注意事項

參賽者請於 10 點半前完成下列步驟進行初賽登錄，主辦單位將依完成此步驟之隊伍數決定各組最後得獎名額，請務必完成登錄動作，以免影響你的權益。

一、請將你的隊伍參賽資料 E-mail 至 boyu@vlsilab.ee.ncku.edu.tw，信件內容格式如下:

參賽組別：(A:大學組全客戶設計；B:研究所組全客戶設計；C:標準單元設計；D:類比單元設計；E:大學組可程式邏輯設計)

參賽編號：(例：951001)

參賽姓名：張三、李四

二、信件標題請標示為「**IC 設計競賽初賽資料登錄**」

2006 University/College IC Design Contest Preliminary

Cell-Based IC Category

Triangle Rendering Engine

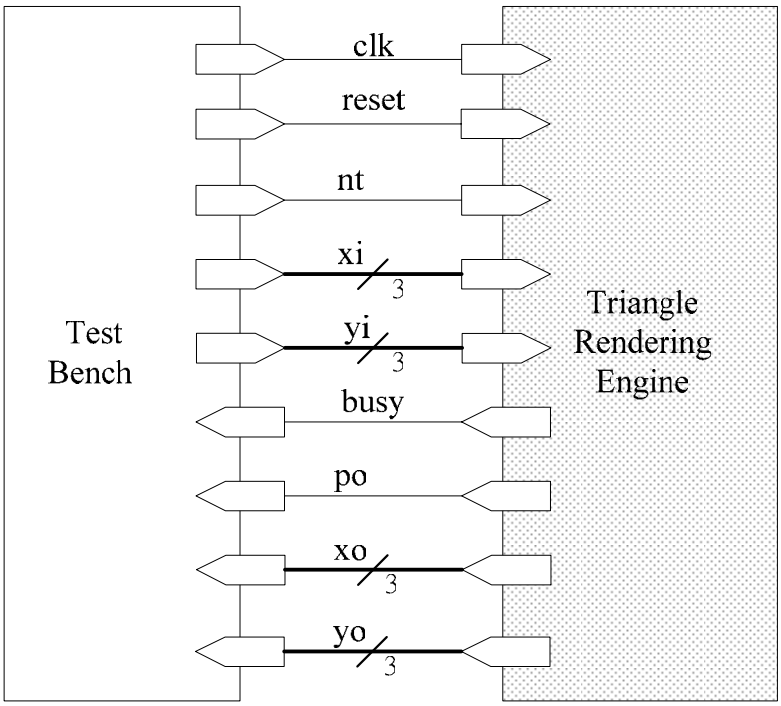
1. Problem Description

Please design a triangle rendering engine. The triangle rendering engine performs the rendering computations and then outputs a sequence of valid outputs *(xo, yo)* after the testbench provides 3 consecutive pairs of coordinate inputs for a triangle, i.e. *(x1,y1), (x2,y2)* and *(x3,y3)*. The detailed design specifications, including block overview, I/O interface, functional behavior and timing requirement, will be given in the next sections. Each team should finish the design by utilizing the design environment provided by CIC within the given time.

After the end of the contest, CIC will rate the successful designs according to the scoring rule defined in section 3. All the related materials as illustrated in Appendix D should be transferred to CIC. And the procedures for transferring are described in Appendix E. Note that the time for this contest is from 8:30 to 20:30. That means you have to transfer all the related materials to CIC before 20:30.

2. Design Specifications

2.1 Block Overview



2.2 I/O Interface

Signal Name	Direction	Width	Description
reset	input	1	Active-high asynchronous reset signal. When the signal *reset* is asserted, the design is asynchronous reset.
clk	input	1	Clock source. The design is a synchronous design triggered at the positive edge of *clk*.
nt	input	1	New triangle indication. When the signal *nt* is active high, it means there are 3 new consecutive pairs of coordinate inputs for a triangle. Note that the signal *nt* is active high only if the signal *busy* is low.
xi	input	3	The x-coordinate input of the triangle.
yi	input	3	The y-coordinate input of the triangle.
busy	output	1	Busy signal. When the signal *busy* is high, it indicates the triangle rendering engine is performing the operations, and thus can prevent the new coordinate inputs of the triangle, i.e. the signal *nt* will not be active.
po	output	1	Valid output indication. When the signal *po* is high, there are a sequence of coordinate outputs for a triangle valid on the output ports (*xo* and *yo*).
xo	output	3	The output result of x-coordinate.
yo	output	3	The output result of y-coordinate.

2.3 Functional Description

The design is asynchronously reset by asserting the signal *reset*. After releasing the signal *reset*, the design begins to function and is clocked at the edge of the signal *clk*. The signal *nt* is active only if the signal *busy* is low. When the signal *nt* is active, the testbench starts to provide 3 consecutive pairs of coordinate inputs for a triangle in the following order, i.e. *(x1,y1), (x2,y2)* and *(x3,y3)*. The triangle rendering engine should assert the signal *busy* before the last coordinate input, *(x3,y3)*, to prevent the new coordinate inputs for next triangle. After a period of the time, the signal *po* is active to indicate there is a valid coordinate output *(xo, yo)* on the output ports. The valid coordinate outputs *(xo, yo)* will be verified automatically by the testbench provided by the sponsor only if the signal *po* is active. The valid coordinate outputs *(xo, yo)* should appear in output ports in the same order specified in example 1.

The three input *(x1,y1), (x2,y2)* and *(x3,y3)* will not in the same line and will meet the following relation.

$$x1 = x3$$

$$y1 < y2 < y3$$

This means that one side of the triangle will be vertical.

Figure 1 shows the example of a triangle. Here gives an example that the coordinate inputs *(x1,y1), (x2,y2)* and *(x3,y3)* of a triangle are *(1,1), (6,3) and (1,6),* respectively. The triangle rendering engine should output the following 17 valid coordinate outputs in the following order:*(1,1), (1,2), (2,2), (3,2), (1,3) ,(2,3), (3,3), (4,3) ,(5,3) ,(6,3), (1,4), (2,4), (3,4) , (4,4) , (1,5) , (2,5) , (1,6).*

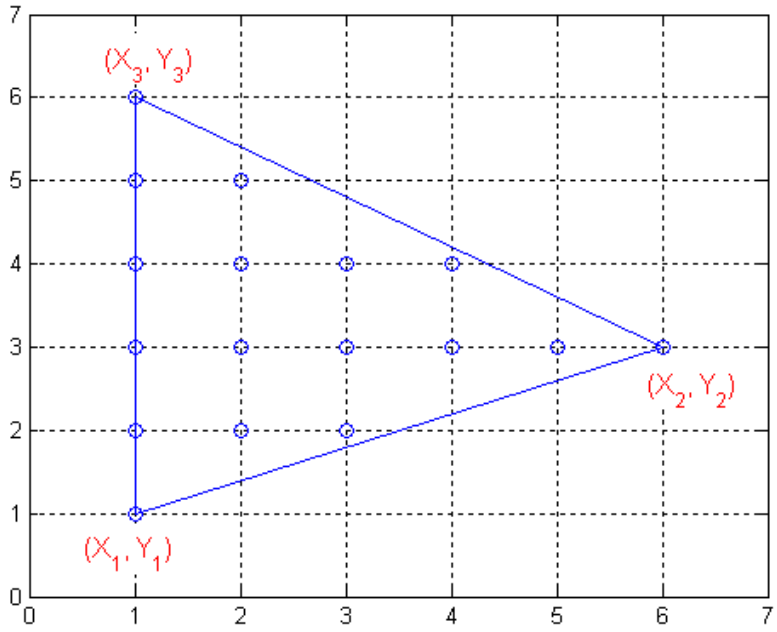


Figure 1: Example of a triangle

The data width of all the coordinate inputs and outputs is 3-bits.

[Hint]

Given two points (x1, y1) and (x2,y2). The equation of a line passes the (x1,y1) and (x2,y2) is

$$\frac{x - x_1}{y - y_1} - \frac{x_2 - x_1}{y_2 - y_1} = 0 \qquad [1]$$

Any points on the line will satisfy the equation [1].

Any points in the right hand side of the line will satisfy the following equation [2].

$$\frac{x - x_1}{y - y_1} - \frac{x_2 - x_1}{y_2 - y_1} > 0 \qquad [2]$$

Any points in the left hand side of the line will satisfy the following equation [3].

$$\frac{x - x_1}{y - y_1} - \frac{x_2 - x_1}{y_2 - y_1} < 0 \qquad [3]$$

Designer can use the above equation to identify if the points is enclosed by the triangle or not.

2.4 Timing Diagrams

The timing diagram of the design is depicted in Figure 2.

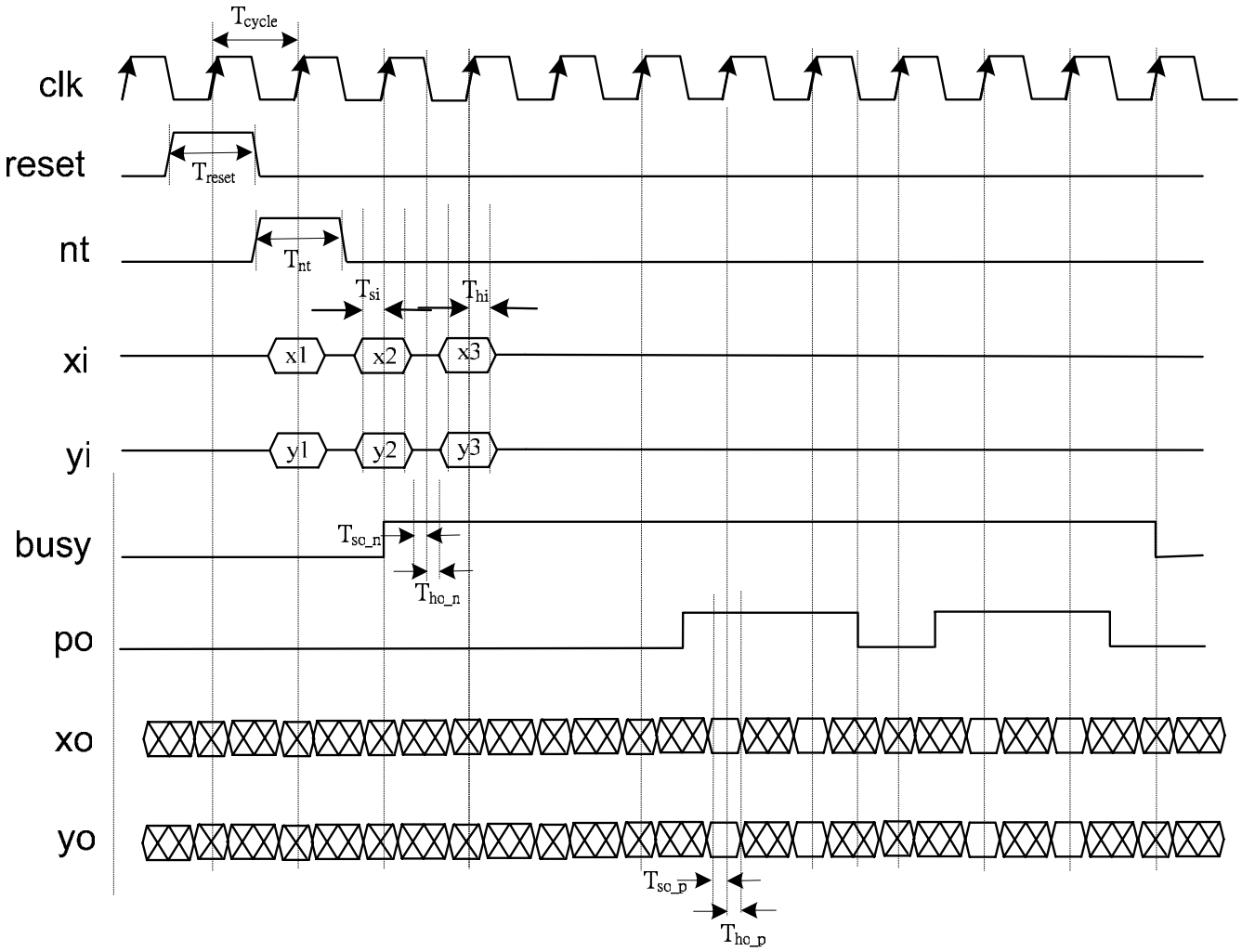


Figure 2: Timing diagram

Symbol	Description	Value
T_{cycle}	Clock (clk) period with duty cycle 50%	User defined
T_{reset}	Reset pulse width, active between negative edges of clk.	= T_{cycle}
T_{nt}	New triangle pulse width, active between negative edges of clk.	= T_{cycle}
T_{si}	Time period from valid signal to positive edge of clk.	=0.2 T_{cycle}.
T_{hi}	Time period from positive edge of clk to invalid signal.	=0.2 T_{cycle}.
T_{so_p}	Setup time from valid output to positive edge of clk.	>0.5ns
T_{ho_p}	Hold time from positive edge of clk to invalid output.	>0.5ns
T_{so_n}	Setup time from valid output to negative edge of clk.	>0.5ns
T_{ho_n}	Hold time from negative edge of clk to invalid output.	>0.5ns

3. Scoring

The referees will check the completeness as well as the correctness of your design data. If all materials listed in RTL category and Gate-Level Category of Table I are uploaded to CIC, the design is complete. The correctness means the result of pre-layout gate-level simulation under the user-defined clock rate is the same as the golden result provided by CIC and without any setup/hold time violations.

Once the completeness and correctness are confirmed by the referees, the score is calculated. The scoring rule is quite simple. The earlier you complete your design, the higher score you'll get. And the higher the score, the higher the rank will be. Note that the time you complete your design is determined by the time the design of the latest version is uploaded to CIC's ftp sites.

If necessary, those designs with only files in RTL category uploaded are also ranked. In this case, the correctness means RTL simulation result is the same as the golden result provided by CIC.

Appendix

There are two triangles provided in the testbench. The Appendix A shows each diagram and

corresponding coordinate outputs. The related files and notes provided by sponsor for **Verilog** category and **VHDL** category are listed in the **Appendix B** and **Appendix C**, respectively. The materials that each team should hand in are listed in the Appendix D, while the transferring steps are described in the Appendix E.

Appendix A (Triangles in the Testbench)

First triangle:

Coordinate inputs:

(1,0), (7,2) and (1,7)

Coordinate outputs in the following order:

(1,0), (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2), (5,2), (6,2), (7,2), (1,3), (2,3), (3,3), (4,3) ,(5,3), (1,4), (2,4), (3,4), (4,4), (1,5), (2,5), (3,5), (1,6), (2,6), (1,7)

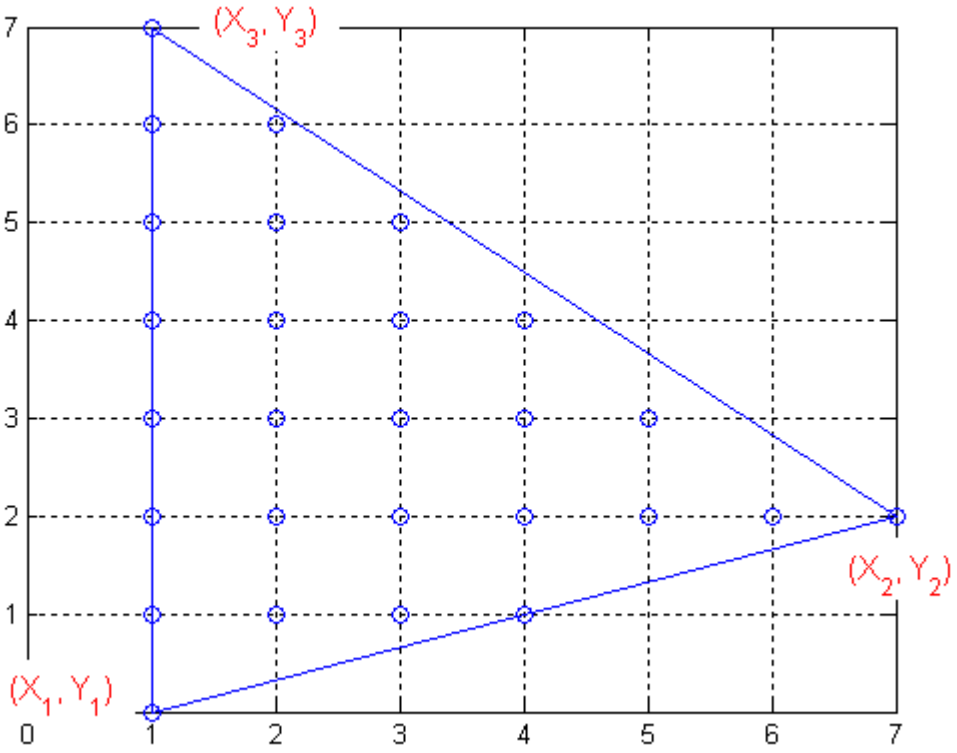


Figure 3: First triangle in testbench

Second triangle:

Coordinate inputs:

(6,1), (0,3) and (6,6)

Coordinate outputs in the following order:

(6,1), (3,2), (4,2), (5,2), (6,2), (0,3), (1,3), (2,3), (3,3), (4,3), (5,3), (6,3), (2,4), (3,4), (4,4), (5,4), (6,4), (4,5), (5,5), (6,5), (6,6)

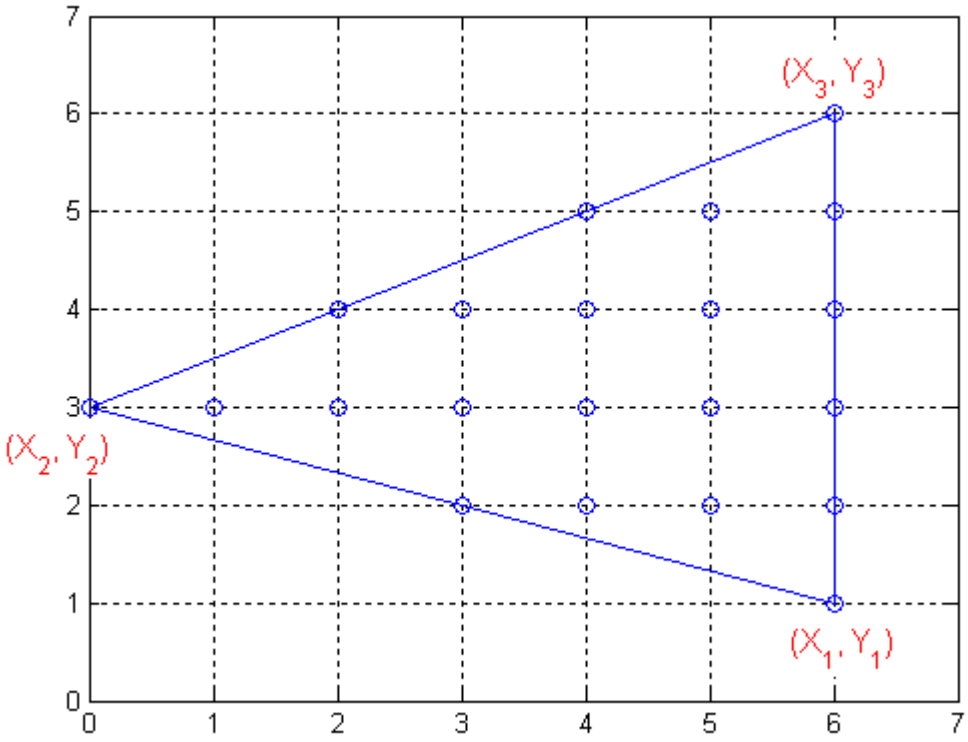


Figure 4: Second triangle in testbench

Appendix B (For Verilog category)

1. The following files are provided by sponsor:

   00.README: Readme file.

   testfixture.v: Test bench includes the definition of clock period and the file names of test patterns.

   triangle.v : The input and output ports are declared in this file.

triangle.sdc : The operating conditions and boundary conditions file for the Synopsys Design Compiler setup (without the statements of design constraint).

input.dat: The input coordinates (*xi*, *yi*) of the triangle are listed in this file.

expect.dat: The expect results *(xo,yo)* are recorded in this file.

report.000: This file is used to describe the materials that should be handed in by each team. The design and related file names, tool names, related specifications and others are described in this file.

2. Please use the module ***triangle*** in the file, ***triangle.v***, to design a Triangle Rendering Engine. The names and port types of all the IOs are declared as follows:

```
module triangle (clk, reset, nt, xi, yi, busy, po, xo, yo);
input        clk, reset, nt;
input  [2:0] xi, yi;
output       busy, po;
output [2:0] xo, yo;
endmodule
```

3. Please use the test bench (testfixture.v) provided by sponsor to verify your design. The referee will verify the correctness by using this test bench. Besides, referee may use other test patterns to verify the design of each team.
   A. Test bench is included in this testfixture.v. The signals including *clk, reset, nt, xi,* and *yi* are generated by testbench for verification. The parameter, CYCLE, defined in the testfixture.v can be modified for different CLK period.
   B. After all the test patterns are verified, testfixture prompts **PASS** as all the results are correct. Otherwise, the cycle count, error value and expected value are displayed.
   C. The .synopsys_dc.setup contains the environment setup of Synopsys Design Compiler. Modify the "<Your_Design_Kit_Path>" keyword in this file to the path of the installed directory of CIC 0.18um TSMC/Artisan cell-based design kit. Then Synopsys Design Compiler will set the environment according to it when starting up. Any modifications to this file except mentioned above are illegal. Your design may be treated as not correct due to these illegal conditions. Attempts to change the important settings in this file are also illegal. For example, try to override the link library setting with some related commands. The important settings that are not allowed to change are listed below.

```
Link library:        slow.db fast.db dw_foundation.sldb
Target library:      slow.db fast.db
```

   D. For the timing constraints, please use "triangle.sdc" as the template. Modify the parameters, "cycle", "t_in", and "t_out" to the target clock period and reasonable I/O delays. And then

load the constraints. There are two ways to load the constraints into Synopsys Design Compiler. One is GUI method. This is done by clicking “File/Execute Script” item in Design Vision menu banner. The other is command method. This is done by executing the command “source CS.sdc” in dc shell. Any modifications except mentioned above are illegal. Your design may be treated as not correct due to these illegal conditions. Attempts to change the important settings in this file are also illegal. For example, try to override the output load setting with some related commands. The important settings that are not allowed to change are listed below.

```
Clock uncertainty:            0.1ns
Clock latency:                0.5ns
Working environment:          slow fast
Wire load model:              tsmc18_wl10
All input drive (except clk): 1ns/pf
All output load:              1pf
```

E. In the gate-level simulation, the simulation model (tsmc18.v) and the sdf file which is created by Synopsys Design Compiler are needed. Even if the gate level netlist is not created by Synopsys Design Compiler, this gate level netlist should be loaded by Synopsys Design Compiler with the setup environment, triangle.sdc, and then the sdf file (version 2.1) should be created for the simulator.

Appendix C (For VHDL category)

1. The following files are provided by sponsor:

   00.README: Readme file.

   testfixture.v: Test bench includes the definition of clock period and the file names of test patterns.

   triangle.vhd : The input and output ports are declared in this file.

   triangle.sdc : The operating conditions and boundary conditions file for the Synopsys Design

Compiler setup (without the statements of design constraint).

input.dat: The input coordinates (*xi*, *yi*) of the triangle are listed in this file.

expect.dat: The expect results *(xo,yo)* are recorded in this file.

report.000: This file is used to describe the materials that should be handed in by each team. The design and related file names, tool names, related specifications and others are described in this file.

2. Please use the ENTITY ***triangle*** in the file, ***triangle.vhd***, to design a Triangle Rendering Engine. The file name of the RTL code should be "triangle.vhd" and the names of the top module I/O ports should be the same as those defined in the file "triangle.vhd" CIC provides. Otherwise, your design will not be able to be recognized by the test bench. **Please don't use configuration design unit in the triangle.vhd in order to run VHDL and Verilog co-simulation.**
3. Please use the test bench (testfixture.v) provided by sponsor to verify your design. The referee will verify the correctness by using this test bench. Besides, referee may use other test patterns to verify the design of each team. **For VHDL simulation, NC-Sim or ModelSim is recommended since we have to run VHDL and Verilog co-simulation now.**
   A. Test bench is included in this testfixture.v. The signals including *clk, reset, nt, xi,* and *yi* are generated by testbench for verification. The parameter, CYCLE, defined in the testfixture.v can be modified for different CLK period.
   B. After all the test patterns are verified, testfixture prompts **PASS** as all the results are correct. Otherwise, the cycle count, error value and expected value are displayed.
   C. The .synopsys_dc.setup contains the environment setup of Synopsys Design Compiler. Modify the "<Your_Design_Kit_Path>" keyword in this file to the path of the installed directory of CIC 0.18um TSMC/Artisan cell-based design kit. Then Synopsys Design Compiler will set the environment according to it when starting up. Any modifications to this file except mentioned above are illegal. Your design may be treated as not correct due to these illegal conditions. Attempts to change the important settings in this file are also illegal. For example, try to override the link library setting with some related commands. The important settings that are not allowed to change are listed below.

```
Link library:       slow.db fast.db dw_foundation.sldb
Target library:     slow.db fast.db
```

   D. For the timing constraints, please use "triangle.sdc" as the template. Modify the parameters, "cycle", "t_in", and "t_out" to the target clock period and reasonable I/O delays. And then load the constraints. There are two ways to load the constraints into Synopsys Design Compiler. One is GUI method. This is done by clicking "File/Execute Script" item in Design Vision menu banner. The other is command method. This is done by executing the command "source CS.sdc" in dc shell. Any modifications except mentioned above are illegal. Your design may be treated as not correct due to these illegal conditions. Attempts to change the important settings in this file are also illegal. For example, try to override the

output load setting with some related commands. The important settings that are not allowed to change are listed below.

```
Clock uncertainty:            0.1ns
Clock latency:                0.5ns
Working environment:          slow fast
Wire load model:              tsmc18_wl10
All input drive (except clk): 1ns/pf
All output load:              1pf
```

E. In the gate-level simulation, the simulation model (tsmc18.v) and the sdf file which is created by Synopsys Design Compiler are needed. Even if the gate level netlist is not created by Synopsys Design Compiler, this gate level netlist should be loaded by Synopsys Design Compiler with the setup environment, triangle.sdc, and then the sdf file (version 2.1) should be created for the simulator.

Appendix D

The materials that each team should hand in are listed below.

Table I - Necessary Deliverables

RTL category		
Design Stage	*File*	*Description*

N/A	`report.xxx`	design report
RTL Simulation	`*.v or *.vhd`	Verilog (or VHDL) synthesizable RTL code
Gate-Level category		
Design Stage	*File*	*Description*
Pre-layout Gate-level Simulation	`*_syn.vg`	Verilog gate-level netlist generated by Synopsys Design Compiler
	`*_syn.sdf`	SDF timing information generated by Synopsys Design Compiler
	`*_syn.db`	design database generated by Synopsys Design Compiler

Several rules and tips of generating these files are stated below. Please read them carefully before you hand in the materials.

- *For RTL category*

Once you finish your design, you have to fill in the fields as many as possible in "`report.xxx`" of which the extension "`xxx`" indicate the revision number. Referees will check the correctness of your design according to the materials you hand in. Refer to "`report.tem`" if you have any difficulties filling in the fields.

Note that the RTL code should be synthesizable. Otherwise, the design will be treated as not correct.

- *For Gate-Level category*

The gate-level netlist should not contain any behavioral statements, for example, assign statement. Otherwise, your design will be treated as not correct. **About the SDF timing information, please use version 2.1 for pre-layout gate level simulation.**

Appendix E

All files listed in Table I have to be handed in. Before the files are transferred to CIC, all files should be archived and compressed. The following is a step-by-step illustration.

1. Create a new directory.
2. Copy all the files to be handed in except the "`report.xxx`" file into the created directory. The file

extension “xxx” means the revision number. For example, for the first revision, the file name is “report.000”. You may have improvements on your design later and want to re-transmit the design data. Now, the report file name should be “report.001” to let referees always get the most updated design data.

3. Execute the following UNIX command to create an archive. After this operation, you will get “your_username_xxx.tar”. The purpose of using “xxx” postfix is the same as that of using “xxx” extension for report file.

   > tar cvf your_username_xxx.tar *

4. Execute the following UNIX command to compress the archive. After this operation, you will get “your_username_xxx.tar.Z”.

   > compress your_username_xxx.tar

5. Check if all necessary fields of the report form (report.xxx) are filled in.
6. Transfer the compressed file “your_username_xxx.tar.Z” and the report file “report.xxx” via ftp with binary mode to the following ftp sites. The username and password will be given 4 days before of the contest via email. Any problems, please contact CIC.

   FTP site1 (NTU)：iccftp.ee.ntu.edu.tw (140.112.20.92)
   FTP site2 (CIC)：iccftp.cic.org.tw (140.126.24.18)
   FTP site3 (NCKU)：iccftp.ee.ncku.edu.tw (140.116.156.55)

7. Repeat the above steps if you have updated design. Remember to change the extension of the report file name as well as the postfix of the compressed file name to indicate the revision number. Also remember that the content of the report should be updated.