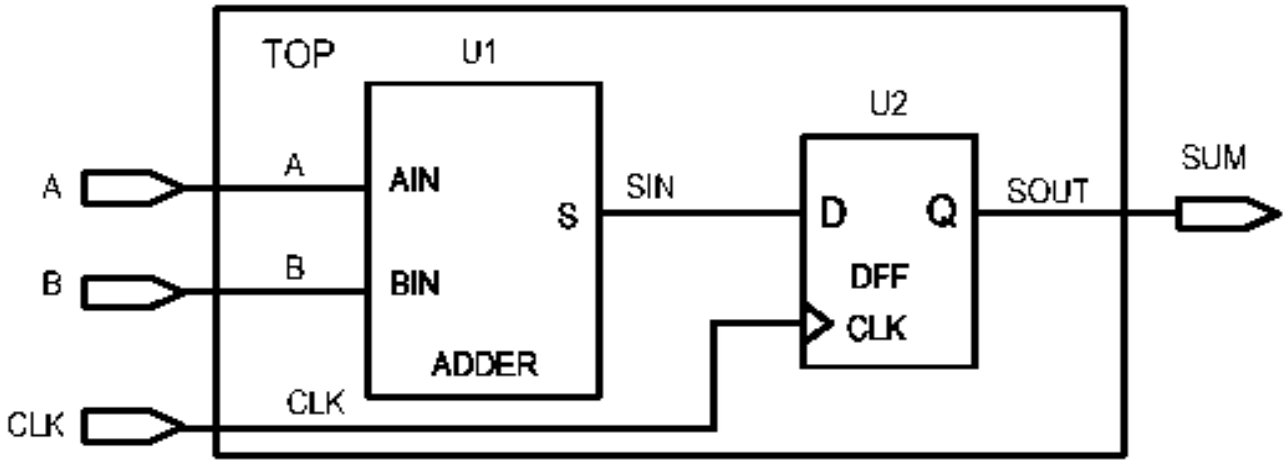
get_ & Wildcard Characters Exercise



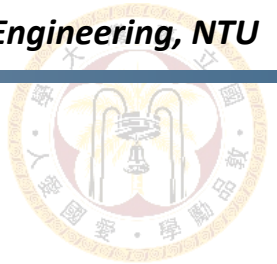
- Find a list of all the ports in the design?(`get_ports`)
- Find a list of all the cells that have the letter "U" in their name? (`get_cells *U*`)
- Find a list of all the nets ending with "CLK"? (`get_nets *CLK`)
- Find a list of all the "Q" pins in the design? (`get_pins */Q`)
- Find a list of all the references? (`get_references`)

Pins must appear w/ cell

get_ & Wildcard Characters Exercise



- Find a list of all the ports in the design?(`{A B CLK SUM}`)
- Find a list of all the cells that have the letter "U" in their name? (`{U1 U2}`)
- Find a list of all the nets ending with "CLK"? (`{CLK}`)
- Find a list of all the "Q" pins in the design? (`{U2/Q}`)
- Find a list of all the references? (`{ADDER DFF}`)



Outline

- Introduction
- RTL Coding Related to Synthesis
- Synopsys Environment
- **Synthesis Design Flow**
- Gate-Level Simulation
- Files Demo

What is Logic Synthesis?

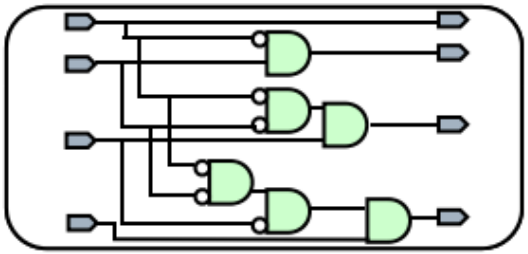
Synthesis = translation + optimization + mapping

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

HDL Source
(RTL)

Translate (HDL Compiler)

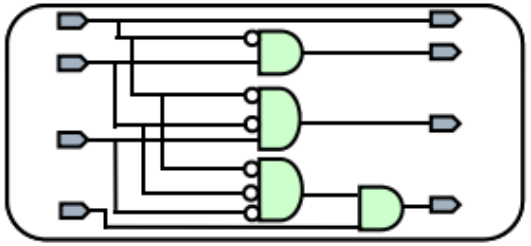
No Timing Info.



Generic Boolean
(GTECH)

Optimize + Mapping
(Design Compiler)

Timing Info.
Constraints



Target Technology

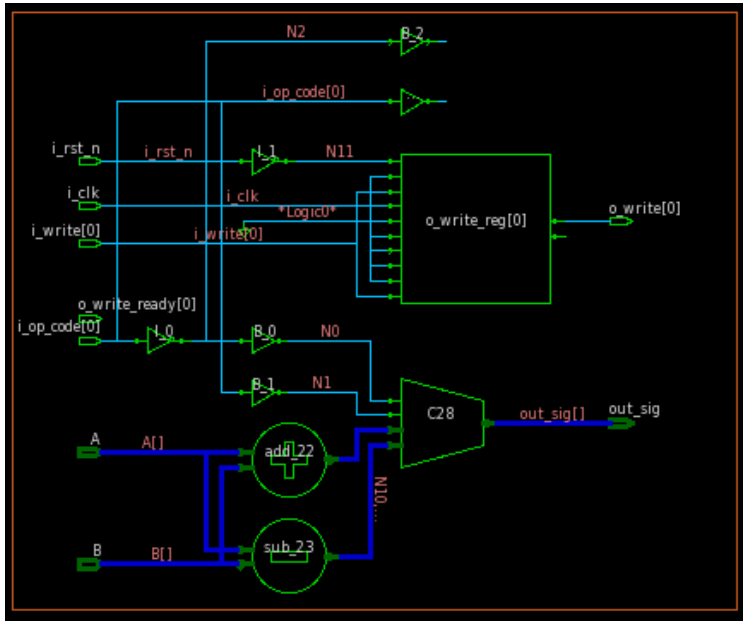
Translation

- HDL Compiler translates Verilog HDL descriptions into **GTECH**
- GTECH (generic tech.): a built-in intermediate format in Design Compiler
  - **Technology-independent**
  - **No timing information**
  - +, –, *, / **operators not unfolded** to gate level

```
module alu (
    ...
    assign y = a + b;
    assign o = s? y:z;
    ...
endmodule
```

RTL

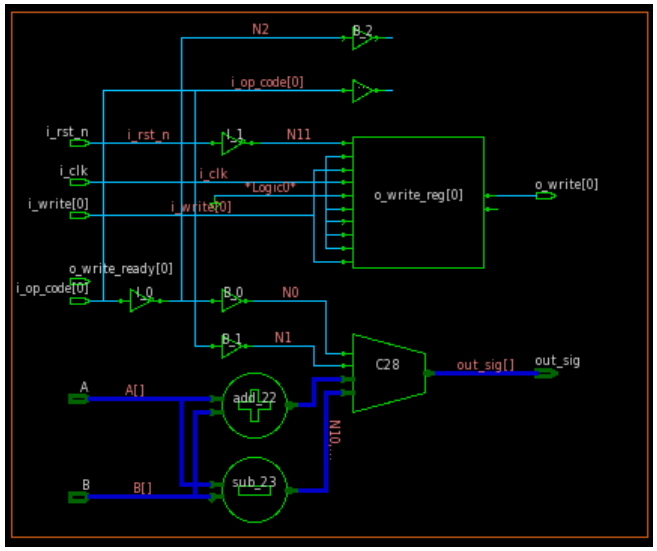
Translate



GTECH

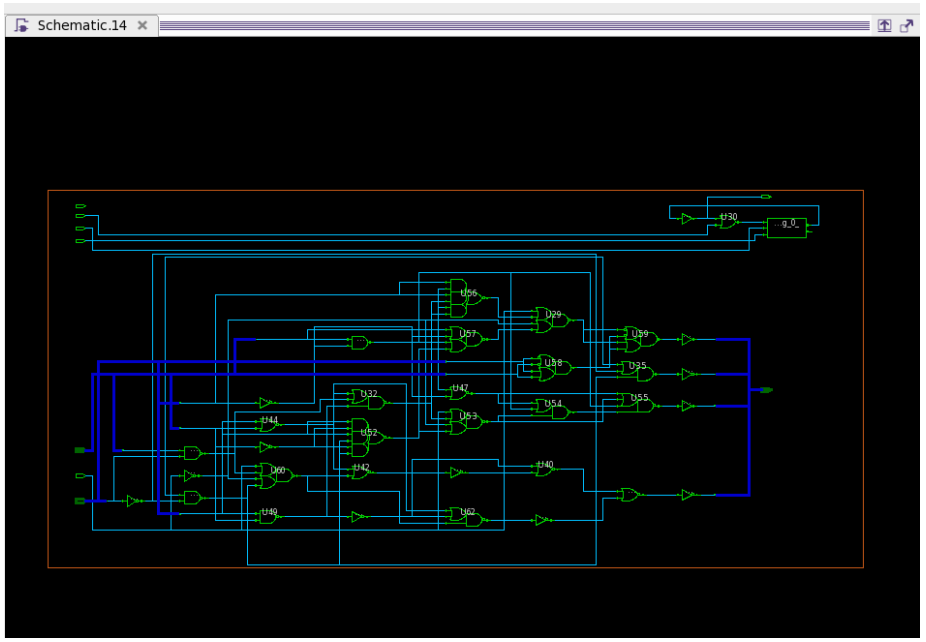
Optimization + Mapping

- Design Compiler maps GTECH to **gate-level netlist**
  - Using cells from user-specified technology library (e.g., 0.18um TSMC)
  - Iteratively optimized with area/timing constraints



GTECH

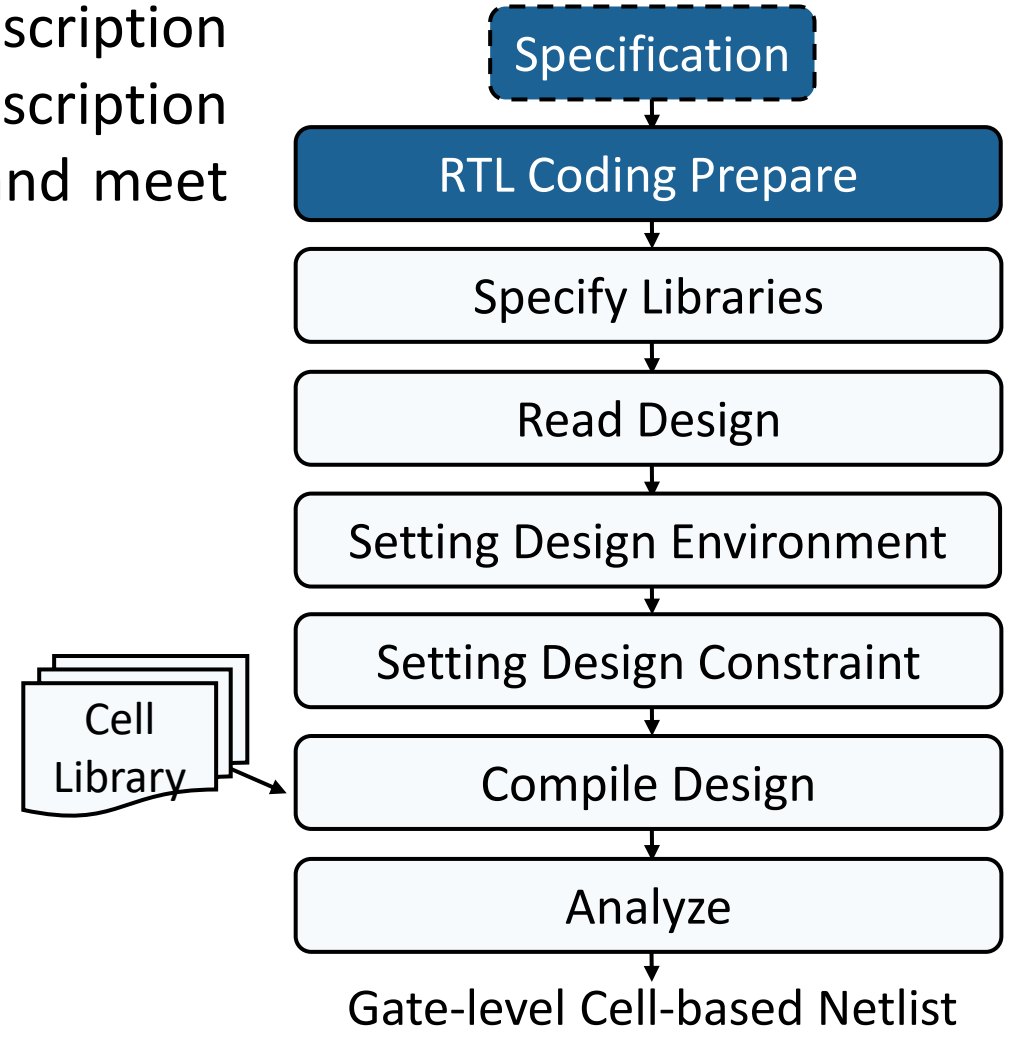
Optimize
Mapping



Gate-level Netlist

Synthesis Design Flow

- Develop the HDL design description and simulate the design description to verify that it is correct and meet the specification.



Synthesis Design Flow

- Specify libraries in **.synopsys_dc.setup**

Example files:

Technology library files:

`fast.db` `slow.db`

`typical.db`

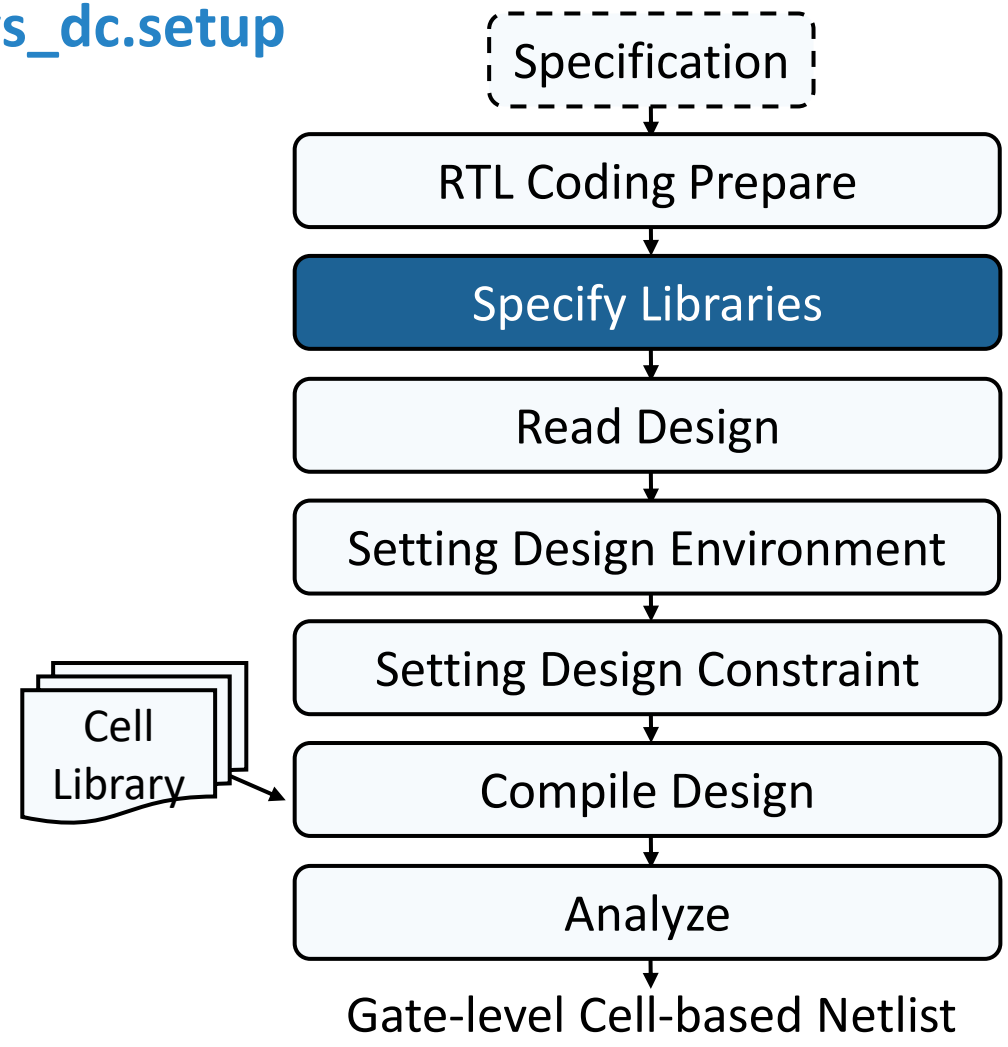
DesignWare library file:

`dw_foundation.sldb`

Other macro file:

`sram.db`

```
set link_library
set target_library
set symbol_library
set synthetic_library
```



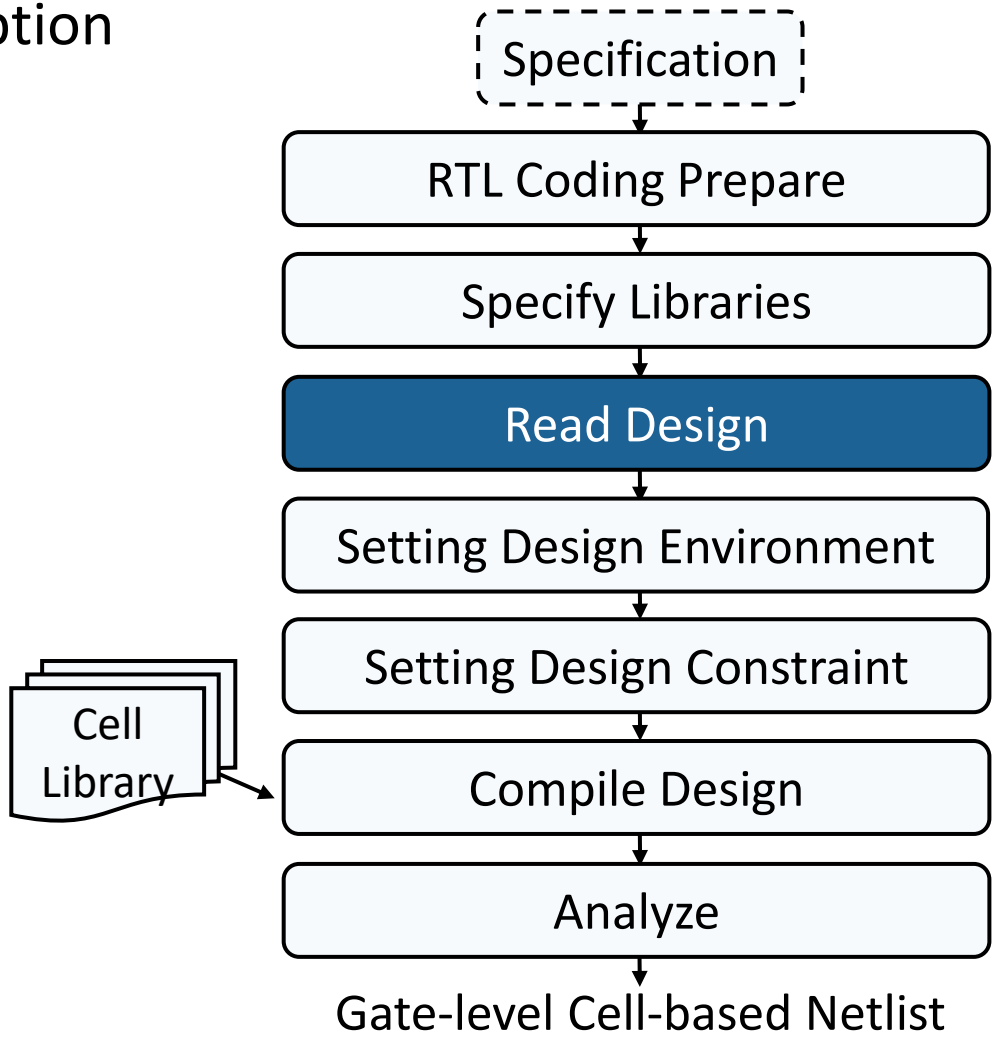
Synthesis Design Flow

- Read the HDL design description
- Check synthesizability
- Perform Presto compilation

```
analyze
elaborate
```

or

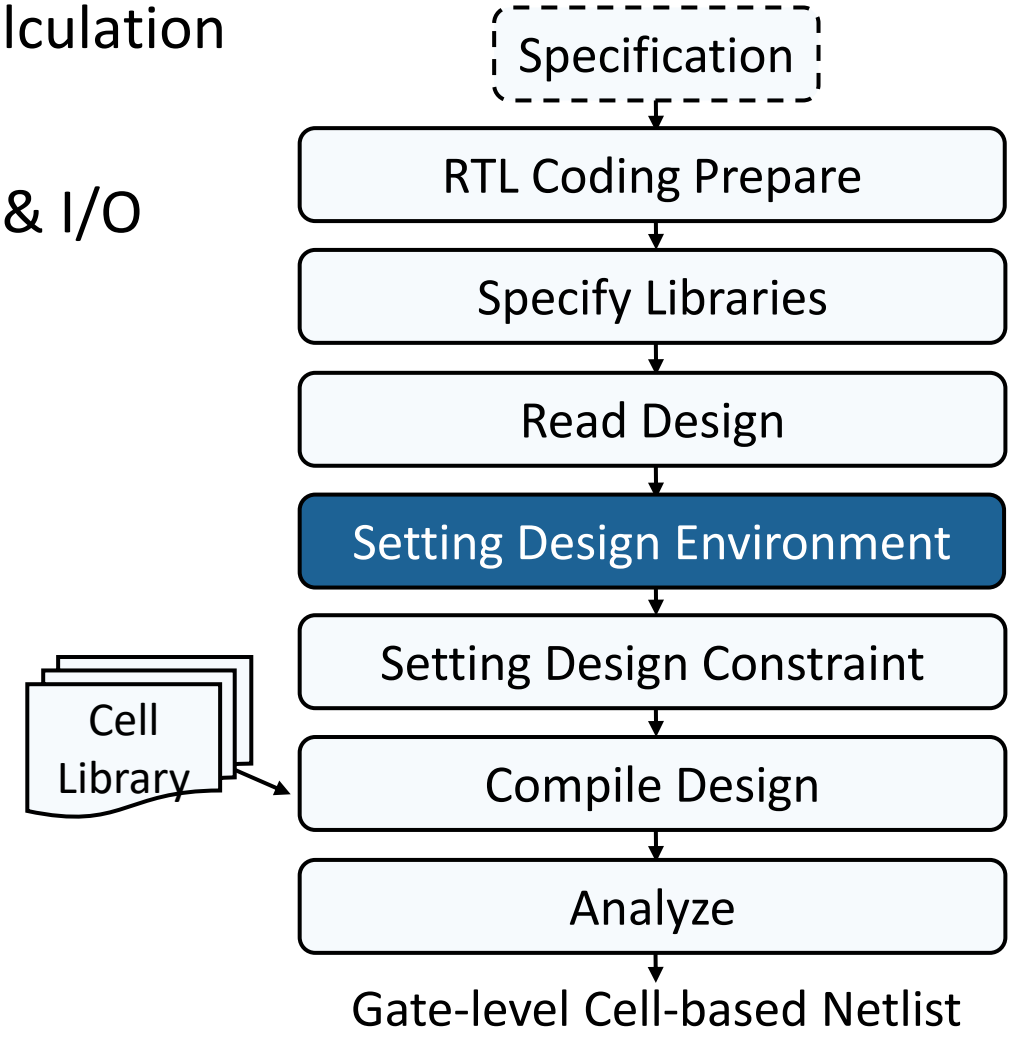
```
read_file
```



Synthesis Design Flow

- Set parameters for delay calculation
  - Operating condition
  - RC information of wires & I/O
  - I/O delay

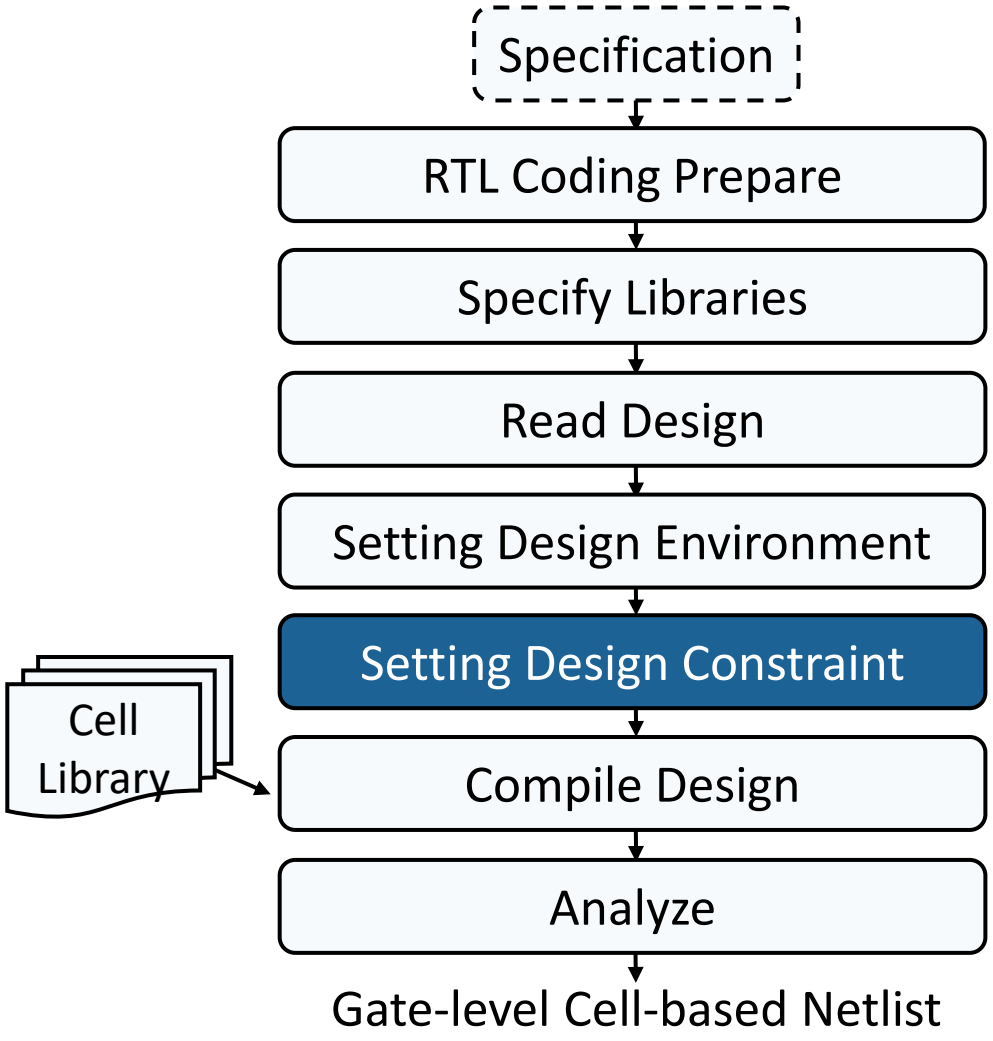
```
set_operating_conditions
set_wire_load_model
set_drive
set_load
set_input_delay
set_output_delay
```



Synthesis Design Flow

- Set constraints for synthesis
  - Design rule constraints
  - Clock constraints
  - Special circuits
  - Optimization objective

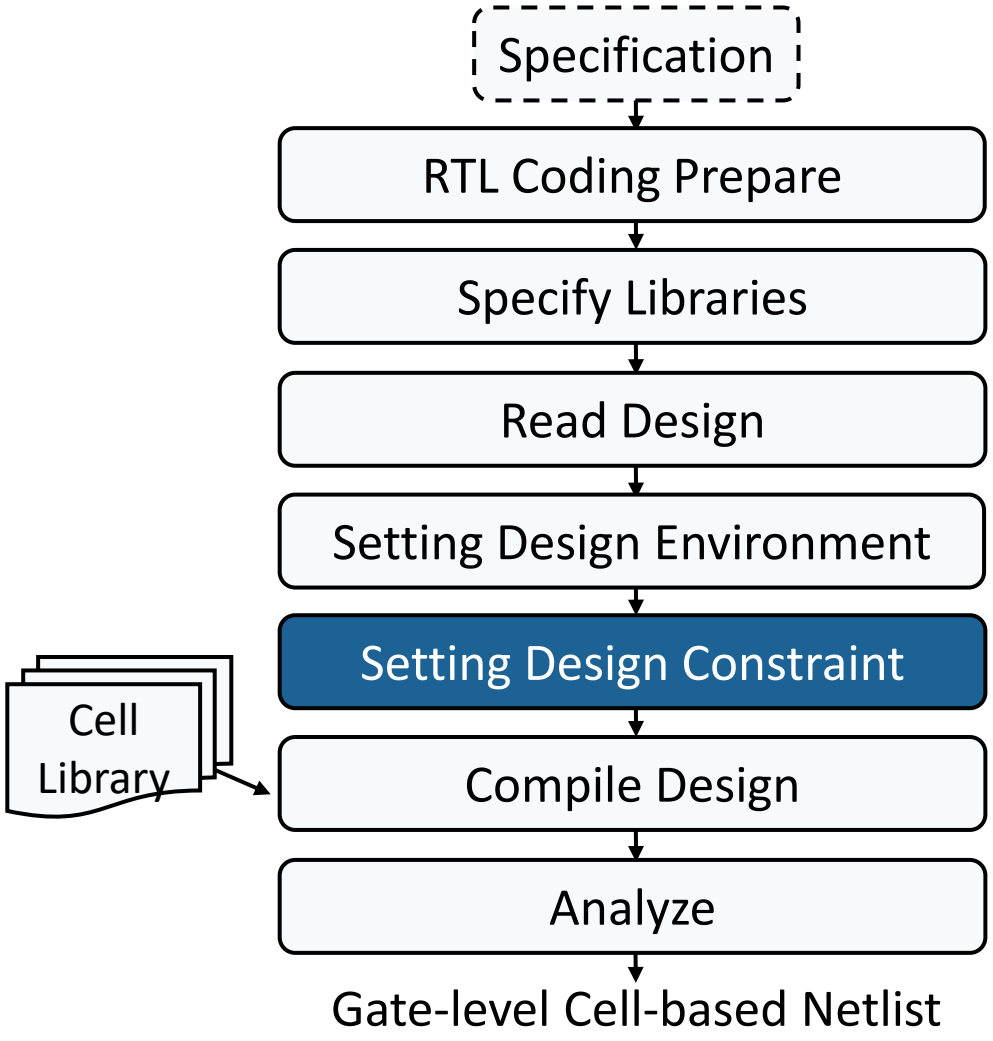
```
# Design Rule Constraints
set_max_transition
set_max_fanout
set_max_capacitance
```



Synthesis Design Flow

- Set constraints for synthesis
  - Design rule constraints
  - Clock constraints
  - Special circuits
  - Optimization objective

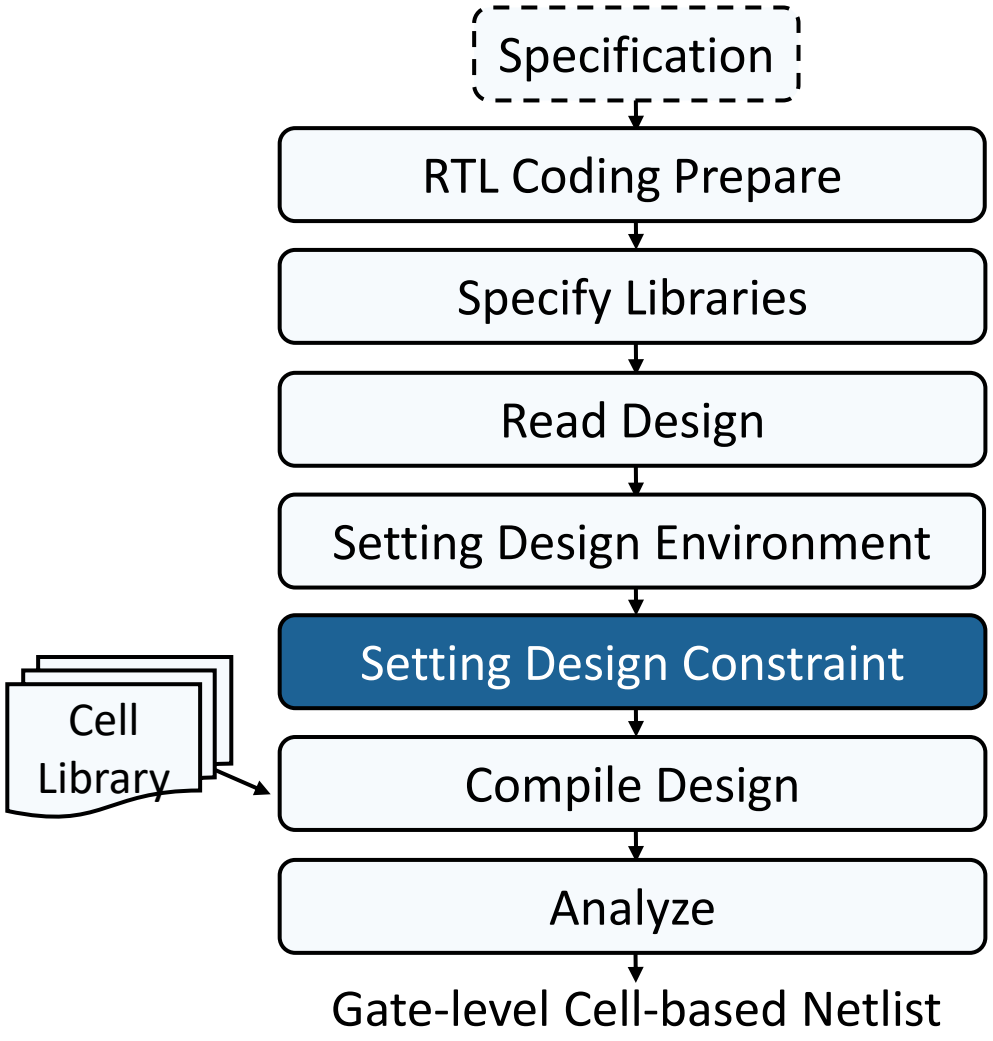
```
# Clock Constraints
create_clock
set_fix_hold
set_dont_touch_network
set_ideal_clock
set_clock_latency
set_clock_uncertainty
set_input_transition
set_clock_transition
```



Synthesis Design Flow

- Set constraints for synthesis
  - Design rule constraints
  - Clock constraints
  - Special circuits
  - Optimization objective

```
# Special Circuits
set_false_path
replace_clock_gates

# Optimization Objective
set_max_area
```



Synthesis Design Flow

- Optimization & mapping to gate-level netlist

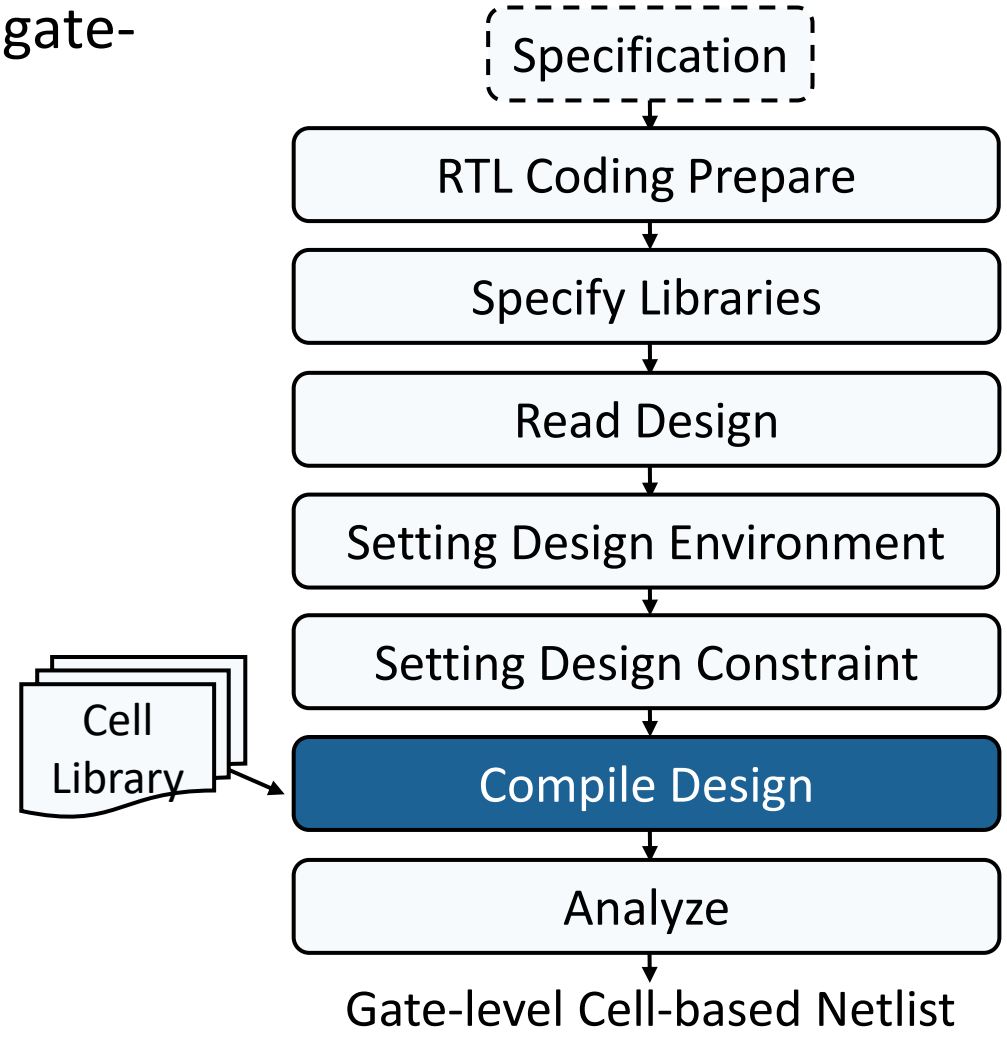
```
compile
```

or

```
compile -inc
```

or

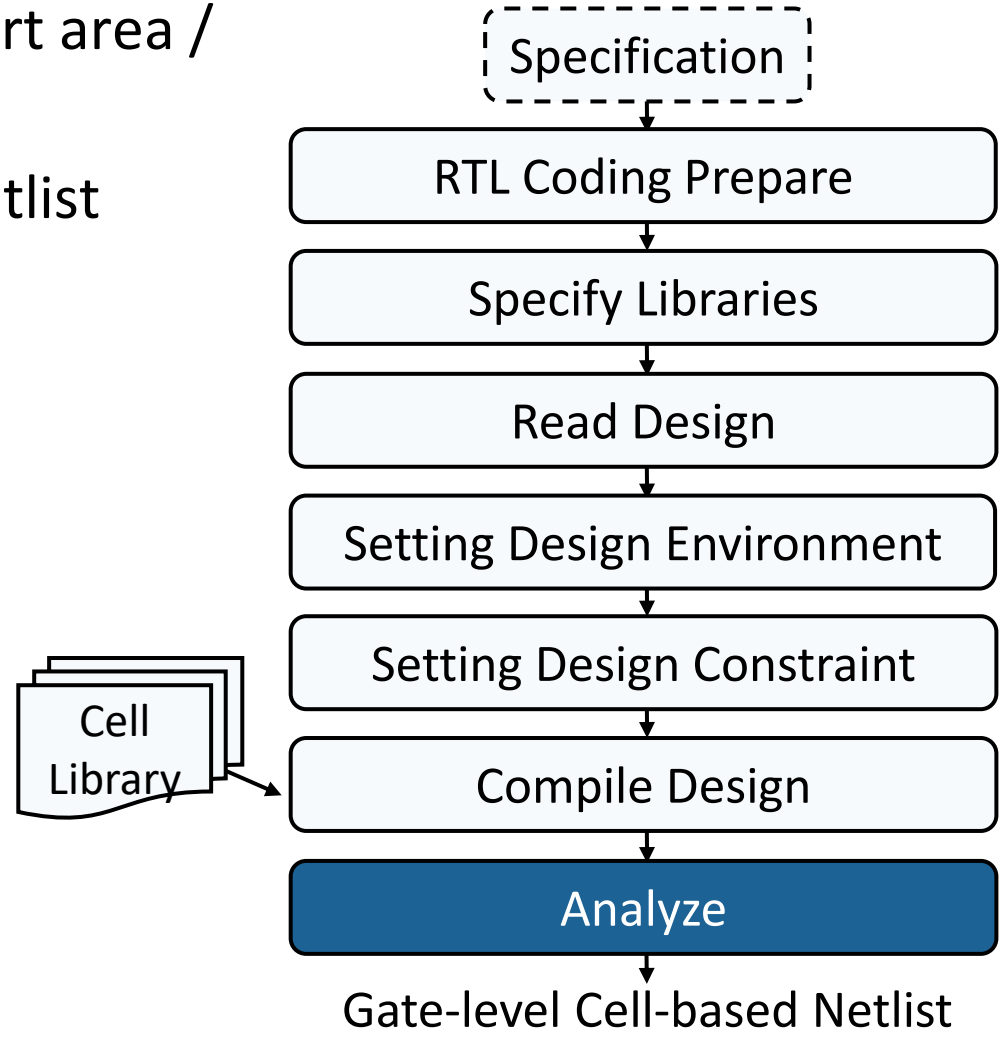
```
compile_ultra
```



Synthesis Design Flow

- Check design and then report area / constraints / timing issue
- Output files of gate-level netlist

```
check_design
report_area
report_constraint
report_timing

write_sdc
write_sdf
write -format verilog
```



Synthesis Design Flow

- Specify libraries in **.synopsys_dc.setup**

Example files:

Technology library files:

`fast.db` `slow.db`

`typical.db`

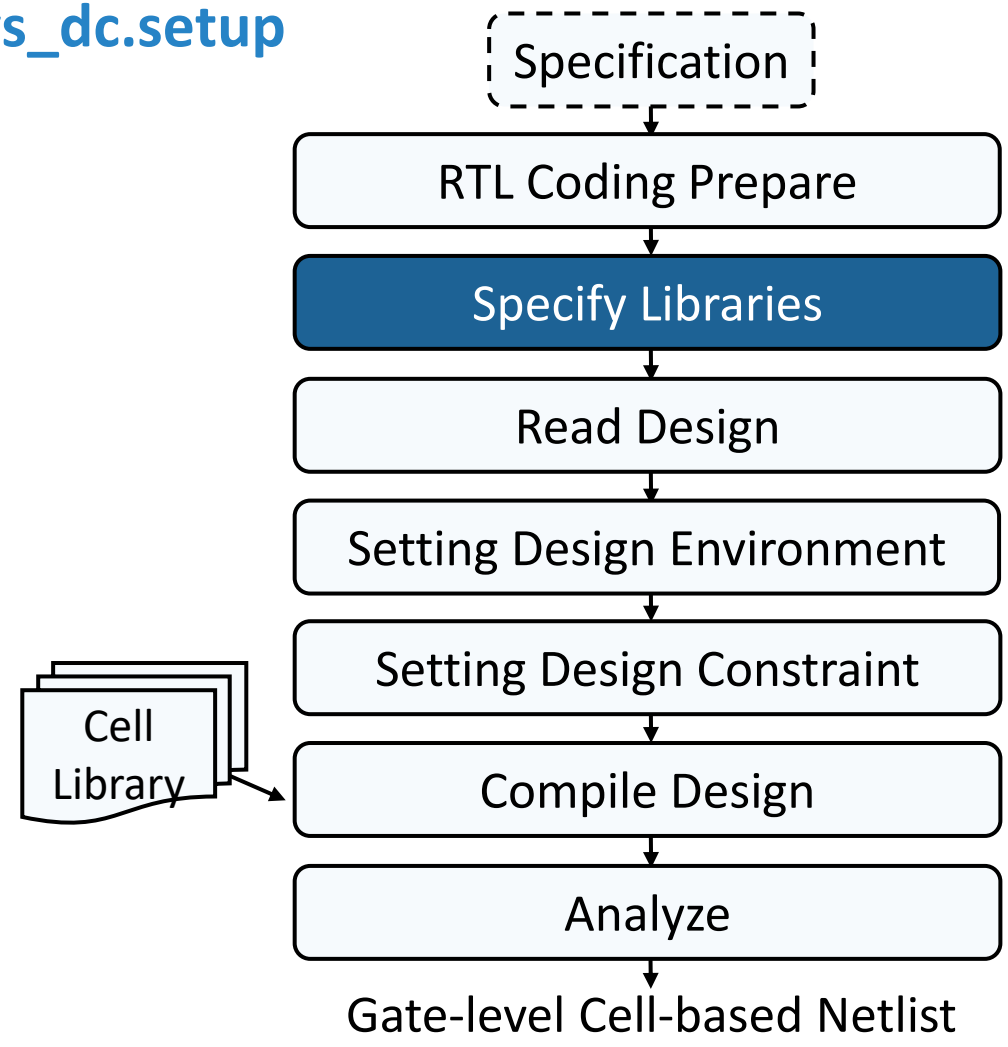
DesignWare library file:

`dw_foundation.sldb`

Other macro file:

`sram.db`

```
set link_library
set target_library
set symbol_library
set synthetic_library
```



Technology Library

- Technology library includes information of standard cells
  - Operating conditions
    - Process, voltage, temperature
  - Wire load model
  - Standard cell descriptions
    - Functionality, pin info.
    - Timing, area, power info.
  - Etc.

Same set of cells, different PVT conditions

```
fast.db
typical.db
slow.db
```

Technology Library

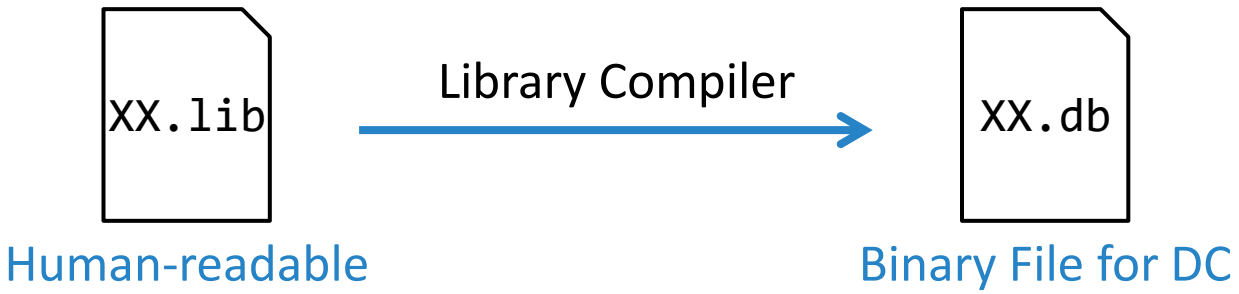
- Standard cell descriptions
  - Combinational cell & sequential cell list of the technology
  - Detailed **area, power, timing, function** info. of each cell
- Provides basic elements to form gate-level netlist

```
cell (NAND2BX1) {
  cell_footprint : nand2b;
  area : 6.789600;
  pin(AN) {
    direction : input;
    capacitance : 0.001359;
  }
  pin(B) {
    direction : input;
    capacitance : 0.002767;
  }
  pin(Y) {
    direction : output;
    capacitance : 0.0;
    function : "(!(!AN B))";
    internal_power() {
      related_pin : "AN";
      rise_power(energy_template_7x7) {
        index_1 ("0.042, 0.066, 0.112, 0.206, 0.392, 0.764, 1.5");
        index_2 ("0.00079, 0.002054, 0.00474, 0.010112, 0.020856, 0.042186, 0.08532");
        values ( \
          "0.003768, 0.003806, 0.003858, 0.003891, 0.003848, 0.003657, 0.003192", \
          "0.003744, 0.003780, 0.003827, 0.003858, 0.003820, 0.003634, 0.003170", \
          "0.003750, 0.003777, 0.003815, 0.003845, 0.003812, 0.003634, 0.003179", \
          "0.003795, 0.003808, 0.003829, 0.003848, 0.003819, 0.003654, 0.003213", \
          "0.003976, 0.003976, 0.003975, 0.003967, 0.003926, 0.003768, 0.003340", \
          "0.004549, 0.004535, 0.004513, 0.004410, 0.004341, 0.004171, 0.003750", \
          "0.005815, 0.005774, 0.005726, 0.005691, 0.005496, 0.005223, 0.004791");
      }
      fall_power(energy_template_7x7) {
        index_1 ("0.042, 0.066, 0.112, 0.206, 0.392, 0.764, 1.5");
        index_2 ("0.00079, 0.002054, 0.00474, 0.010112, 0.020856, 0.042186, 0.08532");
        values ( \
          "0.005118, 0.005214, 0.005344, 0.005457, 0.005528, 0.005567, 0.005591", \
          "0.005100, 0.005197, 0.005326, 0.005439, 0.005511, 0.005551, 0.005576", \
          "0.005113, 0.005210, 0.005342, 0.005459, 0.005535, 0.005578, 0.005606", \
          "0.005164, 0.005254, 0.005380, 0.005493, 0.005572, 0.005620, 0.005652", \
          "0.005377, 0.005454, 0.005565, 0.005669, 0.005743, 0.005795, 0.005832", \
          "0.006123, 0.006079, 0.006074, 0.006168, 0.006224, 0.006269, 0.006305", \
          "0.007591, 0.007527, 0.007456, 0.007403, 0.007346, 0.007374, 0.007400");
      }
    }
    timing() {
      related_pin : "AN";
      timing_sense : positive_unate;
      cell_rise(delay_template_7x7) {
        index_1 ("0.042, 0.066, 0.112, 0.206, 0.392, 0.764, 1.5");
        index_2 ("0.00079, 0.002054, 0.00474, 0.010112, 0.020856, 0.042186, 0.08532");
        values ( \
          "0.103606, 0.115100, 0.137951, 0.181525, 0.266709, 0.434270, 0.772003", \
```

slow.lib

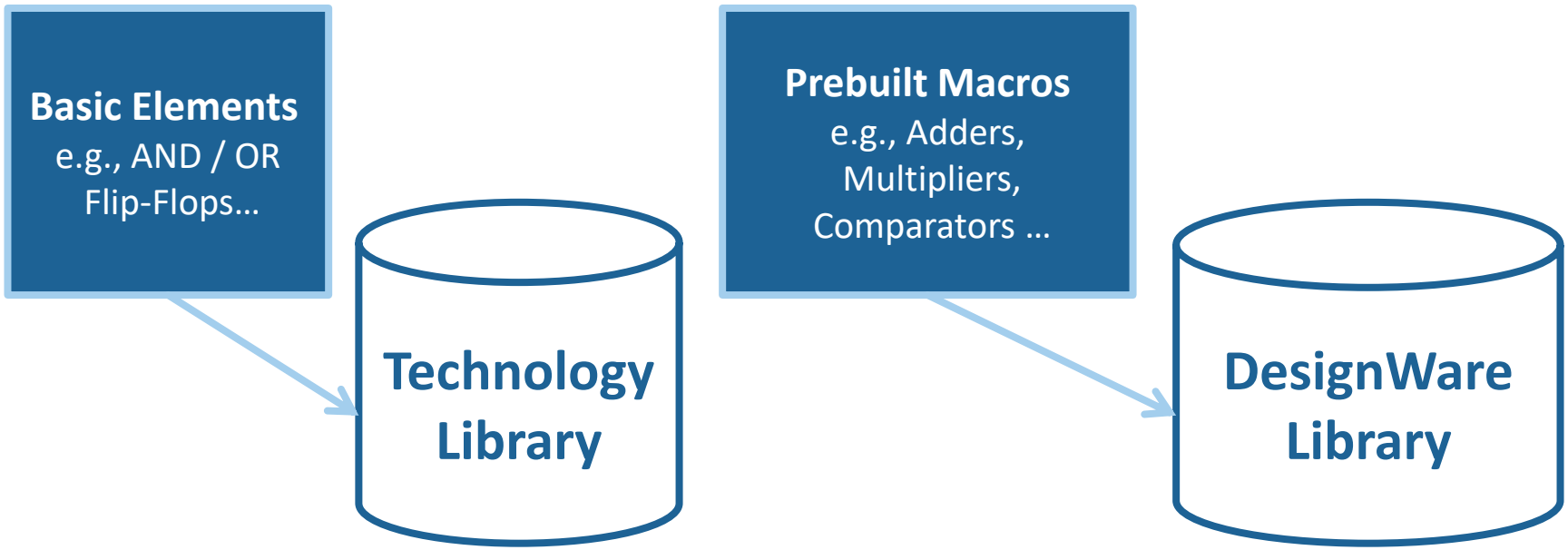
Library Compiler

- Library compiler (`lc_shell`) can generate .db file from .lib file
  - .lib contains information about cells
  - .db files are binary files for DC



DesignWare Library

- A **technology-independent** soft macros
- These macros can be synthesized into gates from target library
  - Usage 1: Multiple implementation for +,*,>,etc. operators
  - Usage 2: User directly instantiated building blocks

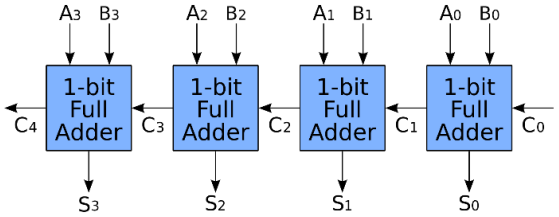


DesignWare Library

- Multiple architectures for each operators like +, –, *, /, <, etc.
  - In DC, these operators are mapped to DW library
  - DC selects appropriate implementation for your constraints
- Still can be optimized in compilation process

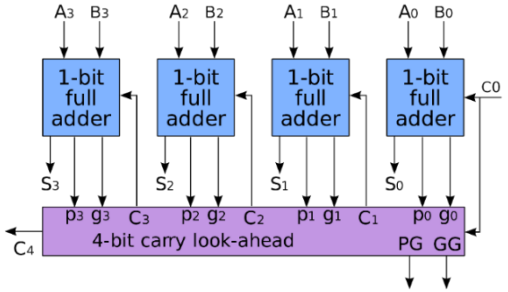
```
module alu (
    ...
    assign z = x + y;
    ...
endmodule
```

Area-constrained Design



DW Implementation 1:
Ripple Carry Adder – **Smaller**

Timing-constrained Design



DW Implementation 2:
Carry-lookahead Adder – **Faster**

DesignWare Building Blocks

- Besides basic operators, various building blocks are provided
  - E.g., float arithmetic, DSP, coding

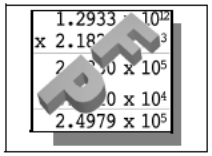
```
Path: /usr/cad/synopsys/synthesis/cur/dw/doc/manuals/dwbb_userguide.pdf
Website (Latest Ver.): https://www.synopsys.com/dw/buildingblock.php
```

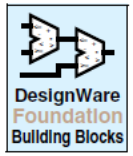
Table 2-1 List of DesignWare Building Block IP

Component	Inference?	Description
Application Specific: Control Logic		
DW_arb_2t	No	Two-Tier Arbiter with Dynamic/Fair-Among-Equal Scheme
DW_arb_dp	No	Arbiter with Dynamic Priority Scheme
DW_arb_fcfs	No	Arbiter with First-Come-First-Served Priority Scheme
DW_arb_rr	No	Arbiter with Round Robin Priority Scheme
DW_arb_sp	No	Arbiter with Static Priority Scheme
Datapath: Arithmetic Components		
DW01_absval	Function	Absolute Value
DW01_add	Operator	Adder
DW01_addsub	Operator	Adder-Subtractor
DW_addsub_dx	No	Duplex Adder/Subtractor with Saturation and Rounding

DesignWare Building Blocks

Path: /usr/cad/synopsys/synthesis/cur/doc/dw/dw_bb/datasheets/dw_fp_mult.pdf





DW_fp_mult

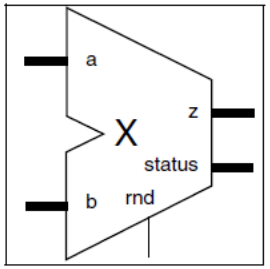
Floating-Point Multiplier

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- Hardware for denormal numbers of IEEE 754 standard is selectively provided.
- Configurable to be fully compliant with the IEEE Std 754-1985 standard
- Configurable for NaN representation compatible with the IEEE Std 754-2008 standard (controlled by the *ieee_compliance* parameter)
- DesignWare datapath generator is employed for better timing and area



Description

DW_fp_mult is a floating-point multiplier that multiplies two floating-point values, a and b, to produce a floating-point product, z.

Component pins are described in Table 1-1 and configuration parameters are described in Table 1-2.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*exp_width* + *sig_width* + 1 bits	Input	Multiplier
b	*exp_width* + *sig_width* + 1 bits	Input	Multiplicand
rnd	3 bits	Input	Rounding mode: supports all rounding modes described in the

DesignWare Building Blocks

- Example: instantiation in Verilog

Path: /usr/cad/synopsys/synthesis/cur/dw/examples/DW_fp_mult_inst.v
Website: https://www.synopsys.com/dw/doc.php/doc/dwf/examples/DW_fp_mult_inst.v

```
module DW_fp_mult_inst( inst_a, inst_b, inst_rnd, z_inst, status_inst );

parameter sig_width = 23;
parameter exp_width = 8;
parameter ieee_compliance = 1;


input [sig_width+exp_width : 0] inst_a;
input [sig_width+exp_width : 0] inst_b;
input [2 : 0] inst_rnd;
output [sig_width+exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_mult
    DW_fp_mult #(sig_width, exp_width, ieee_compliance)
              U1 ( .a(inst_a), .b(inst_b), .rnd(inst_rnd), .z(z_inst), .status(status_inst) );

endmodule
```

DesignWare Building Blocks

- Note: if DW building blocks are used in your design, remember to include DW behavior model in RTL simulation

```
ncverilog test_top.v top.v -y /usr/cad/synopsys/synthesis/cur/dw/sim_ver
+libext+.v +define+RTL +access+r +notimingchecks
```

```
vcs test_top.v top.v –full64 –R –dubug_access+all +v2k
+incdir+/usr/cad/synopsys/synthesis/cur/dw/sim_ver +notimingchecks
```

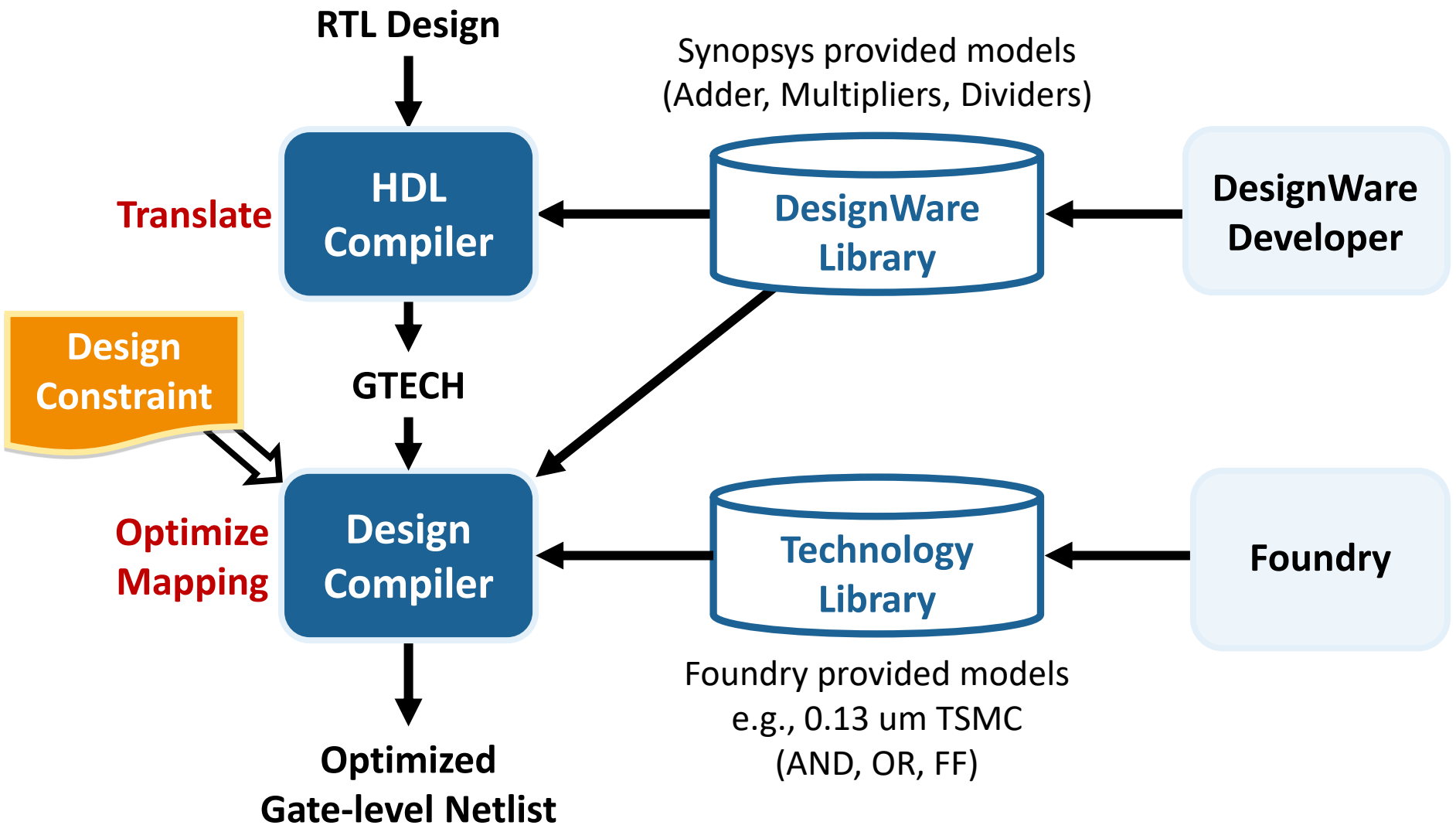
Including .db File for Hard Macros

- Hard macros (e.g., SRAM, IO pad, other external IPs)
  - Black box for design compiler, cannot be optimized
  - Note: soft macros (e.g., building blocks in DW library) can still be optimized in compilation

```
library(sram_256x8) {
    delay_model     : table_lookup;
    revision        : 1.1;
    date            : "2013-03-29 08:26:06Z";
    comment         : "Confidential Information of Artisan Components, Inc.  Use sub
ject to Artisan Components license. Copyright (c) 2013 Artisan Components, Inc.";
    time_unit               : "1ns";
    voltage_unit            : "1V";
    current_unit            : "1mA";
    leakage_power_unit      : "1mW";
    nom_process             : 1;
    nom_temperature         : 125.000;
    nom_voltage             : 1.080;
    capacitive_load_unit     (1,pf);
    pulling_resistance_unit  : "1kohm";
```

sram_256x8_slow_syn.lib

Libraries in Logic Synthesis



DC Library Setup: .synopsys_dc.setup

- .synopsys_dc.setup is automatically sourced when launching DC
  - Remember to put it in the directory where you launch DC

- The following variables are set
  - search_path
  - link_library
  - target_library
  - symbol_library
  - synthetic_library
  - Others

DC Library Setup: .synopsys_dc.setup

- **Search path**
  - Lists all the related directories of design and libraries

```
set search_path {CAD_path/CBDK013_TSMC_Artisan/CIC/SynopsysDC/db \
/home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/lib \
/home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db \
$search_path }
```

- **Link library**
  - Used to solve references, set operating conditions & wire load models, etc.
  - Usually set to technology libraries, macros & DW library

```
set link_library "typical.db slow.db fast.db dw_foundation.sldb"
```

- **Target library**
  - Provide timing, area, power info. for optimization
  - Usually set to technology libraries, macros

```
set target_library "typical.db slow.db fast.db"
```

DC Library Setup: .synopsys_dc.setup

- **Symbol library**
  - Defines symbols of schematic view for Design Vision

```
set symbol_library "generic.sdb"
```

- **Synthetic library**
  - Defines built-in operators in Synopsys

```
set synthetic_library "dw_foundation.sldb"
```

.synopsys_dc.setup Examples

Library setup example

```
set search_path  {CAD_path/CBDK013_TSMC_Artisan/CIC/SynopsysDC/db \
                  /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/lib \
                  /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db \
                 $search_path }
set link_library      "typical.db slow.db fast.db dw_foundation.sldb"
set target_library    "typical.db slow.db fast.db"
set symbol_library    "generic.sdb"
set synthetic_library "dw_foundation.sldb"
```

Library setup example (with SRAM)

```
set search_path  {CAD_path/CBDK013_TSMC_Artisan/CIC/SynopsysDC/db \
                  /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/lib \
                  /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db \
                 $search_path }
set link_library      "typical.db slow.db fast.db sram_256x8_slow_syn.db sram_256x8_typical_syn.db
                       sram_256x8_fast_syn.db dw_foundation.sldb"
set target_library    "typical.db slow.db fast.db sram_256x8_slow_syn.db sram_256x8_typical_syn.db
                       sram_256x8_fast_syn.db"
set symbol_library    "generic.sdb"
set synthetic_library "dw_foundation.sldb"
```

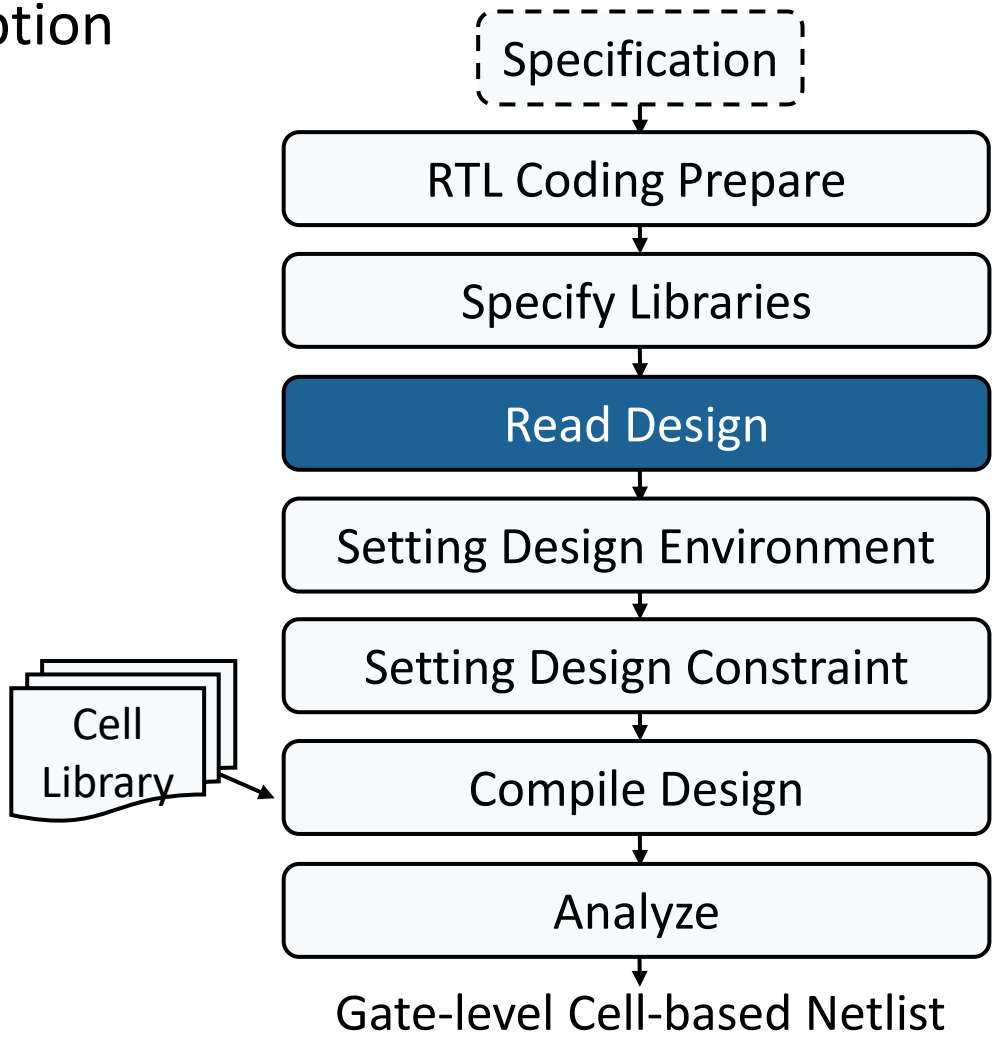
Synthesis Design Flow

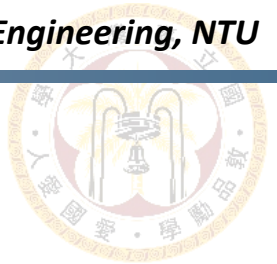
- Read the HDL design description
- Check synthesizability
- Perform Presto compilation

1. `analyze`
   `elaborate`

or

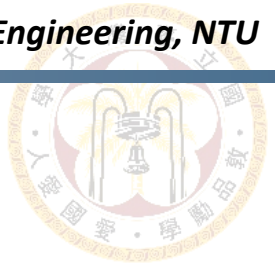
2. `read_file`





Reading Design

- Read netlists or design descriptions into Design Compiler
- Support different formats:
  - Verilog(.v) / VHDL(.vhd) / System Verilog(.sv)
  - Berkeley Espresso format(.pla)
  - Synopsys internal formats(.db, .dc)
- Method1
  - Analyze + Elaborate
- Method2
  - Read_file

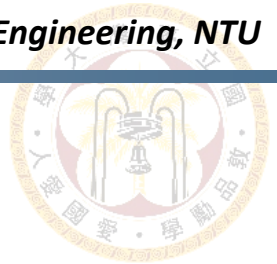


Analyze + Elaborate

- Analyze
  - Check VHDL & Verilog for syntax and synthesizability
  - Create intermediate .syn .mr .pvl files
- Elaborate
  - Translate intermediate objects into technology-independent design (**GTECH**)
  - Load parameters in HDL source files
  - Resolve design references

- ***File / Analyze*** + ***File / Elaborate***

```
analyze -format verilog {top.v fifo.v}
elaborate top
```



Read_file

- Read files with selected format
  - e.g., RTL design, gate-level netlist, .ddc, .db files
- Reading RTL: performs steps similar to analyze + elaborate
  - Does not create intermediate objects
  - Does not work with parameterized designs
- ***File/Read***

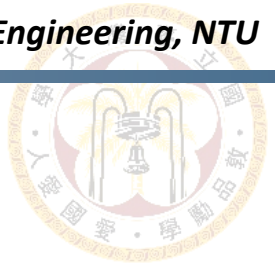
```
read_file -format verilog -rtl {top.v fifo.v}
```

```
read_verilog -rtl {top.v fifo.v}
```

Example of other usages:

```
read_file -format verilog -netlist $Gate_Level_file
```

```
read_file -format ddc $DDC_file
```



Reading Parameterized Design

- Read_file does not load parameters into designs
- Use analyze + elaborate for parameterized design

```
module top
...
fifo #(
    .DEPTH(8),
    .LENGTH(8)
    )u_fifo_0(
    ...
);

fifo #(
    .DEPTH(4),
    .LENGTH(2)
    )u_fifo_1(
    ...
);
...
endmodule
```

```
module fifo #(
    parameter DEPTH = 8,
    parameter LENGTH = 8
    )(
    input                 i_clk,
    input                 i_rst_n,

    input                 i_write,
    output                o_write_ready,
    input  [LENGTH-1:0]i_wdata,

    output                o_read_valid,
    input                 i_read,
    output [LENGTH-1:0]o_rdata
    );
    ...
endmodule
```

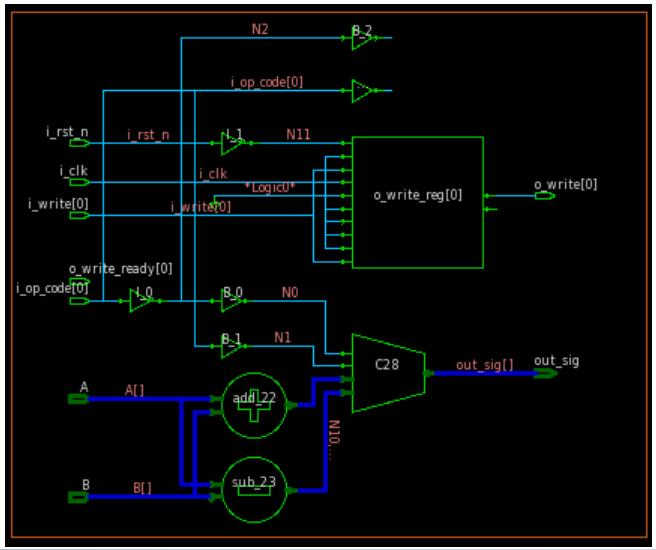
Presto Compilation

- HDL Compiler (Presto Verilog) is automatically invoked in `read_file` or `elaborate`
- GTECH netlist
  - No timing information
  - **Link library** & built-in **GTECH library** used
  - Flip-flops and +, =, *,… operators not unfolded to gate-level

```
module alu (
    ...
    assign y = a + b;
    assign o = s? y:z;
    ...
endmodule
```

RTL

Translate



GTECH

Check Latches

- Flip-flops and latches are listed in Presto Compilation
  - If there are latches, check your RTL design (usually incomplete `if-else` or `case` statements)

```
Inferred memory devices in process
        in routine fifo line 55 in file
                '/home/raid7_2/user09/r09176/CVSD/exp6_sram/fifo.v'.
=================================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=================================================================================
|     wpos_r_reg      | Flip-flop |   4   |  Y  | N  | Y  | N  | N  | N  | N  |
|    buffer_r_reg     | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
|     rpos_r_reg      | Flip-flop |   4   |  Y  | N  | Y  | N  | N  | N  | N  |
=================================================================================
Statistics for MUX_OPs
==================================================
| block name/line  | Inputs | Outputs | # sel inputs |
==================================================
|     fifo/34      |   8    |    8    |      3       |
==================================================
Presto compilation completed successfully.
Current design is now '/home/raid7_2/user09/r09176/CVSD/exp6_sram/fifo.db:fifo'
Loaded 2 designs.
Current design is 'fifo'.
```

Check latches using design compiler

Set Current Design

- Only one design can be set as current design for optimization

```
set DESIGN "top"
current_design $DESIGN
```

After read_file

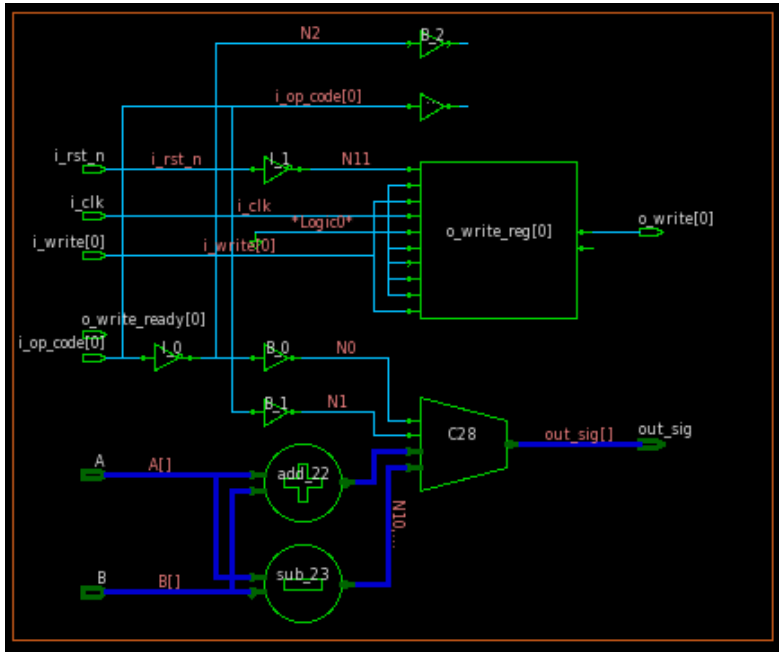
```
==========================================================
Presto compilation completed successfully.
Current design is now '/home/raid7_2/user09/r09176/CVSD/exp5_link/fifo.db:fifo'
Loaded 2 designs.
Current design is 'fifo'.
fifo top
```

Set Current Design

```
dc_shell> current_design [get_designs $DESIGN]
Current design is 'top'.
{top}
dc_shell>
```

Output GTECH Netlist (Optional)

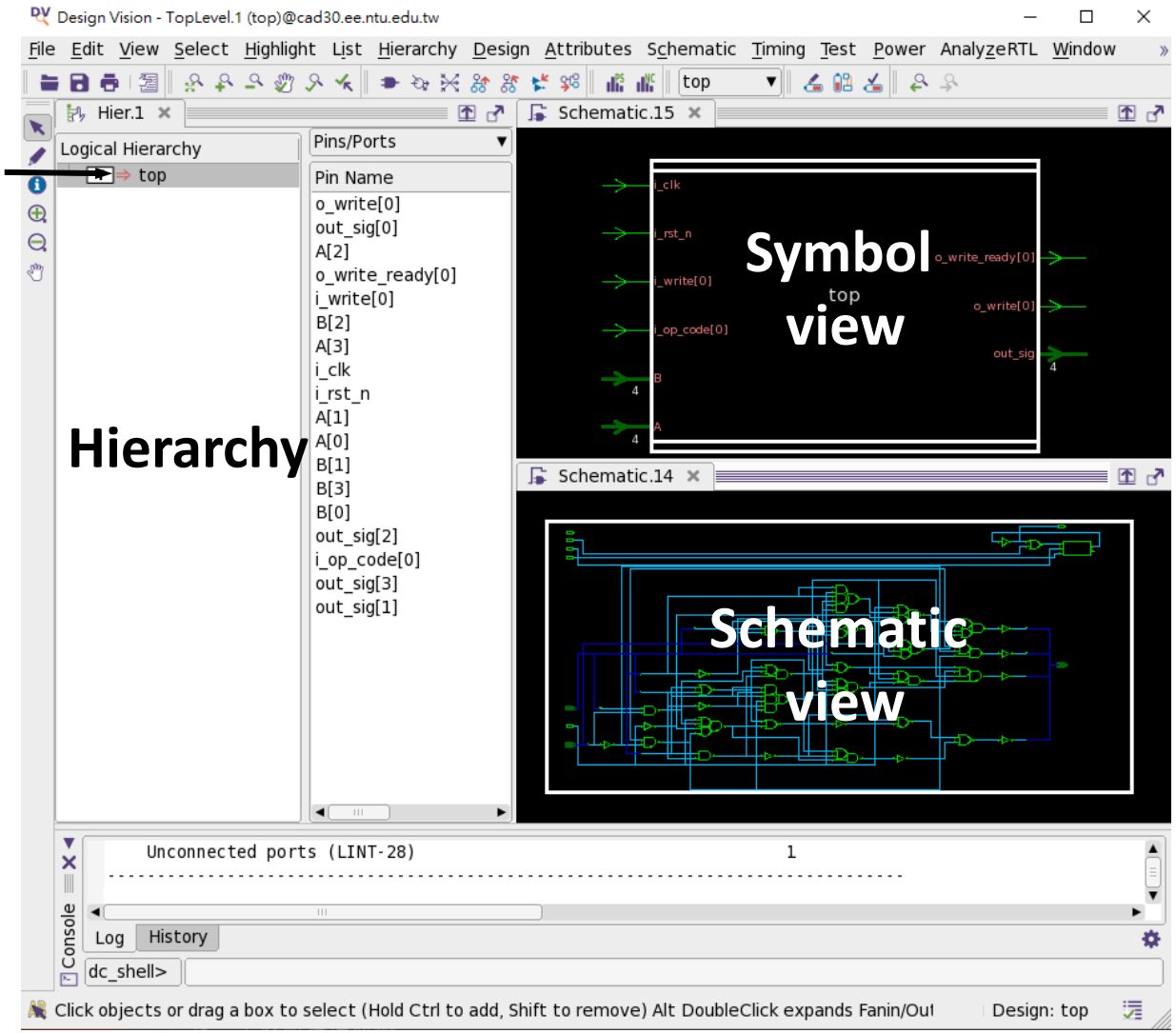
```
1  ////////////////////////////////////////////////////////////////
2  // Created by: Synopsys Design Compiler(R)
3  // Version   : R-2020.09-SP5
4  // Date      : Mon Feb 20 13:23:08 2023
5  ////////////////////////////////////////////////////////////////
6
7
8  module top ( i_clk, i_rst_n, i_write, o_write_ready, A, B, i_op_code, out_sig,
9          o_write );
10   input [0:0] i_write;
11   output [0:0] o_write_ready;
12   input [3:0] A;
13   input [3:0] B;
14   input [0:0] i_op_code;
15   output [3:0] out_sig;
16   output [0:0] o_write;
17   input i_clk, i_rst_n;
18   wire   N0, N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, N11;
19
20   \**SEQGEN**  \o_write_reg[0]  ( .clear(N11), .preset(1'b0), .next_state(
21         i_write[0]), .clocked_on(i_clk), .data_in(1'b0), .enable(1'b0), .Q(
22         o_write[0]), .synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(
23         1'b0), .synch_enable(i_write[0]) );
24   SUB_UNS_OP sub_19 ( .A(A), .B(B), .Z({N10, N9, N8, N7}) );
25   ADD_UNS_OP add_18 ( .A(A), .B(B), .Z({N6, N5, N4, N3}) );
26   SELECT_OP C28 ( .DATA1({N6, N5, N4, N3}), .DATA2({N10, N9, N8, N7}),
27         .CONTROL1(N0), .CONTROL2(N1), .Z(out_sig) );
28   GTECH_BUF B_0 ( .A(N2), .Z(N0) );
29   GTECH_BUF B_1 ( .A(i_op_code[0]), .Z(N1) );
30   GTECH_NOT I_0 ( .A(i_op_code[0]), .Z(N2) );
31   GTECH_BUF B_2 ( .A(N2) );
32   GTECH_BUF B_3 ( .A(i_op_code[0]) );
33   GTECH_NOT I_1 ( .A(i_rst_n), .Z(N11) );
34 endmodule
```




```
write -format verilog -hierarchy -output TOP_GTECH.v
```

Symbol / Schematic View (Optional)

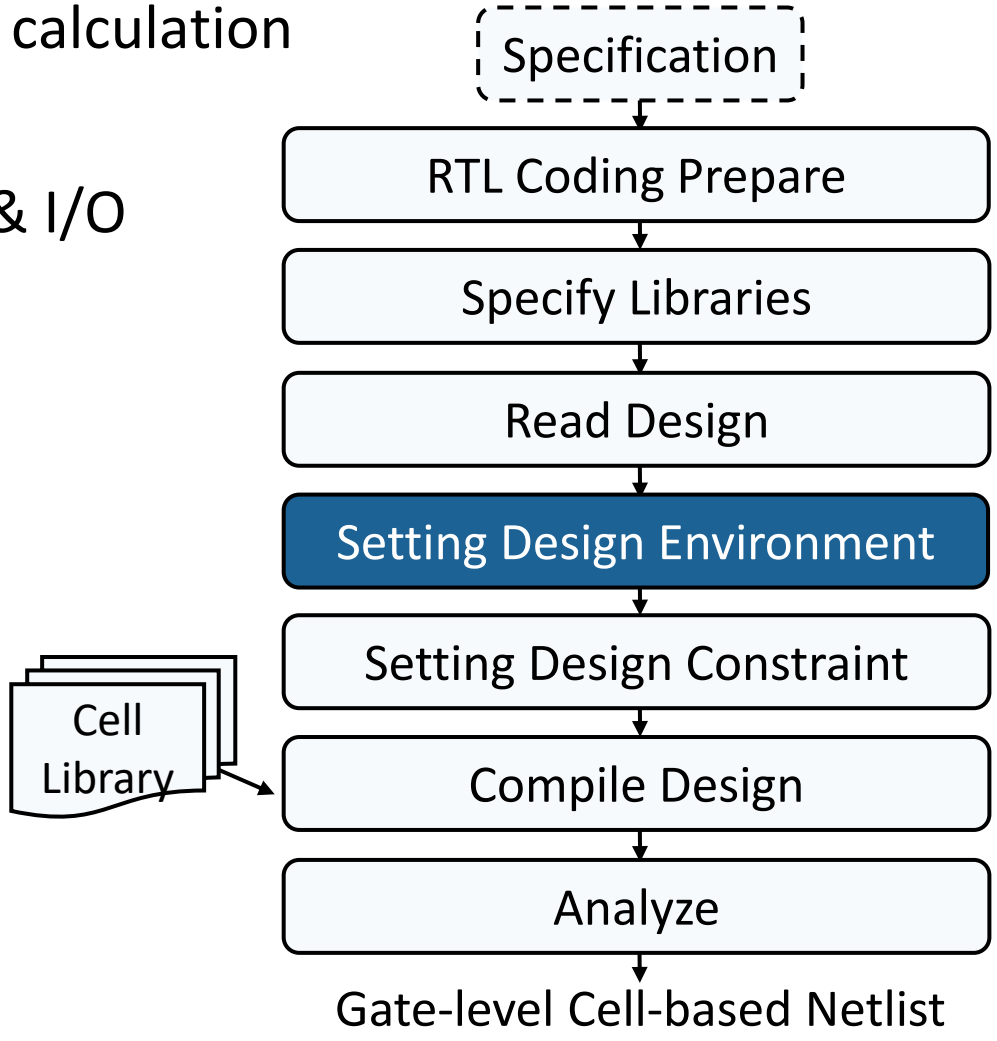
Current Design



Synthesis Design Flow

- Set parameters for RC delay calculation
  - Operating condition
  - RC information of wires & I/O
  - I/O delay

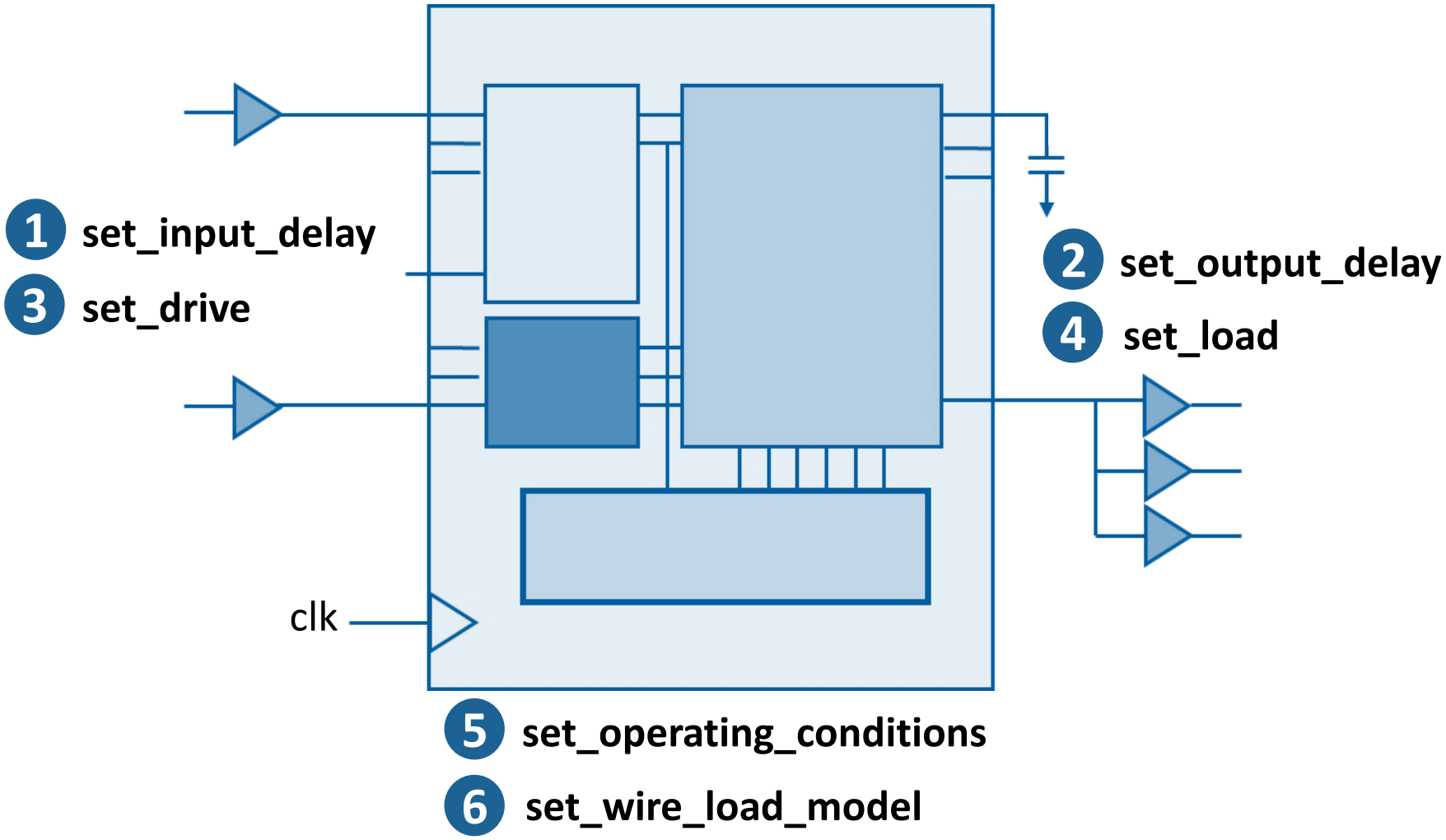
1. set_input_delay
2. set_output_delay
3. set_drive
4. set_load
5. set_operating_conditions
6. set_wire_load_model



Why Describes the Real World Environment

- 1. Operating condition
  - The operating condition affects the components selected from technology library and the timing through your design
- 2. Environment parameters
  - The real world environment you define describes the conditions that the circuit will operate within
- The defaults are not realistic conditions
  - Input drive is not infinite
  - Capacitive loading is usually not zero
  - Consider **process**, **voltage**, and **temperature** (**PVT**) variation

Describing Design Environment

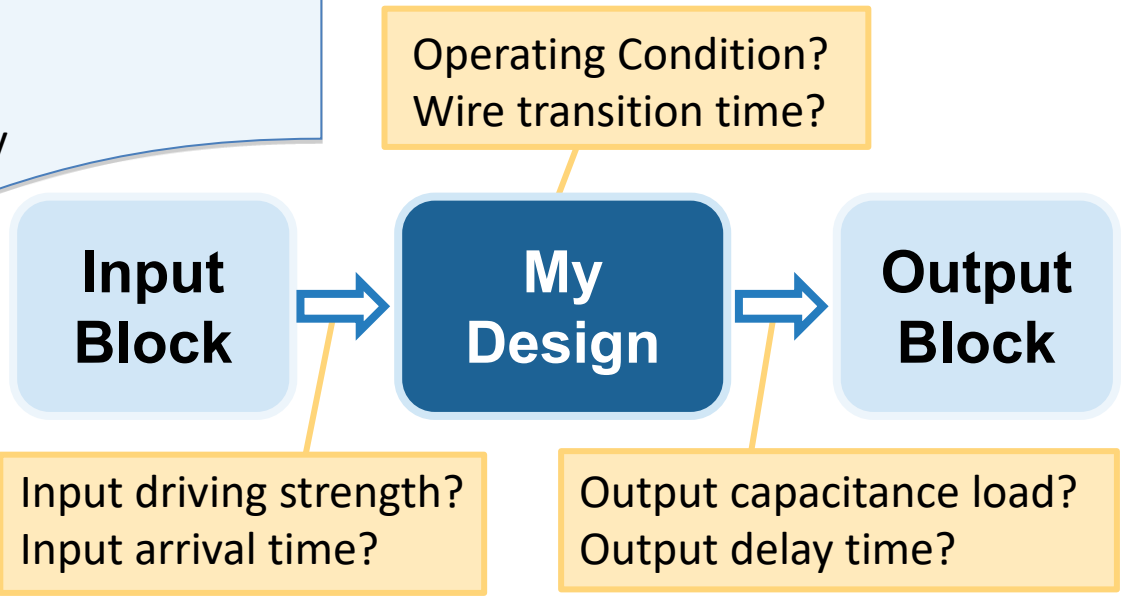


Describing Design Environment

For STA, every node should be known:

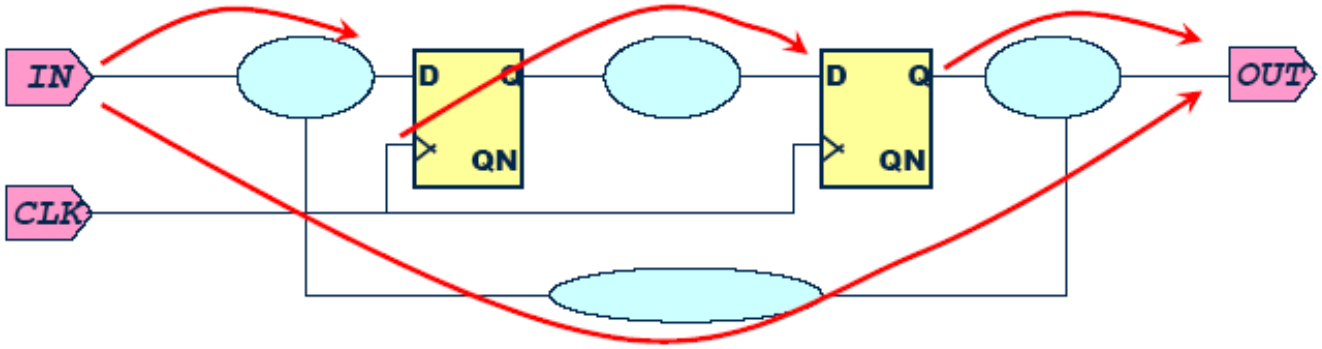
- **RC value**
  - Inside DUT: Tech lib
    1. standard cell/external-IP
    2. wire load model
  - Outside DUT:
    1. set_drive / set_driving_cell
    2. set_load
- **I/O delay**
  - set_input_delay
  - set_output_delay

STA: static timing analysis
DUT: device under test



Recall: Timing Paths

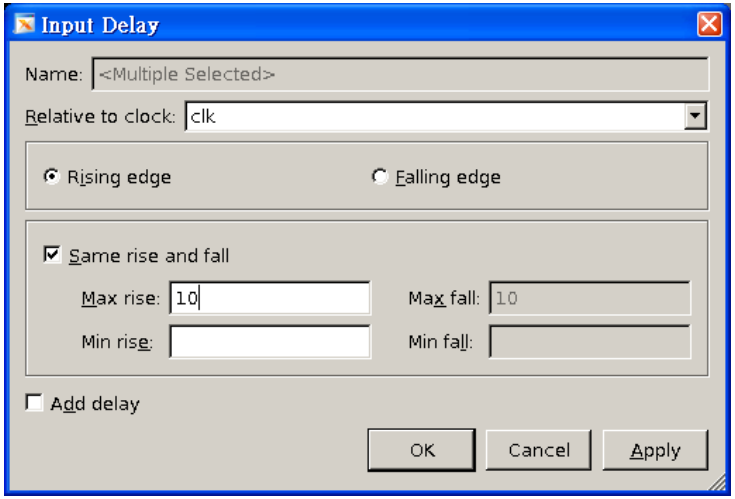
- Using inputs, outputs and registers as endpoints, the combinational part of design can be divided into 4 type of paths
  - Input to register
  - Register to register
  - Register to output
  - Input to output
- In the synthesized netlist, all paths should meet timing requirements
  - E.g., in a gate-level netlist synthesized for 200MHz, delay on all reg-to-reg paths should ≤ 5ns



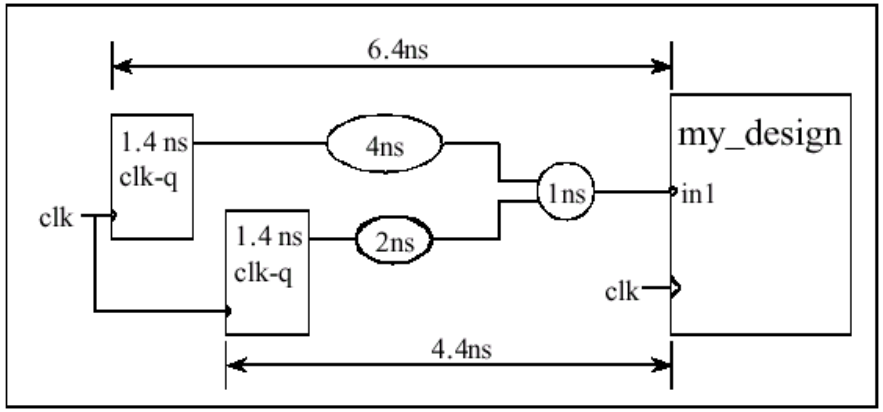
1. Setting Input Delay

- Input delay: input arrival time related to clock

- ***Attributes / Operating Environment / Input Delay***



◆ Example



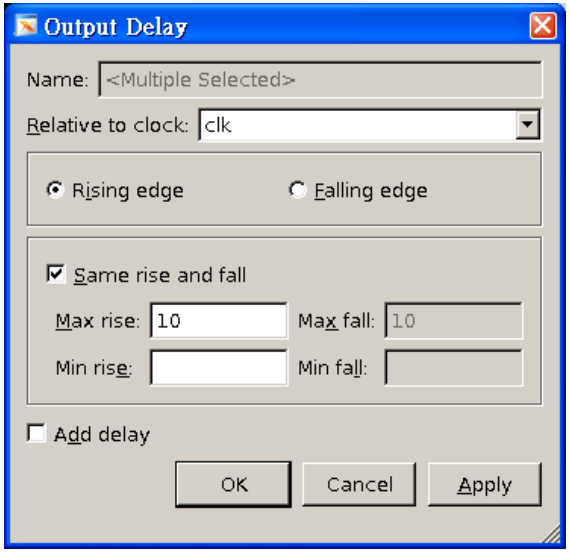

```
set_input_delay -clock clk -max 6.4 [get_ports in1]
set_input_delay -clock clk -min 4.4 [get_ports in1]
```

```
set_input_delay 2.6 -clock clk \
    [remove_from_collection  [all_inputs] [get_ports clk]]
```

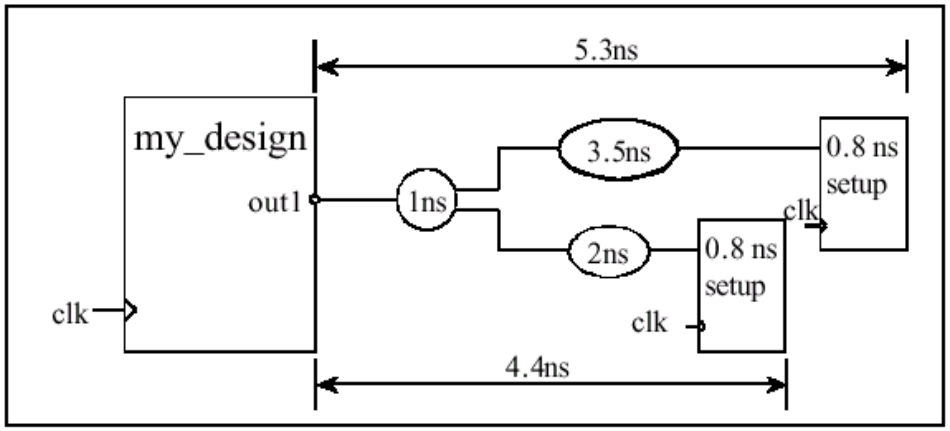
2. Setting Output Delay

- Output require time related to next clock

- ***Attributes / Operating Environment / Output Delay***



◆ Example

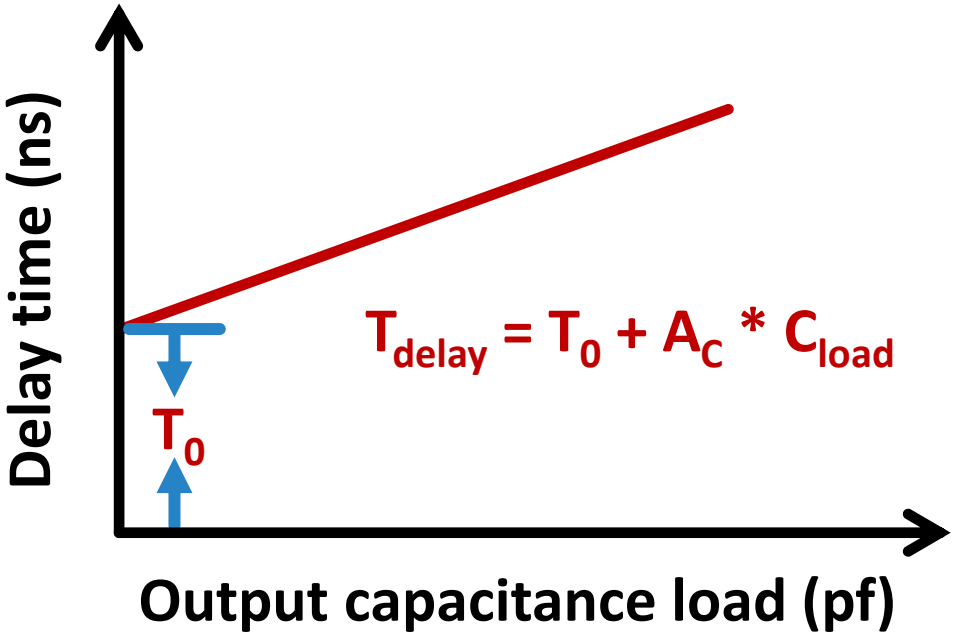



```
set_output_delay -clock clk -max 5.3 [get_ports out1]
```

```
set_output_delay 5.3 -clock clk [all_outputs]
```

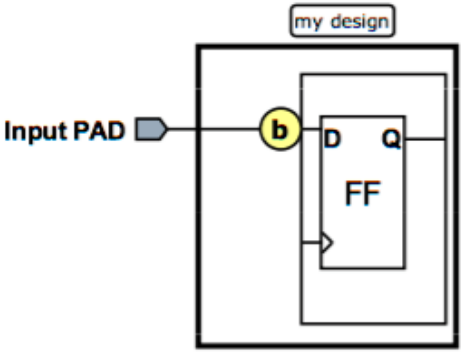
Input Drive Impedance

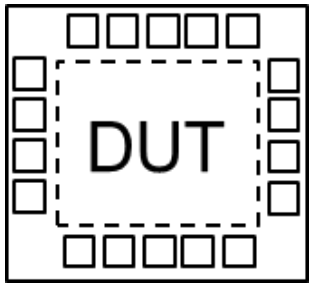
- $T_{delay} = T_0 + A_c * C_{load}$
  - T_0: cell pin to pin intrinsic delay
  - A_c: drive impedance (ns/pf)
  - C_{load}: output capacitance load (pf)

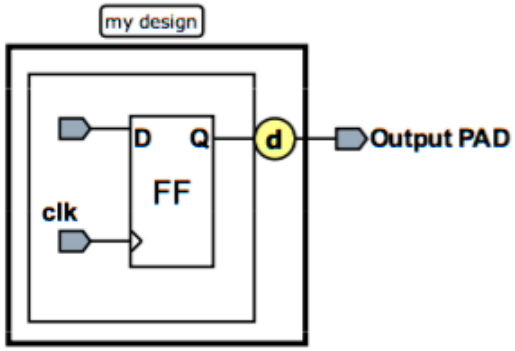


Drive Strength & Load for Pads

- Input Drive Strength for Pads
- Output Load for Pads







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

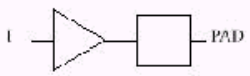
Performance Equations

		RISE	FALL
PC3D01			
tCD	PAD -> CIN	0.3377 + 0.0378 * 0.2468 * Cld	0.2596 + 0.0503 + 0.1978 * Cld
PC3D11			
tCD	PAD -> CIN	0.3761 + 0.0792 + 0.2265 * Cld	0.3787 + 0.0253 + 0.1960 * Cld
PC3D21			
tCD	PAD -> CIN	0.7315 + 0.0574 + 0.2538 * Cld	0.6064 + 0.0595 + 0.2163 * Cld
PC3D31			
tCD	PAD -> CIN	0.6528 + 0.0289 + 0.2236 * Cld	0.5971 + 0.0253 + 0.1976 * Cld

3V CMOS Output Pads

PC3O01 through PC3O05

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



Function Table

INPUT	OUTPUT
I	PAD
L	L
H	H

Cell Description

Macro Name:	PC3O01	PC3O02	PC3O03	PC3O04	PC3O05
Drive Capability	1x	2x	3x	4x	5x
Width (mils):	3.4	3.4	3.4	3.4	3.4
Power (μW/MHz):	198.55	198.59	198.75	198.81	197.58

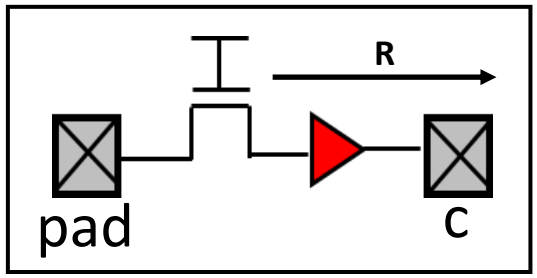
Pin Description

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

3 & 4. Input Drive Strength & Output Load for Pads

- Defined in technology library
- Consider for IO Pads (Synthesis w/IO)

TSMC18: PDIDGZ



- Setting Input Driving Strength

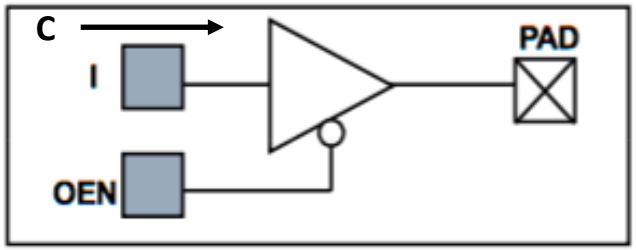
```
set_drive 0.05 [all_inputs]
```

```
set_driving_cell -library tpz973gvwc -lib_cell PDIDGZ -pin {C} [all_inputs]
```

- Setting Output Load

```
set_load [load_of "tpz973gvwc/PDT16DGZ/I"] [all_outputs]
```

- Remember to add ***.db file for IO Pad*** into link_library in .synopsys_setup



TSMC18: PDT16DGZ

Port Report

```
report_port -verbose
```

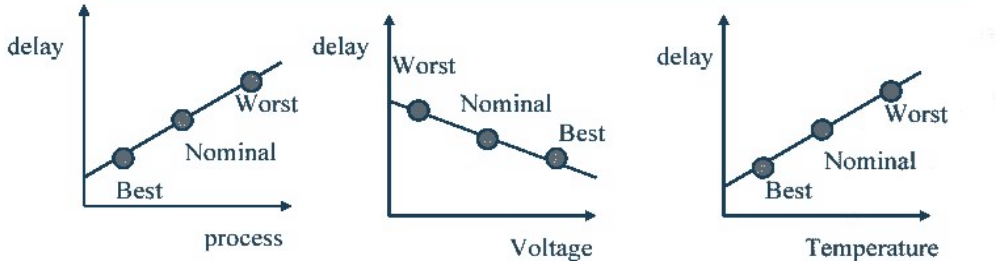
- or in the option menu set verbose

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


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


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

5. Setting Operating Condition

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

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

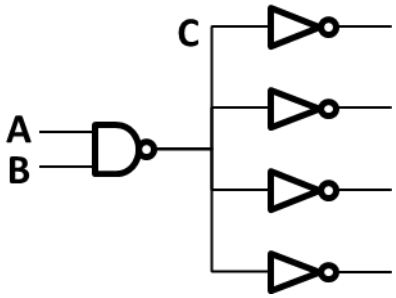


- General operating conditions in technology process
  - Best case (fast) / Typical case / **Worst case (slow)**
- ***Attributes / Operating Environment / Operating Condition***

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

Wire Load Model (WLM)

- Fanout of a wire/pin
  - The **number of load gates** connected to the output of the driving gate
- WLM contains information about
  - A lookup table to estimate wire length by its fanout
    - Extrapolation for fanouts not in list
  - Area/capacitance/resistance coefficients
- Note: WLM only provide a rough estimation, area/delay of wire will be more accurate in automatic place and route (APR) stage



Part of gate-level netlist

WLM

Fanout	Length
1	4.1
2	5.3
3	6.4
4	7.4

Area Coeff.	0.7
Cap. Coeff.	1e-4
Resist. Coeff.	8e-8

Fanout of C = 4

⬇

Estimated length = 7.4

⬇

Estimated Area = 7.4*0.7
Estimated Cap. = 7.4*1e-4
Estimated Resist. = 7.4*8e-8

WLM Example for tsmc13_wl10

- Example: calculating R, C and net area of a net with fanout = 3
  - Length = **66.667**x3
  - C_{wire} = (fanout=3) x capacitance coefficient
    - (length= **66.667**x3) x (**1.5x10^{-4}**) = 0.03 pf
  - R_{wire} = (fanout=3) x resistance coefficient
    - (length= **66.667**x3) x (**8.5x10^{-8}**) = 0.017 ohm
  - Net area = (fanout=3) x area coefficient
    - (length= **66.667**x3) x (**0.7**) = 140 um^2

slow.lib header

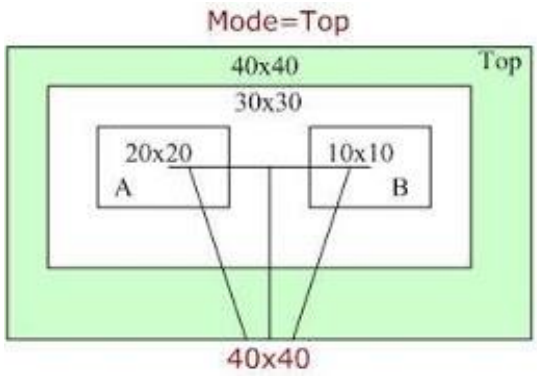
```
/* unit attributes */
time_unit: “1ns”;
voltage_unit: “1V”;
current_unit: “1mA”;
pulling_resistance_unit: “1kohm”;
leakage_power_unit: “1pw”;
Capacitive_load_unit(1.0, pf);
```

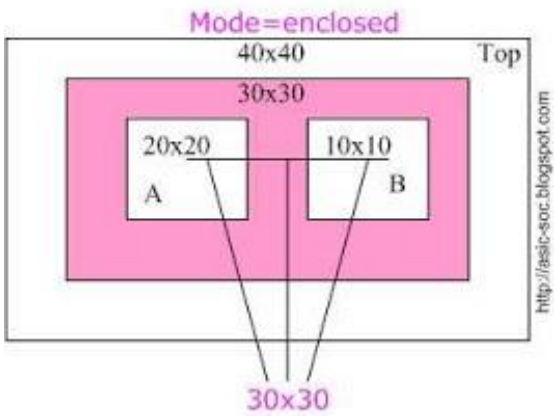
slow.lib

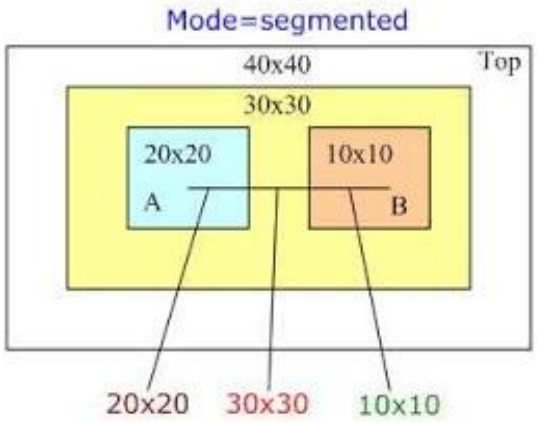
```
/* wire-loads */
wire_load(“tsmc13_wl10”){
   resistance : 8.5e-8;
   capacitance: 1.5e-4;
   area       : 0.7;
   slope      : 66.667;
fanout_length (1, 66.667);
}
```

Wire Load Modes

- 3 hierarchical wire load modes
- Top
  - Applying the wire load models specified for the top level of the design to all nets, the most rough estimation
- Enclosed
  - Using the wire load model which inherits from its upper-level design
- Segmented
  - Nets crossing hierarchical boundaries are divided into segments

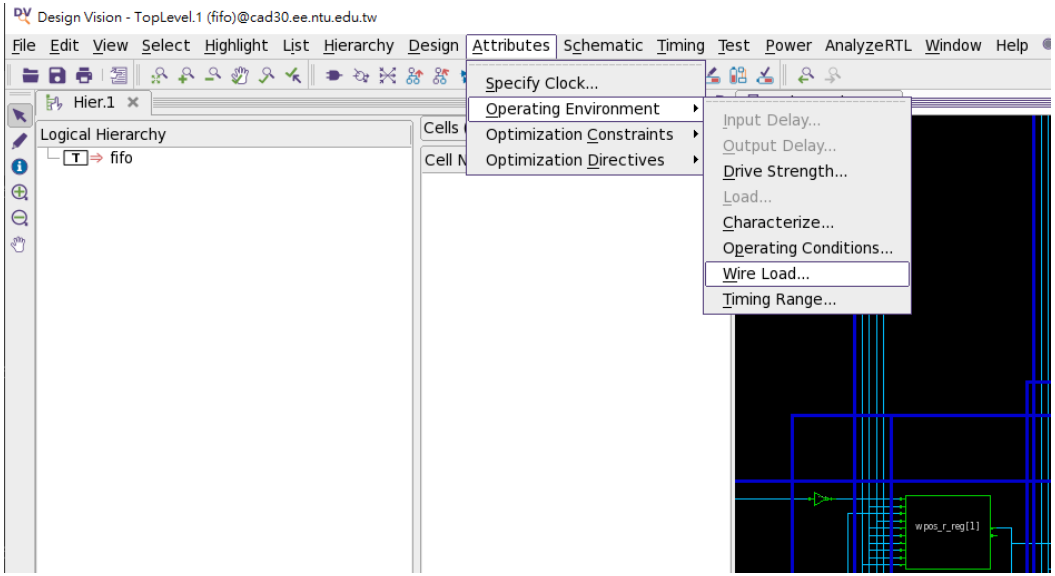


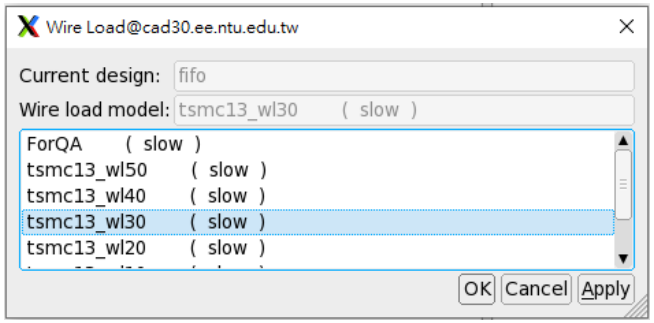




6. Setting Wire Load Model

- ***Attributes/Operating Environment/Wire Load***






```
set_wire_load_model -name tsmc13_w10 -library slow
set_wire_load_mode top
```