Lab06 : Synthesis

Using Design Compiler

Lab06 to-do actions

- Check synthesizability
- Synthesize the design (using TCL-scripts)
- Gate-Level Simulation
- Explore the design

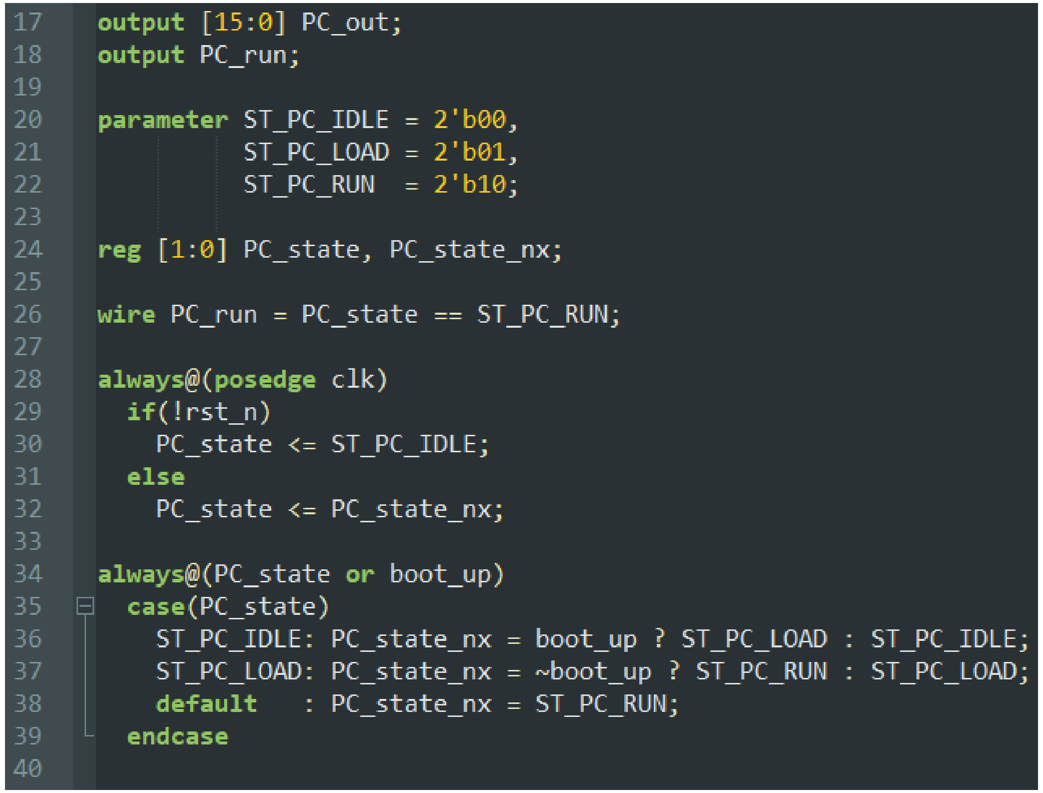
Check synthesizability

- Use Spyglass to help you check synthesizability.

Lab06 to-do actions

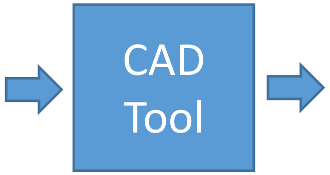
- Check synthesizability
- Synthesize the design (using TCL-scripts)
- Gate-Level Simulation
- Explore the design

Synthesis

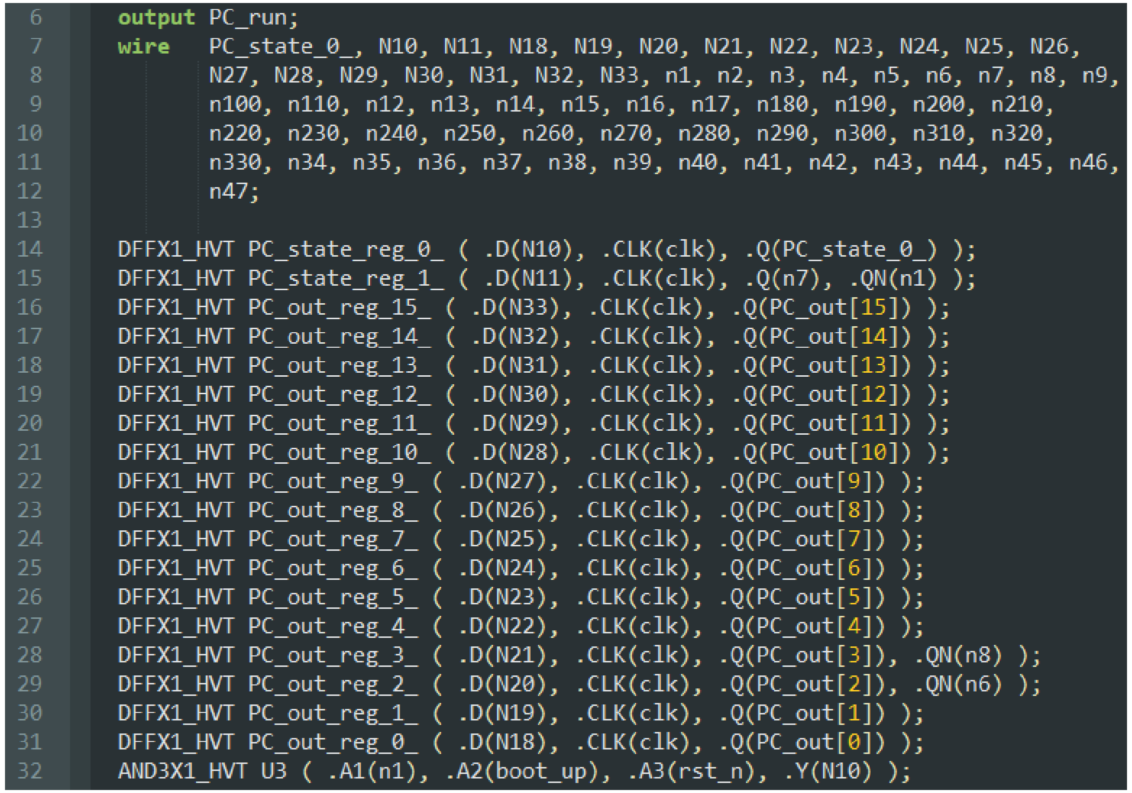


```
output [15:0] PC_out;
output PC_run;

parameter ST_PC_IDLE = 2'b00,
          ST_PC_LOAD = 2'b01,
          ST_PC_RUN  = 2'b10;

reg [1:0] PC_state, PC_state_nx;

wire PC_run = PC_state == ST_PC_RUN;

always@(posedge clk)
  if(!rst_n)
    PC_state <= ST_PC_IDLE;
  else
    PC_state <= PC_state_nx;

always@(PC_state or boot_up)
  case(PC_state)
    ST_PC_IDLE: PC_state_nx = boot_up ? ST_PC_LOAD : ST_PC_IDLE;
    ST_PC_LOAD: PC_state_nx = ~boot_up ? ST_PC_RUN : ST_PC_LOAD;
    default   : PC_state_nx = ST_PC_RUN;
  endcase

```

RTL



CAD Tool



```
output PC_run;
wire   PC_state_0_, N10, N11, N18, N19, N20, N21, N22, N23, N24, N25, N26,
       N27, N28, N29, N30, N31, N32, N33, n1, n2, n3, n4, n5, n6, n7, n8, n9,
       n100, n110, n12, n13, n14, n15, n16, n17, n180, n190, n200, n210,
       n220, n230, n240, n250, n260, n270, n280, n290, n300, n310, n320,
       n330, n34, n35, n36, n37, n38, n39, n40, n41, n42, n43, n44, n45, n46,
       n47;

DFFX1_HVT PC_state_reg_0_ ( .D(N10), .CLK(clk), .Q(PC_state_0_) );
DFFX1_HVT PC_state_reg_1_ ( .D(N11), .CLK(clk), .Q(n7), .QN(n1) );
DFFX1_HVT PC_out_reg_15_ ( .D(N33), .CLK(clk), .Q(PC_out[15]) );
DFFX1_HVT PC_out_reg_14_ ( .D(N32), .CLK(clk), .Q(PC_out[14]) );
DFFX1_HVT PC_out_reg_13_ ( .D(N31), .CLK(clk), .Q(PC_out[13]) );
DFFX1_HVT PC_out_reg_12_ ( .D(N30), .CLK(clk), .Q(PC_out[12]) );
DFFX1_HVT PC_out_reg_11_ ( .D(N29), .CLK(clk), .Q(PC_out[11]) );
DFFX1_HVT PC_out_reg_10_ ( .D(N28), .CLK(clk), .Q(PC_out[10]) );
DFFX1_HVT PC_out_reg_9_ ( .D(N27), .CLK(clk), .Q(PC_out[9]) );
DFFX1_HVT PC_out_reg_8_ ( .D(N26), .CLK(clk), .Q(PC_out[8]) );
DFFX1_HVT PC_out_reg_7_ ( .D(N25), .CLK(clk), .Q(PC_out[7]) );
DFFX1_HVT PC_out_reg_6_ ( .D(N24), .CLK(clk), .Q(PC_out[6]) );
DFFX1_HVT PC_out_reg_5_ ( .D(N23), .CLK(clk), .Q(PC_out[5]) );
DFFX1_HVT PC_out_reg_4_ ( .D(N22), .CLK(clk), .Q(PC_out[4]) );
DFFX1_HVT PC_out_reg_3_ ( .D(N21), .CLK(clk), .Q(PC_out[3]), .QN(n8) );
DFFX1_HVT PC_out_reg_2_ ( .D(N20), .CLK(clk), .Q(PC_out[2]), .QN(n6) );
DFFX1_HVT PC_out_reg_1_ ( .D(N19), .CLK(clk), .Q(PC_out[1]) );
DFFX1_HVT PC_out_reg_0_ ( .D(N18), .CLK(clk), .Q(PC_out[0]) );
AND3X1_HVT U3 ( .A1(n1), .A2(boot_up), .A3(rst_n), .Y(N10) );
```

Netlist

Synthesis

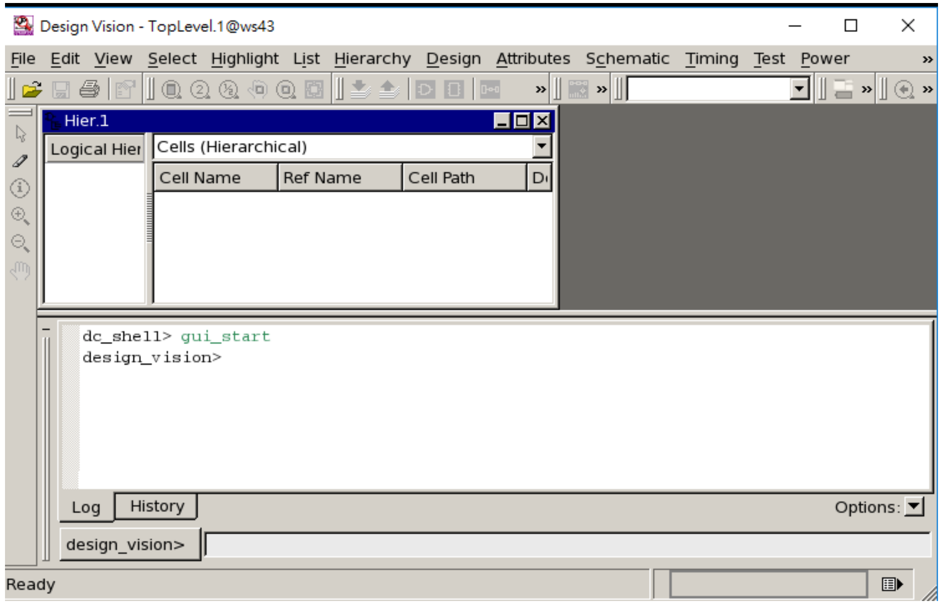
- Use *Design Compiler* to complete the synthesis process
  - Method 1 : GUI
  - Method 2 : Tcl Script (This course)

CAD Tool = *Synopsys Design Compiler*

Synthesize the design

- Use *Design Compiler* to complete the synthesis process
    - GUI

```
$ dv &
```



Design Vision - TopLevel.1@ws43
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power
Hier.1
Logical Hier
Cells (Hierarchical)
Cell Name
Ref Name
Cell Path
Log
History
Options:
Ready

```
dc_shell> gui_start
design_vision>
```

Synthesize the design

- Use *Design Compiler* to complete the synthesis process
  - GUI

```
$ dv &
```

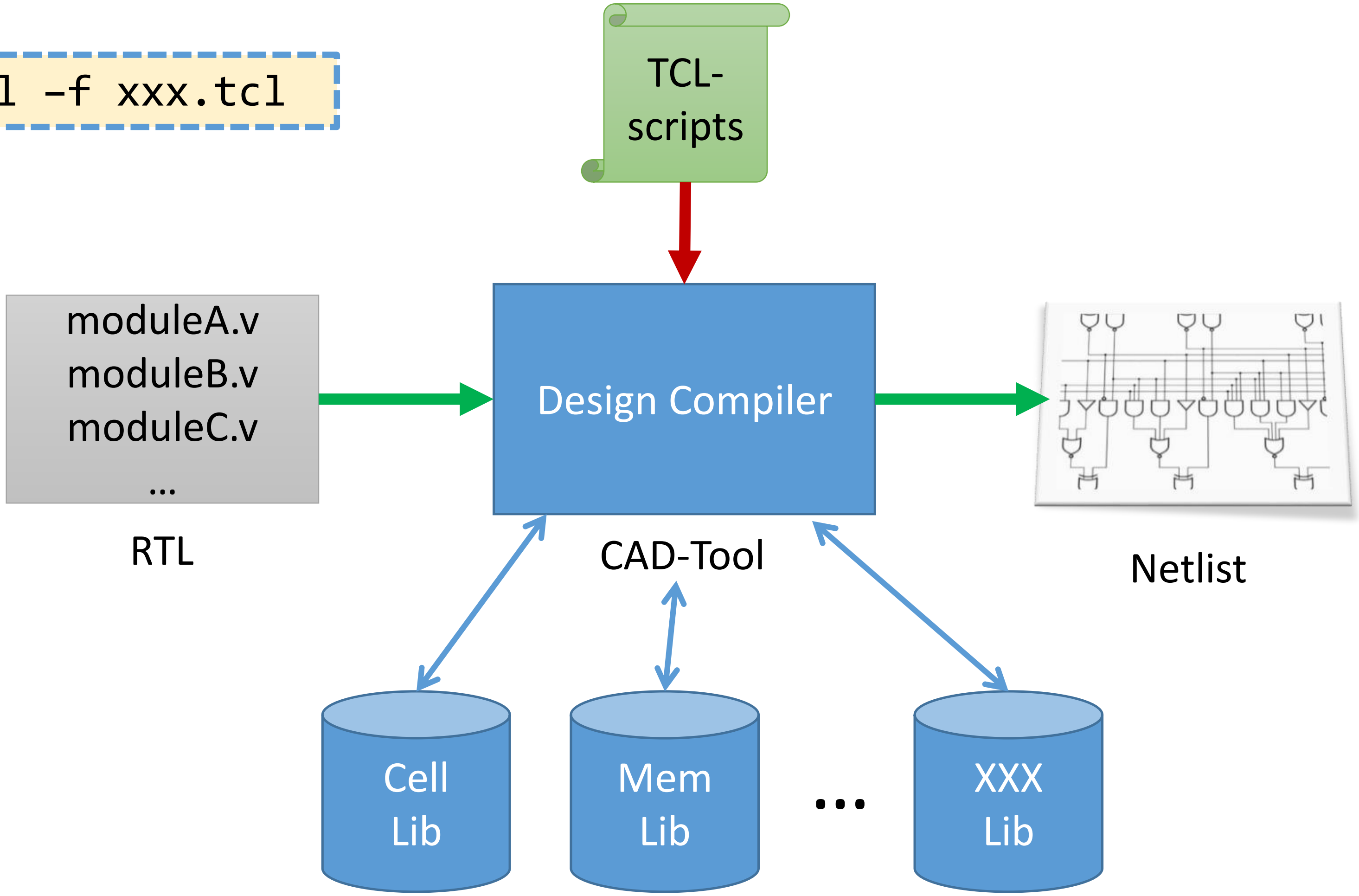
Synthesize the design

- Use *Design Compiler* to complete the synthesis process
  - Tcl Script

```
$ dc_shell -f xxx.tcl
```

TCL scripts

```
$ dc_shell –f xxx.tcl
```



TCL-scripts

moduleA.v
moduleB.v
moduleC.v
…

RTL

Design Compiler

CAD-Tool

Netlist

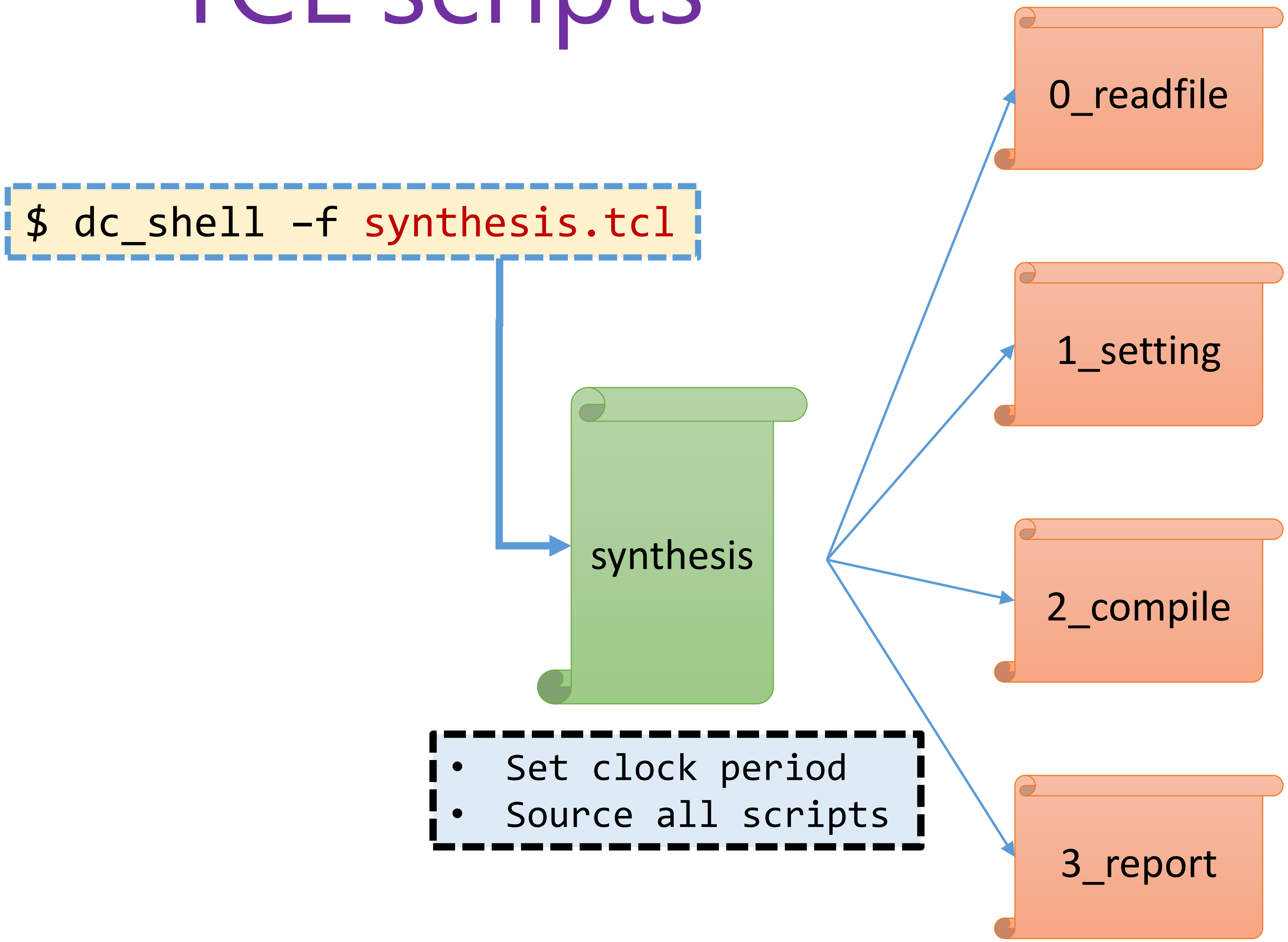
Cell Lib

Mem Lib

…

XXX Lib

TCL scripts



```
$ dc_shell –f synthesis.tcl
```

synthesis

- Set clock period
- Source all scripts

0_readfile

- Create folder to manage files
- Create library
- Read, elaborate, link design

1_setting

- Operating Condition
- Wire load model
- Clock creation, setting
- Constrain

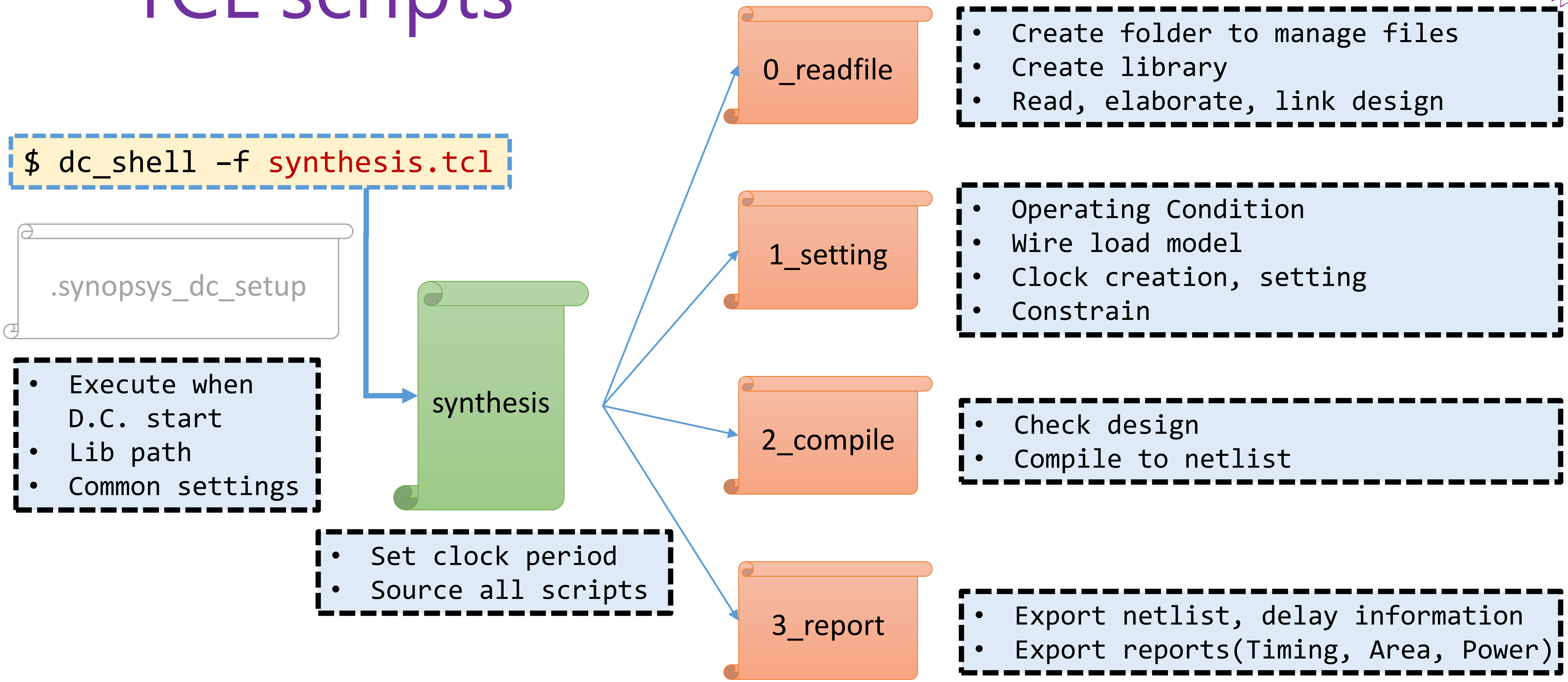
2_compile

- Check design
- Compile to netlist

3_report

- Export netlist, delay information
- Export reports(Timing, Area, Power)

TCL scripts



```
$ dc_shell –f synthesis.tcl
```

.synopsys_dc_setup

- Execute when D.C. start
- Lib path
- Common settings

synthesis

- Set clock period
- Source all scripts

0_readfile

- Create folder to manage files
- Create library
- Read, elaborate, link design

1_setting

- Operating Condition
- Wire load model
- Clock creation, setting
- Constrain

2_compile

- Check design
- Compile to netlist

3_report

- Export netlist, delay information
- Export reports(Timing, Area, Power)

TCL scripts

- Scripts to be filled :
  - .synopsys_dc_setup
  - synthesis.tcl
  - 0_readfile.tcl
  - 1_setting.tcl
  - 2_compile.tcl
  - 3_report.tcl (already done)
- Please follow the tutorials provided in iLMS

TCL scripts

- If you’re feeling confused about some commands
  - Try to find answers in dc_shell by yourself

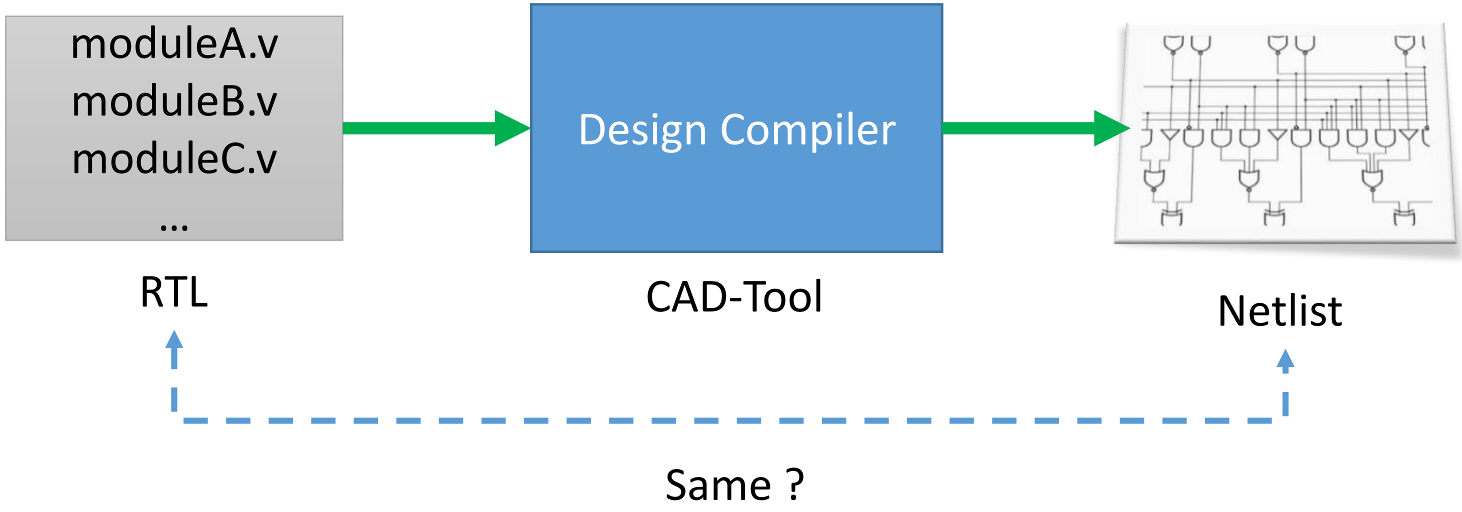
```
$ dc_shell # enter the Design-Compiler shell program

Warning:  Site Information is not available ... Have you run install_site?
……
…

dc_shell>
dc_shell>
dc_shell> man create_clock
<D.C. will give you the explanation of create_clock here>
dc_shell> exit # say goodbye to design compiler
```

Lab06 to-do actions

- Check synthesizability
- Synthesize the design (using TCL-scripts)
- Gate-Level Simulation
- Explore the design

Gate-Level Simulation

- Ensure the functionality of the netlist generated by D.C.



moduleA.v
moduleB.v
moduleC.v
…

RTL

Design Compiler

CAD-Tool

Netlist

Same ?

Gate-Level Simulation

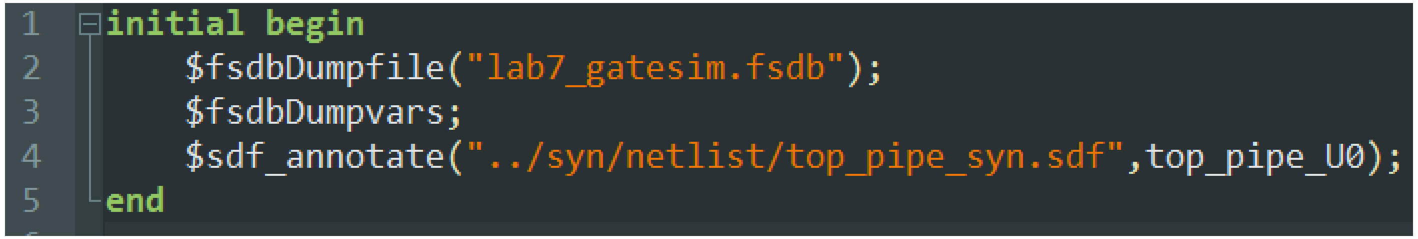
- Files need in pre-simulation
  - RTL modules
  - Testbench
  - Testing data

- Files need in gate-simulation
  - Netlist (xxx_syn.v)
  - Cell model (saed32nm.v)
  - Delay information (xxx_syn.sdf)
  - Testbench
  - Testing data

Gate-Level Simulation

- Include delay information (standard delay file) inside testbench

```
initial begin
        $sdf_annotate(“path/to/your/xxx.sdf”, TOP_MODULE_INSTANCE_NAME)
end
```



```
initial begin
    $fsdbDumpfile("lab7_gatesim.fsdb");
    $fsdbDumpvars;
    $sdf_annotate("../syn/netlist/top_pipe_syn.sdf",top_pipe_U0);
end
```

Lab06 to-do actions

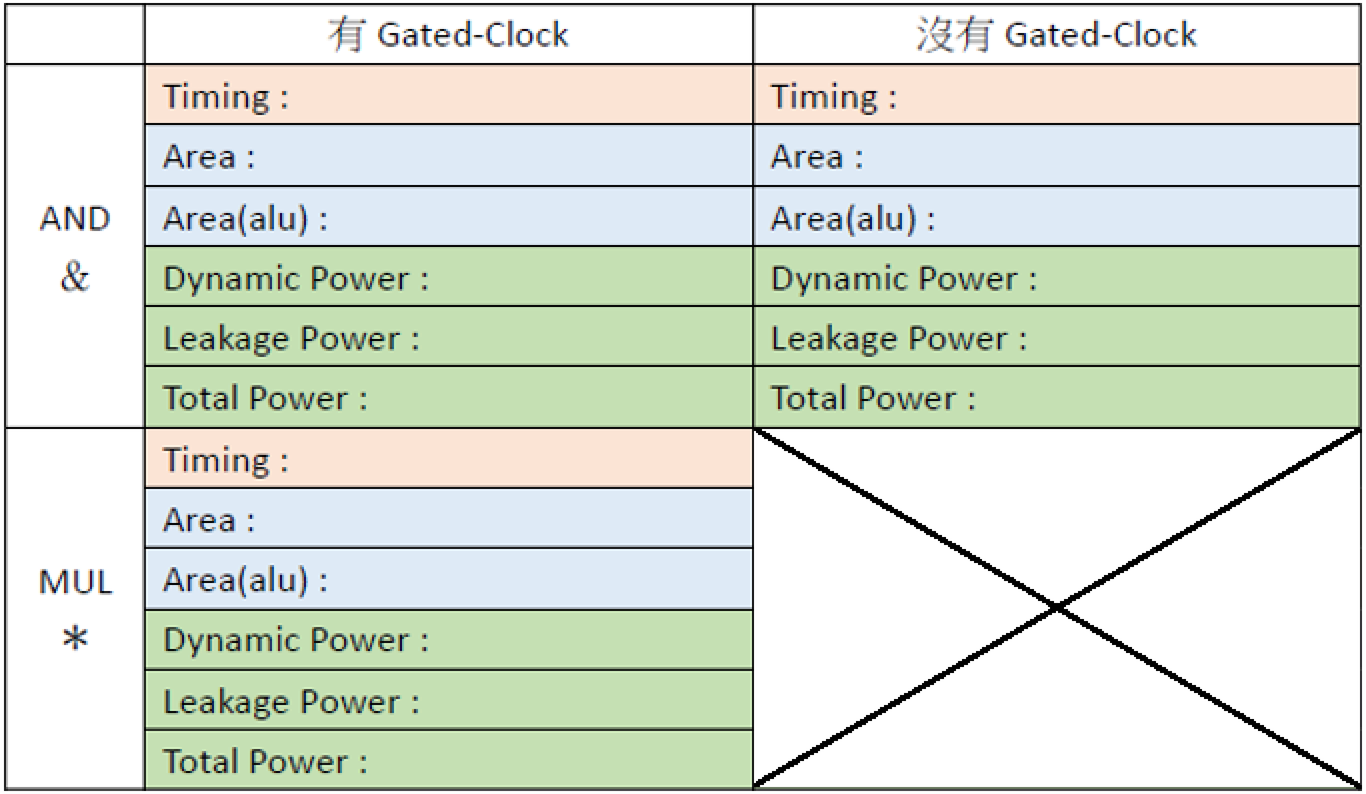
- Check synthesizability
- Synthesize the design (using TCL-scripts)
- Gate-Level Simulation
- Explore the design

Explore the design

- After synthesis, you can see the summary reports under */syn/report*
  - Area
    - report_area_xxx.out
  - Timing
    - report_time_xxx.out
    - report_hold_xxx.out
    - report_setup_xxx.out
  - Power
    - report_power_xxx.out

Explore the design

- Try different configuration/implementation and compare the results



	有 Gated-Clock	沒有 Gated-Clock
AND &	Timing :	Timing :
	Area :	Area :
	Area(alu) :	Area(alu) :
	Dynamic Power :	Dynamic Power :
	Leakage Power :	Leakage Power :
	Total Power :	Total Power :
MUL *	Timing :	
	Area :	
	Area(alu) :	
	Dynamic Power :	
	Leakage Power :	
	Total Power :	

Explore the design

- Gated clock *(/syn/2_compile.tcl)*

  enable gated clock :

  ```
  compile_ultra -gate_clock -exact_map -……
  ```

  disable gated clock:

  ```
  compile_ultra -exact_map -……
  ```

- AND / MUL *(/source/alu.v , and don’t forget to modify the testbench as well)*

  ```
  alu_result_tmp = src1 & src2;
  ```

  ```
  alu_result_tmp = src1 * src2;
  ```

Good Luck

- Synthesis roughly takes 13~15 minutes to complete.