ModelSim and Testbench

2016/03/18 YZU CSE

Outline

- **Introduction**
- **Hello ModelSim**
- **Testbench Tutorial**
  - Lab 1: 1-bit Full Adder
  - Lab 2: Counter
- **Exercise**
  - Lab 3: Seg7viewer

Introduction

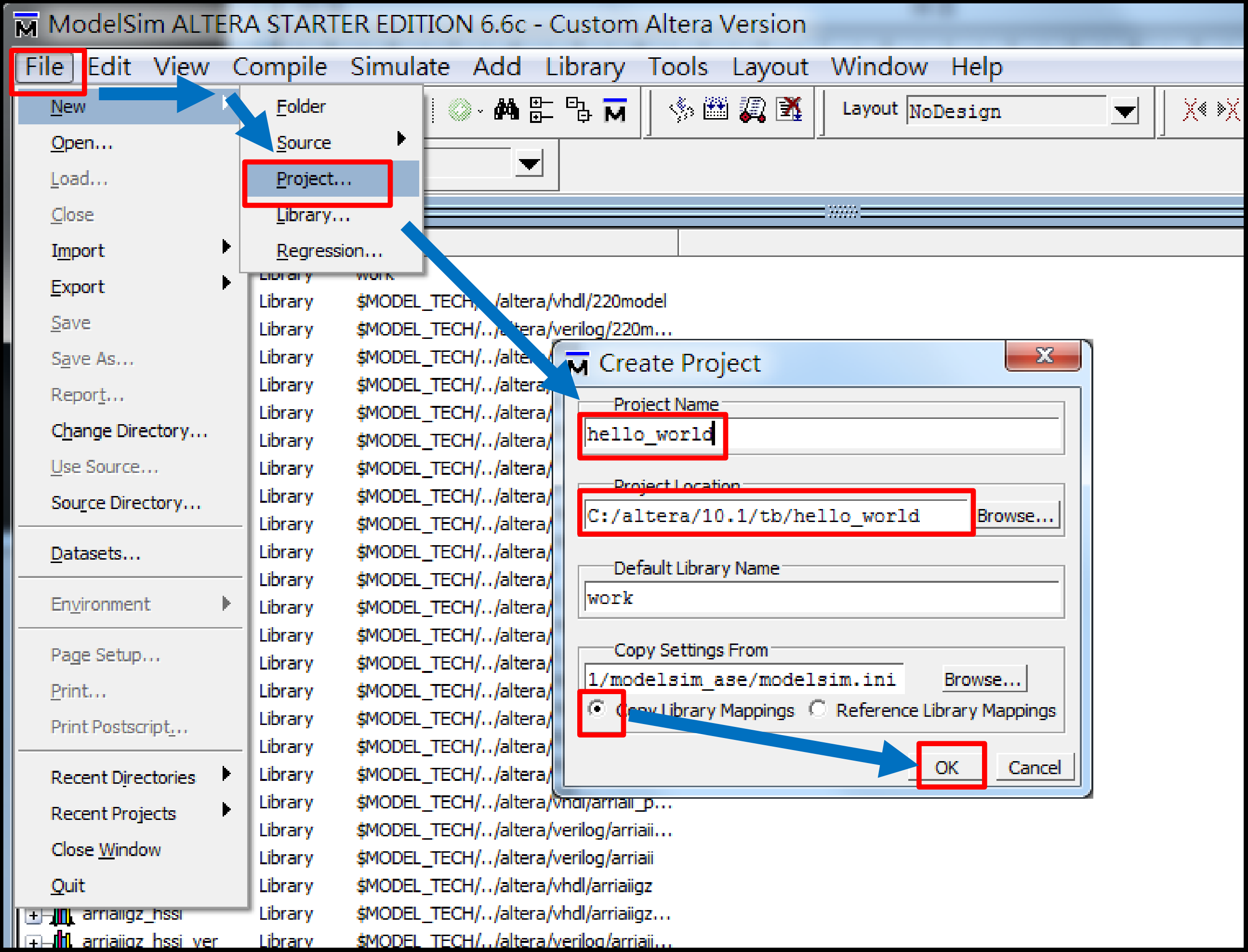
- A testbench is a virtual environment used to verify the correctness or soundness of a design
- Typically , we spend more than 50% of time to verify/test/debug the design
- When the system becomes larger, it’s more and more complex to verify
- Device under test (DUT)
  - Keep tests small and fast
  - Make tests simple to run

Outline

- **Introduction**
- **Hello ModelSim**
- **Testbench Tutorial**
  - Lab 1: 1-bit Full Adder
  - Lab 2: Counter
- **Exercise**
  - Lab 3: Seg7viewer

Create Project

- **File -> New -> Project**



Create Project

Project Name: hello_world

Project Location: C:/altera/10.1/tb/hello_world

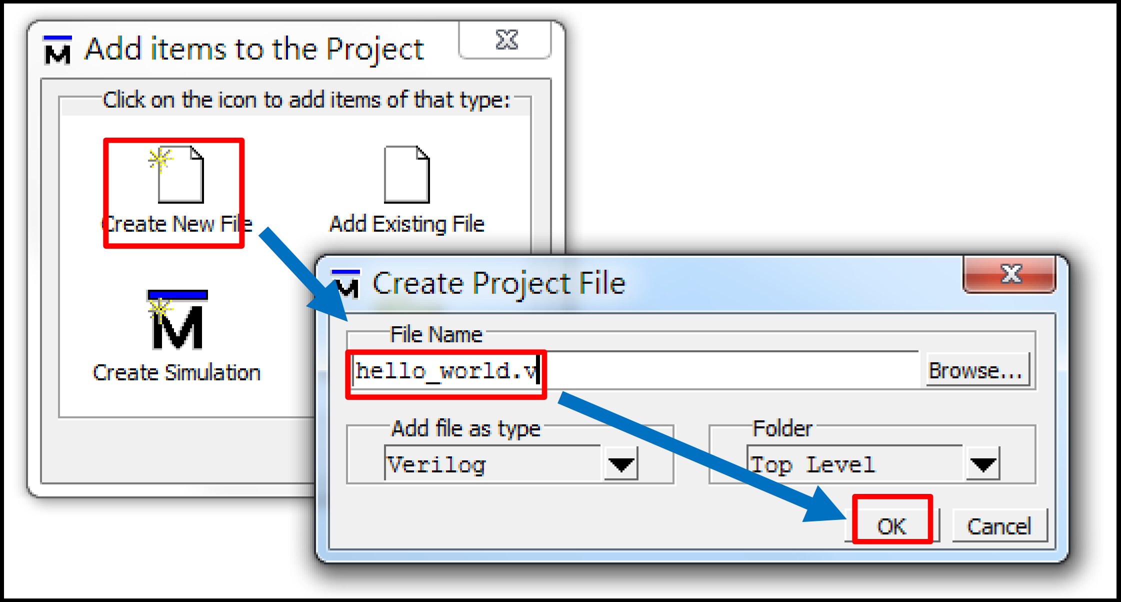
Default Library Name: work

Copy Settings From: 1/modelsim_ase/modelsim.ini

[✓ Copy Library Mappings] Reference Library Mappings

Create New File

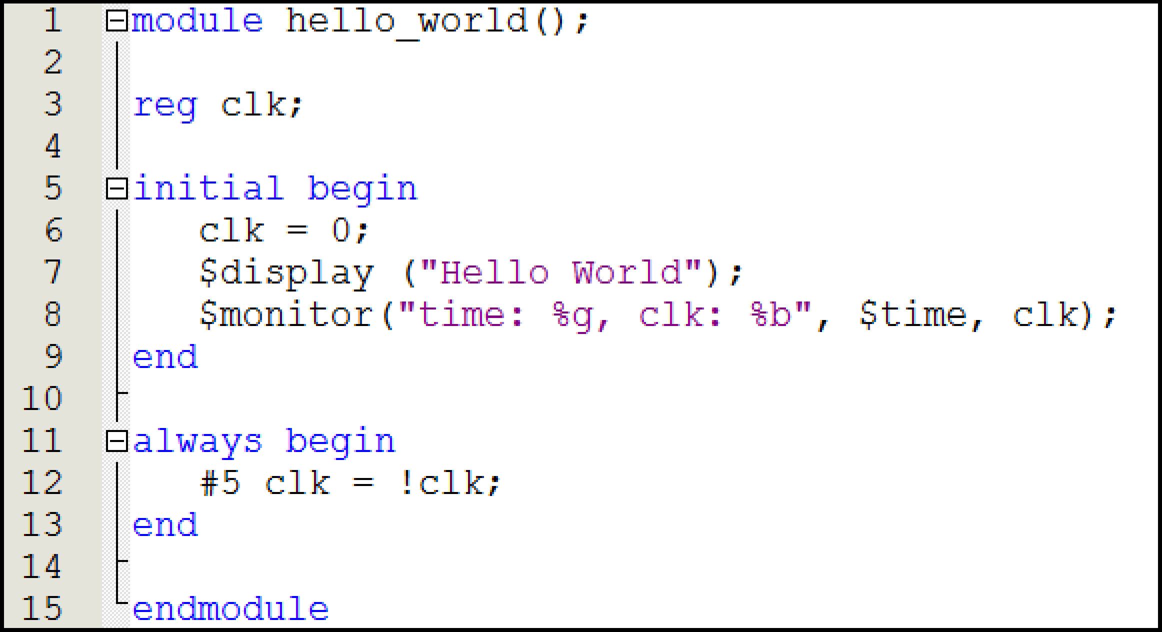
- **Hello_world.v**



File Name: hello_world.v
Add file as type: Verilog
Folder: Top Level

Hello ModelSim

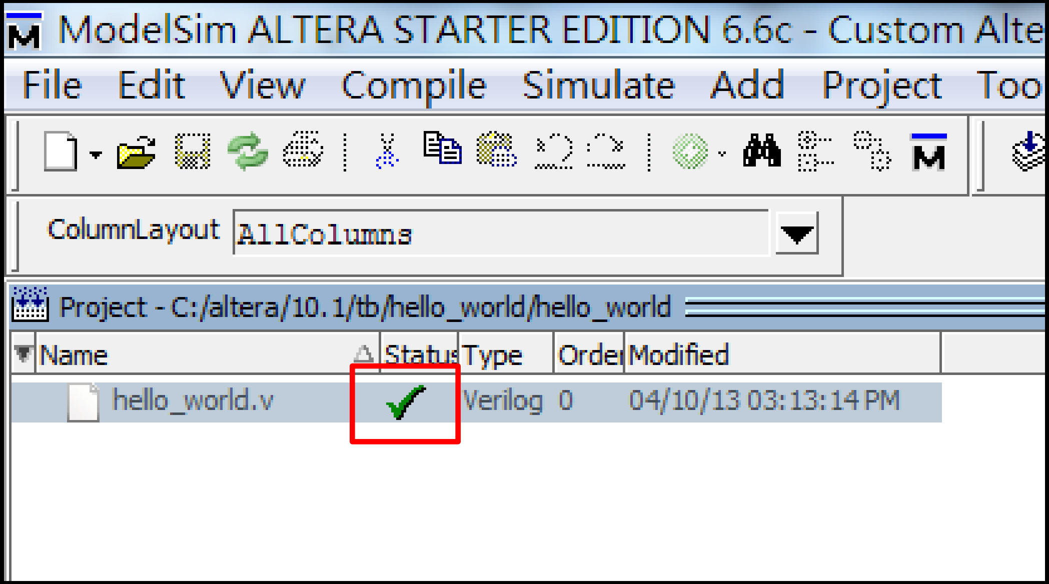
- **The first program of ModelSim**



```verilog
module hello_world();

reg clk;

initial begin
    clk = 0;
    $display ("Hello World");
    $monitor("time: %g, clk: %b", $time, clk);
end

always begin
    #5 clk = !clk;
end

endmodule
```

Compile Veriolog

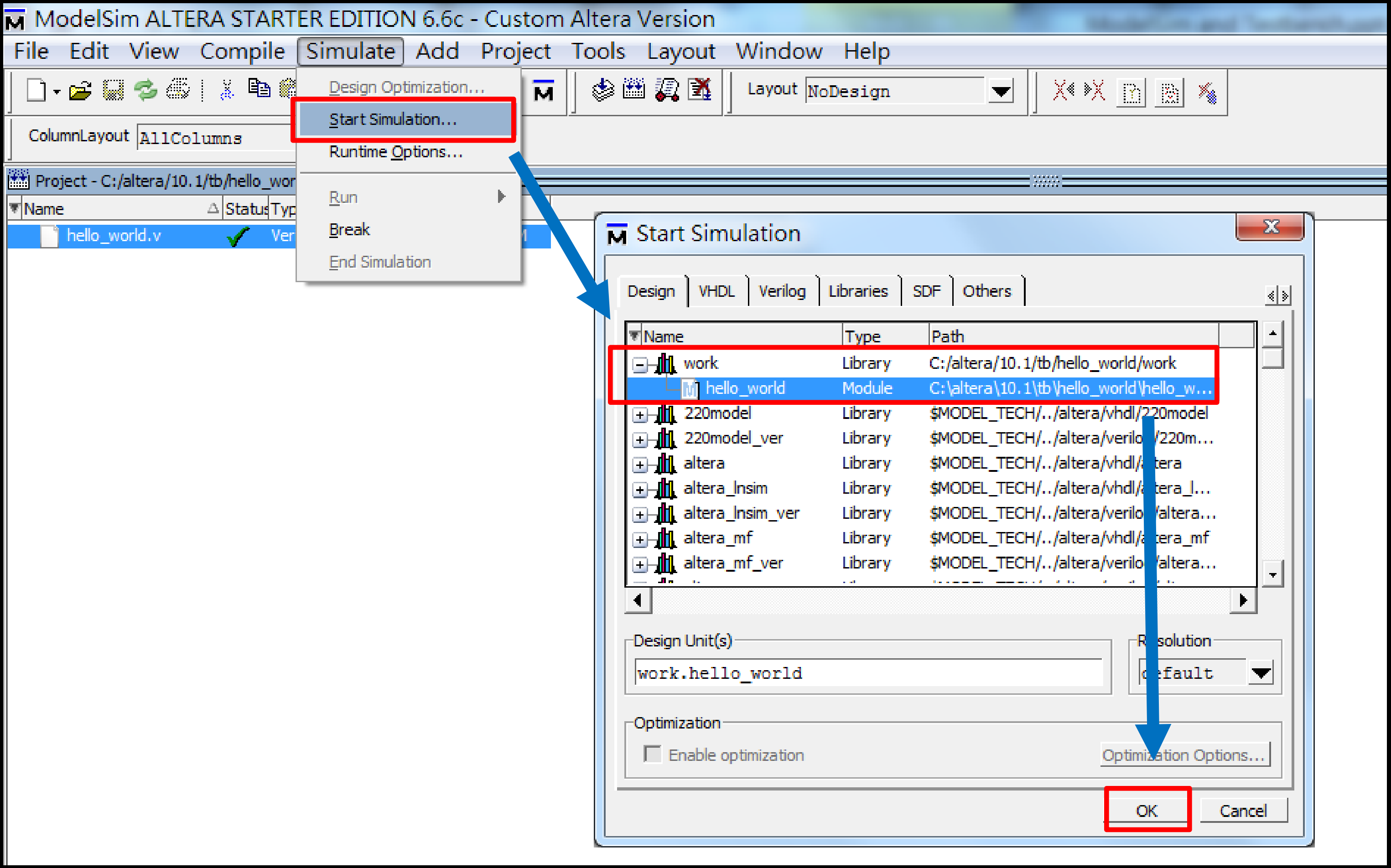
- **Compile -> Compile All**


ModelSim ALTERA STARTER EDITION 6.6c - Custom Alte
File Edit View Compile Simulate Add Project Too
ColumnLayout AllColumns
Project - C:/altera/10.1/tb/hello_world/hello_world

Name	Status	Type	Order	Modified
hello_world.v	✓	Verilog	0	04/10/13 03:13:14 PM

Start Simulation

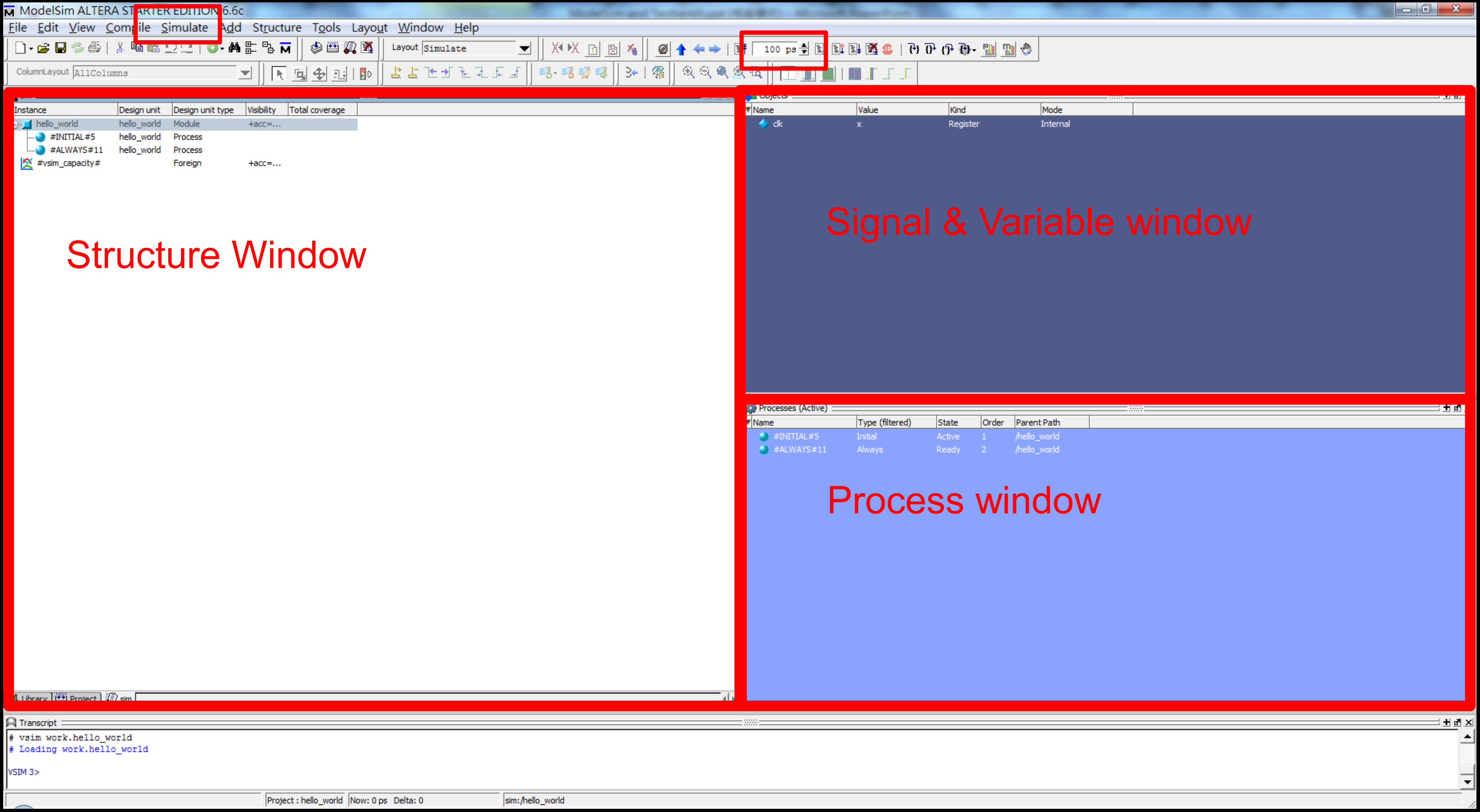
- **Simulate -> Start Simulation**



Name	Type	Path
work	Library	C:/altera/10.1/tb/hello_world/work
hello_world	Module	C:\altera\10.1\tb\hello_world\hello_w...
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220m...
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_l...
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera...
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera...

Design Unit(s): work.hello_world

Run Simulation



Instance	Design unit	Design unit type	Visibility	Total coverage
hello_world	hello_world	Module	+acc=...	
#INITIAL#5	hello_world	Process		
#ALWAYS#11	hello_world	Process		
#vsim_capacity#		Foreign	+acc=...	

Objects

Name	Value	Kind	Mode
clk	x	Register	Internal

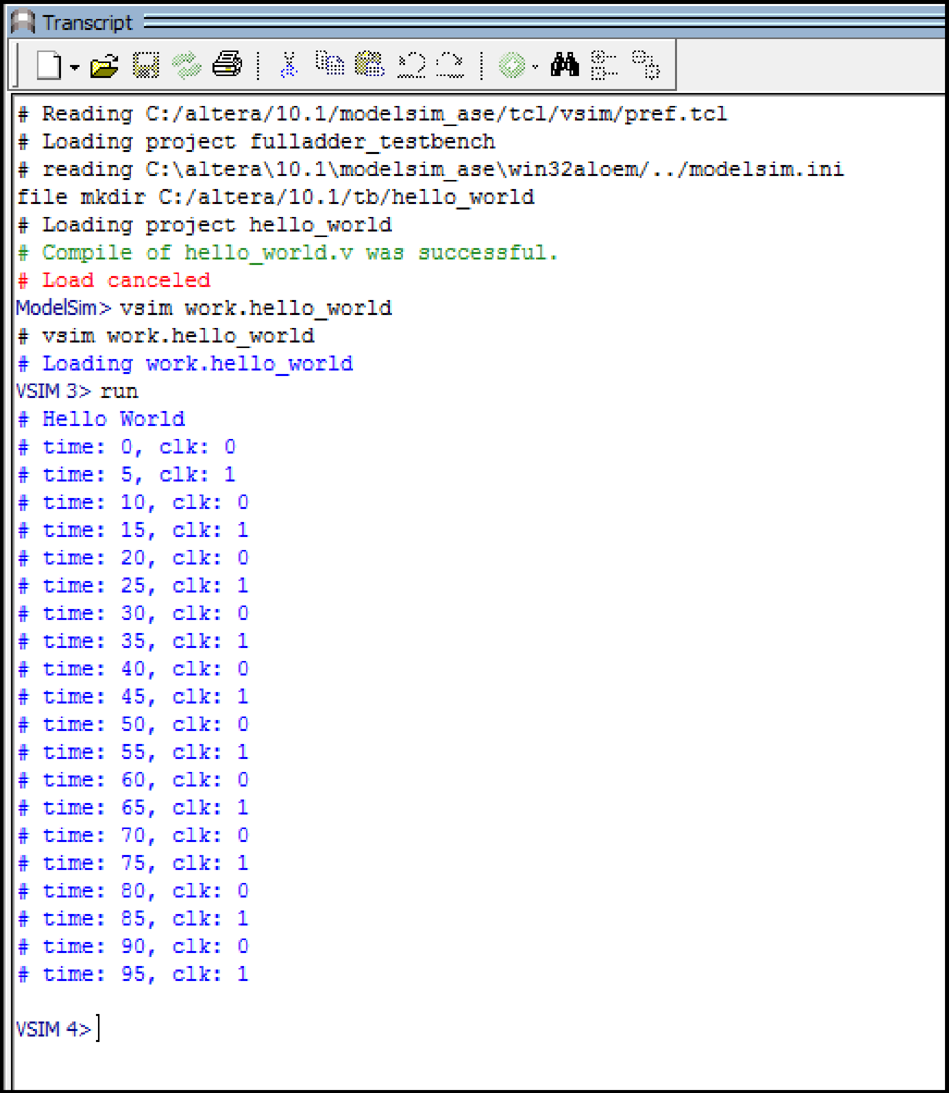
Processes (Active)

Name	Type (filtered)	State	Order	Parent Path
#INITIAL#5	Initial	Active	1	/hello_world
#ALWAYS#11	Always	Ready	2	/hello_world

```
# vsim work.hello_world
# Loading work.hello_world

VSIM 3>
```

Project : hello_world   Now: 0 ps   Delta: 0   sim:/hello_world

Simulation Result

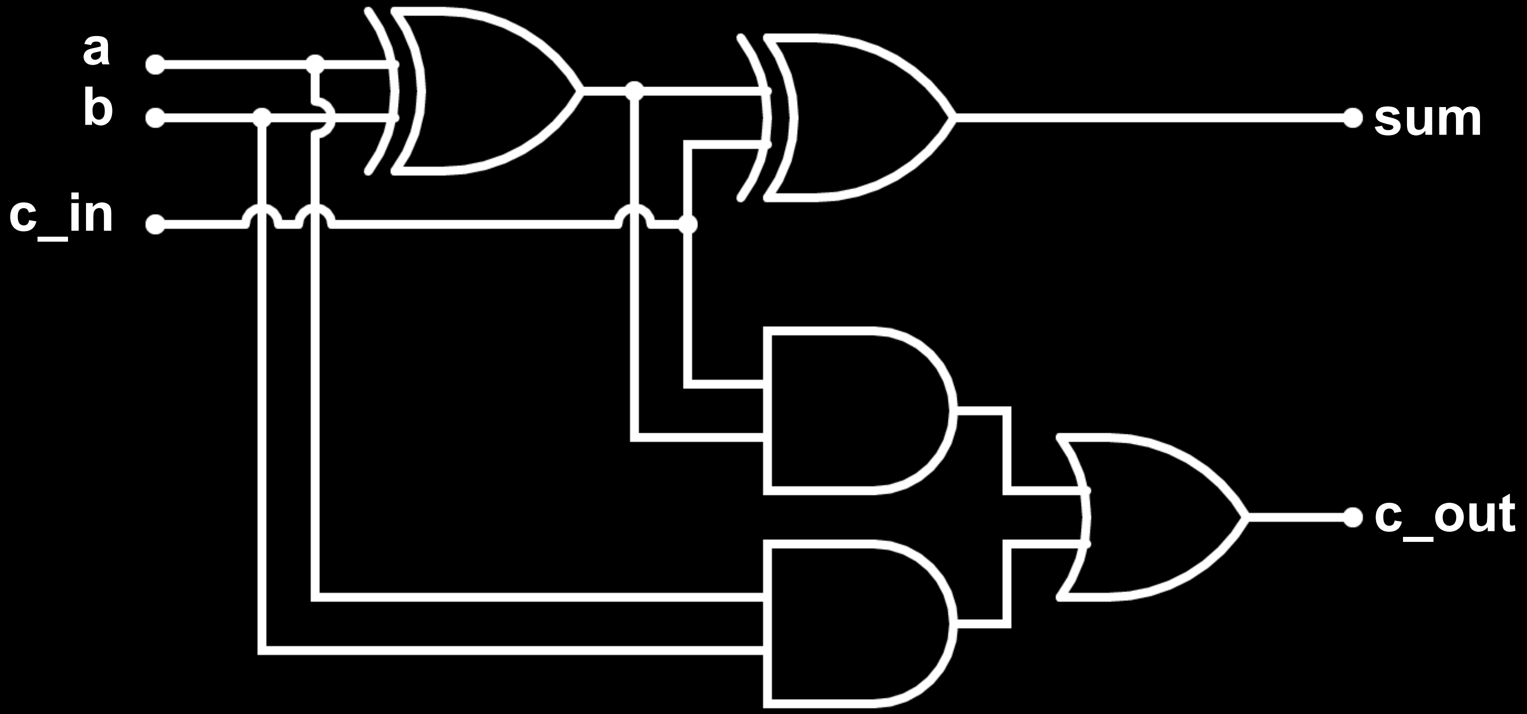


```
# Reading C:/altera/10.1/modelsim_ase/tcl/vsim/pref.tcl
# Loading project fulladder_testbench
# reading C:\altera\10.1\modelsim_ase\win32aloem/../modelsim.ini
file mkdir C:/altera/10.1/tb/hello_world
# Loading project hello_world
# Compile of hello_world.v was successful.
# Load canceled
ModelSim> vsim work.hello_world
# vsim work.hello_world
# Loading work.hello_world
VSIM 3> run
# Hello World
# time: 0, clk: 0
# time: 5, clk: 1
# time: 10, clk: 0
# time: 15, clk: 1
# time: 20, clk: 0
# time: 25, clk: 1
# time: 30, clk: 0
# time: 35, clk: 1
# time: 40, clk: 0
# time: 45, clk: 1
# time: 50, clk: 0
# time: 55, clk: 1
# time: 60, clk: 0
# time: 65, clk: 1
# time: 70, clk: 0
# time: 75, clk: 1
# time: 80, clk: 0
# time: 85, clk: 1
# time: 90, clk: 0
# time: 95, clk: 1

VSIM 4>
```

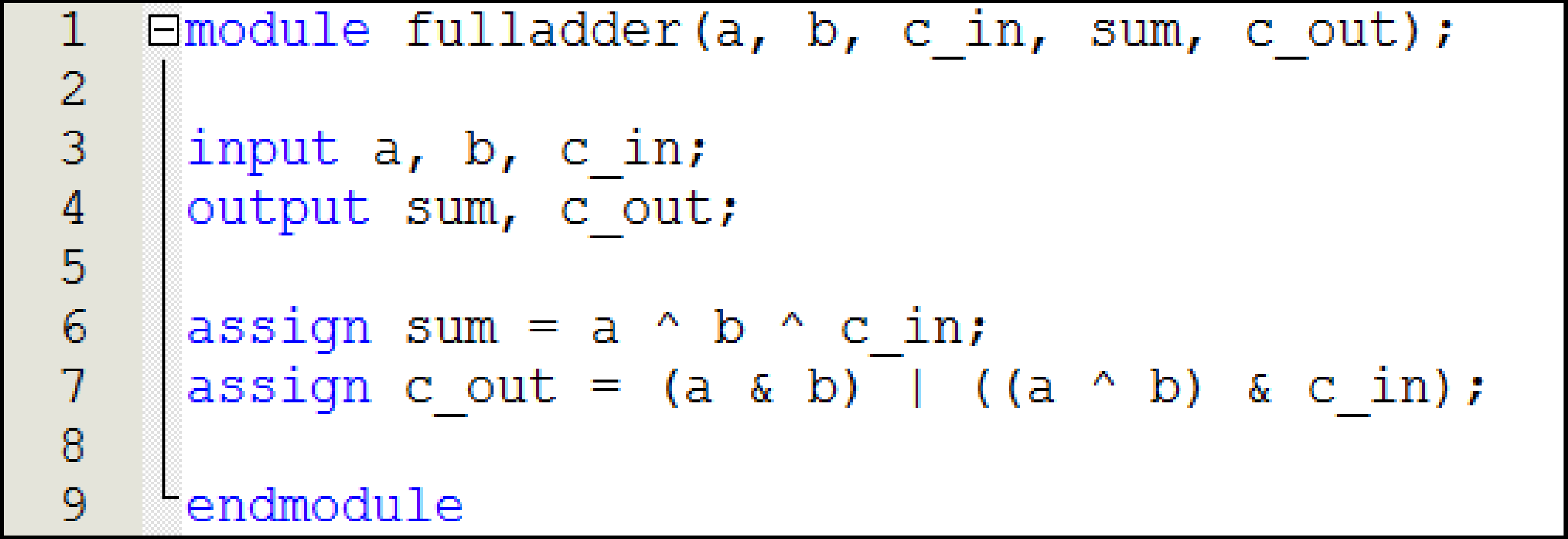
Outline

- **Introduction**
- **Hello ModelSim**
- **Testbench Tutorial**
  - Lab 1: 1-bit Full Adder
  - Lab 2: Counter
- **Exercise**
  - Lab 3: Seg7viewer

Lab 1: 1-bit Full Adder



Verilog Design

- 

```
module fulladder(a, b, c_in, sum, c_out);

input a, b, c_in;
output sum, c_out;

assign sum = a ^ b ^ c_in;
assign c_out = (a & b) | ((a ^ b) & c_in);

endmodule
```

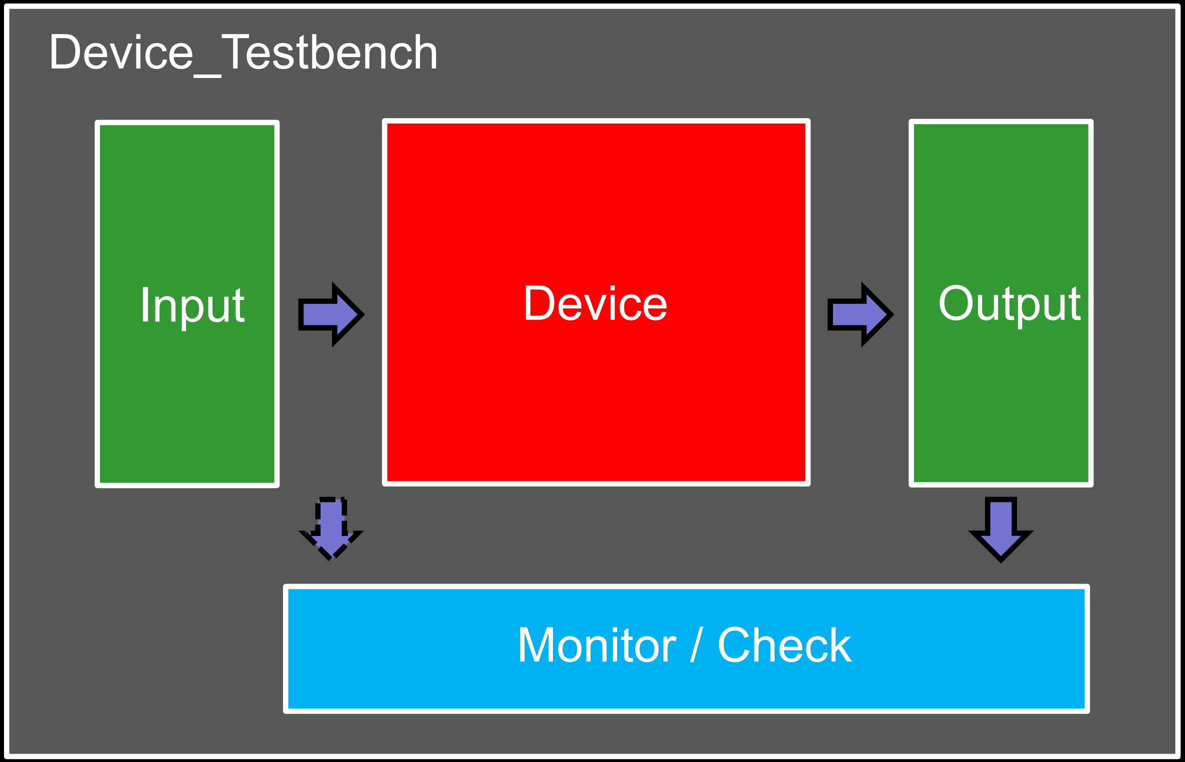
- **How to verify?**

Truth Table

a	b	c_in		sum	c_out
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1

Test Plan

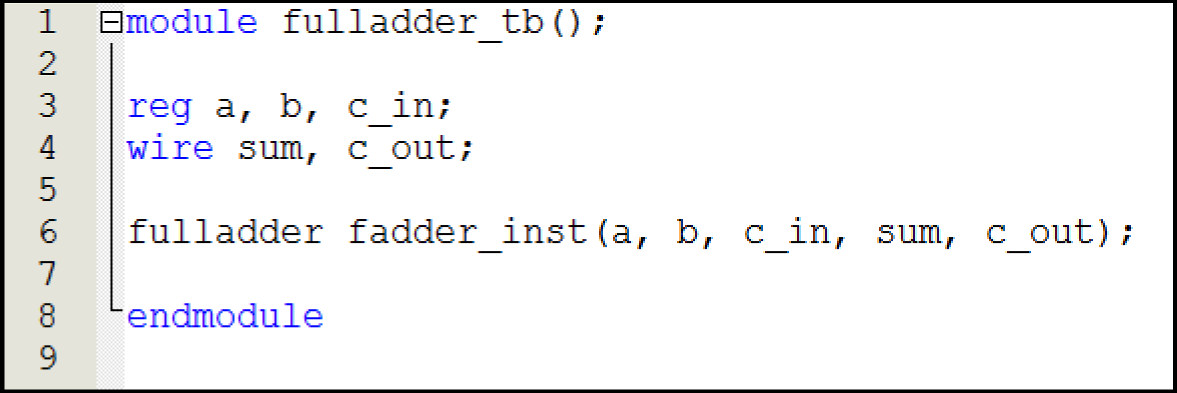
- **Device under test (DUT)**



- **What is the first step?**

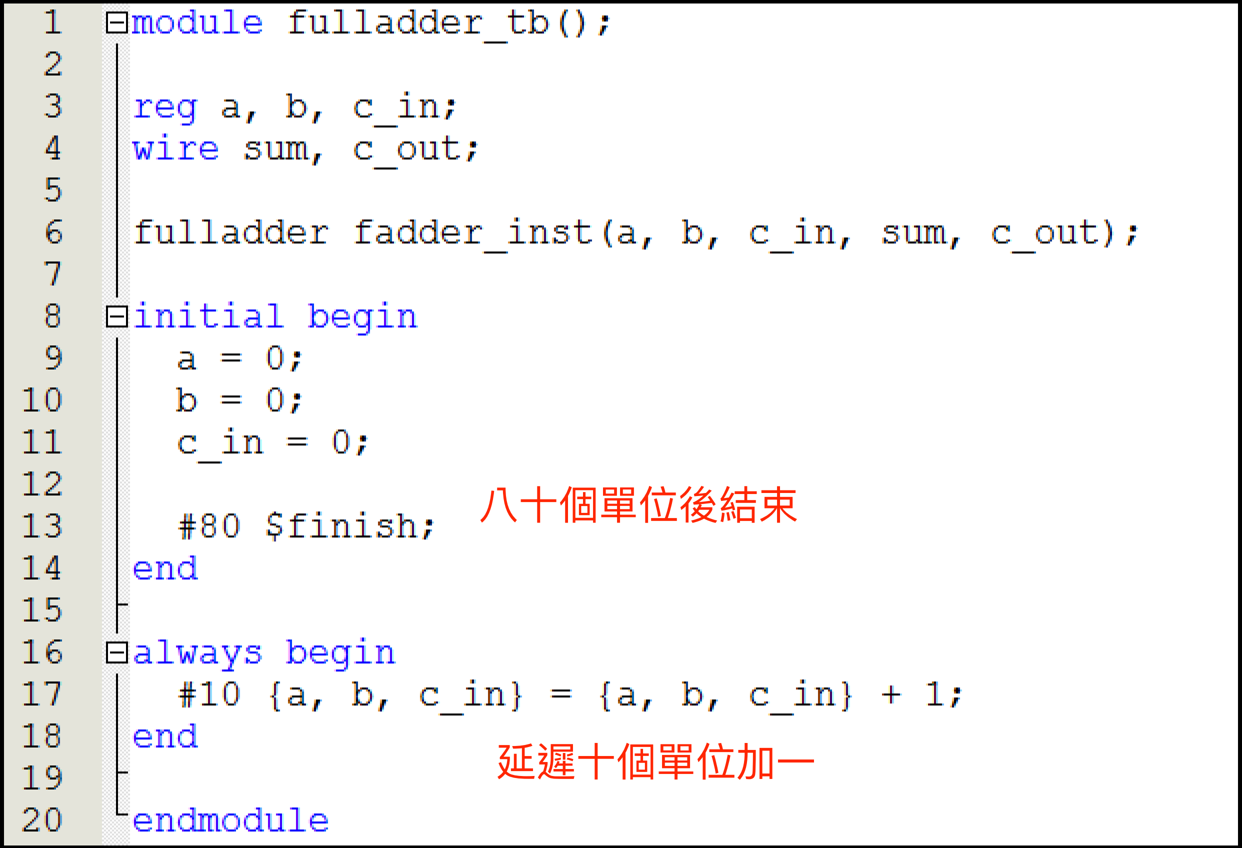
Write a Testbench

- **Create a top module of testbench**



```
module fulladder_tb();

reg a, b, c_in;
wire sum, c_out;

fulladder fadder_inst(a, b, c_in, sum, c_out);

endmodule
```

Test Case



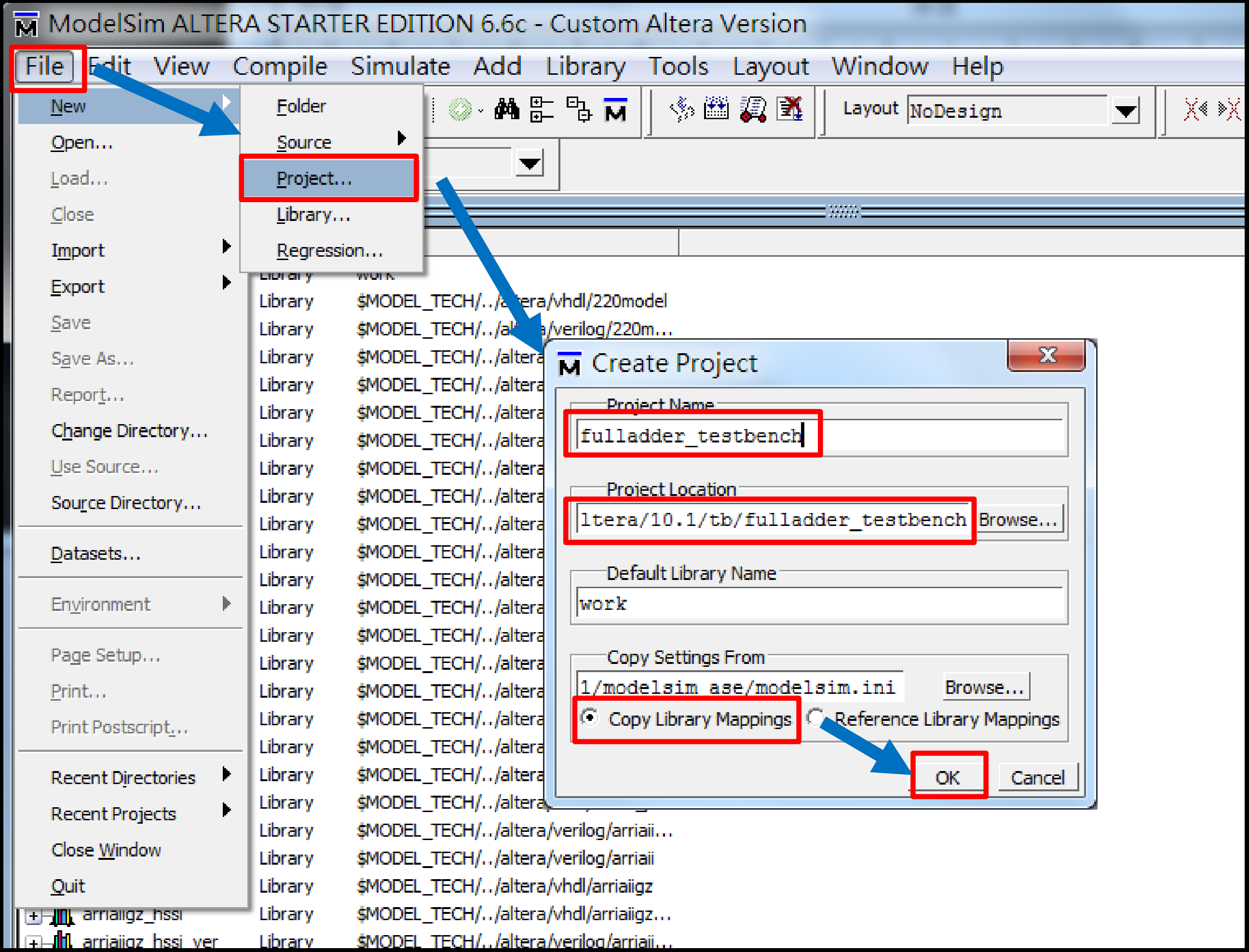
```
module fulladder_tb();

reg a, b, c_in;
wire sum, c_out;

fulladder fadder_inst(a, b, c_in, sum, c_out);

initial begin
  a = 0;
  b = 0;
  c_in = 0;

  #80 $finish;
end

always begin
  #10 {a, b, c_in} = {a, b, c_in} + 1;
end

endmodule
```

八十個單位後結束

延遲十個單位加一

Create Project

- **File -> New -> Project**



Create Project

Project Name: fulladder_testbench

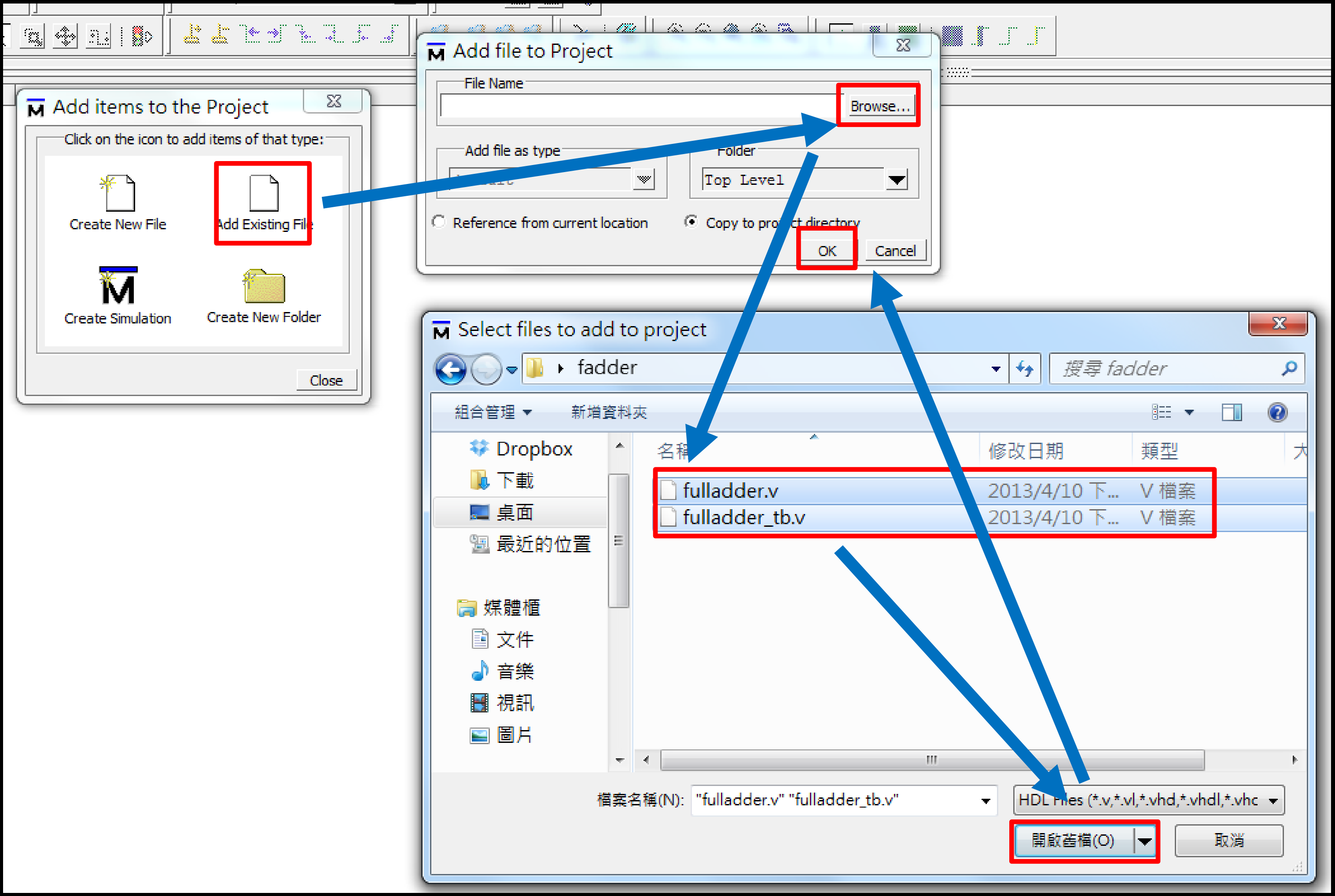
Project Location: ltera/10.1/tb/fulladder_testbench

Default Library Name: work

Copy Settings From: 1/modelsim_ase/modelsim.ini

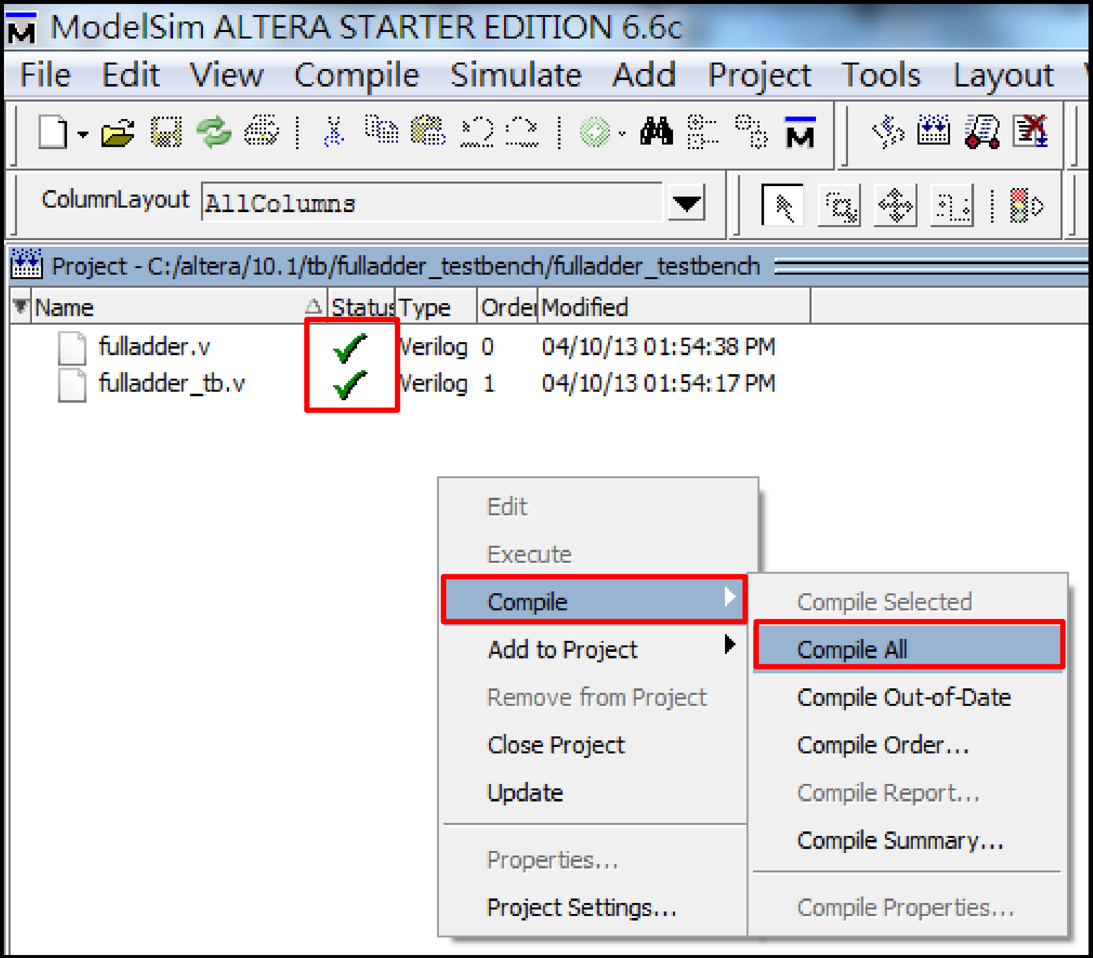
[✓ Copy Library Mappings] Reference Library Mappings

Add Existing File



Compile All

- **Compile -> Compile All**


ModelSim ALTERA STARTER EDITION 6.6c

File Edit View Compile Simulate Add Project Tools Layout

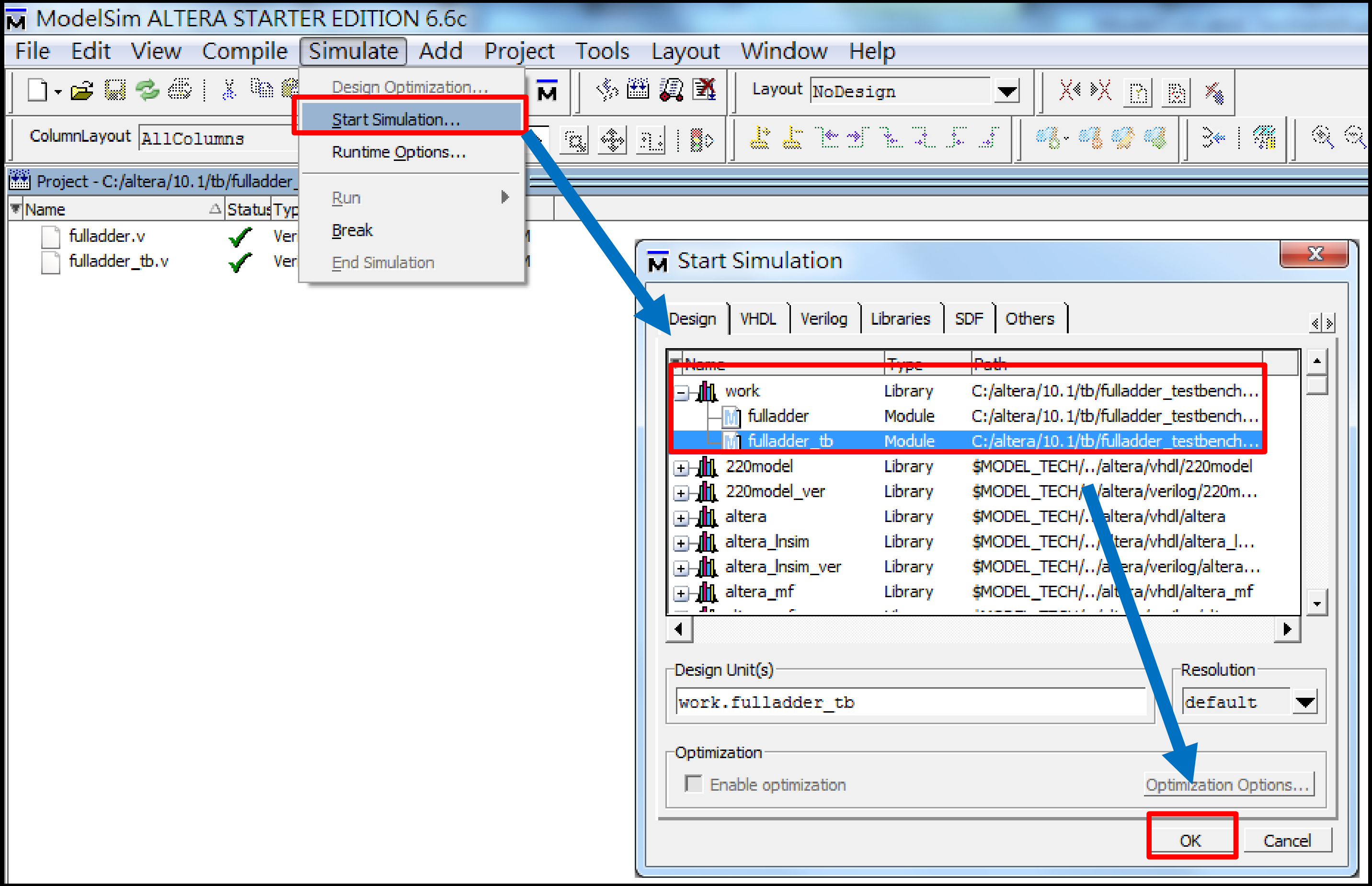
ColumnLayout AllColumns

Project - C:/altera/10.1/tb/fulladder_testbench/fulladder_testbench

Name	Status	Type	Order	Modified
fulladder.v	✓	Verilog	0	04/10/13 01:54:38 PM
fulladder_tb.v	✓	Verilog	1	04/10/13 01:54:17 PM

Start Simulation

- **Simulate -> Start Simulation**

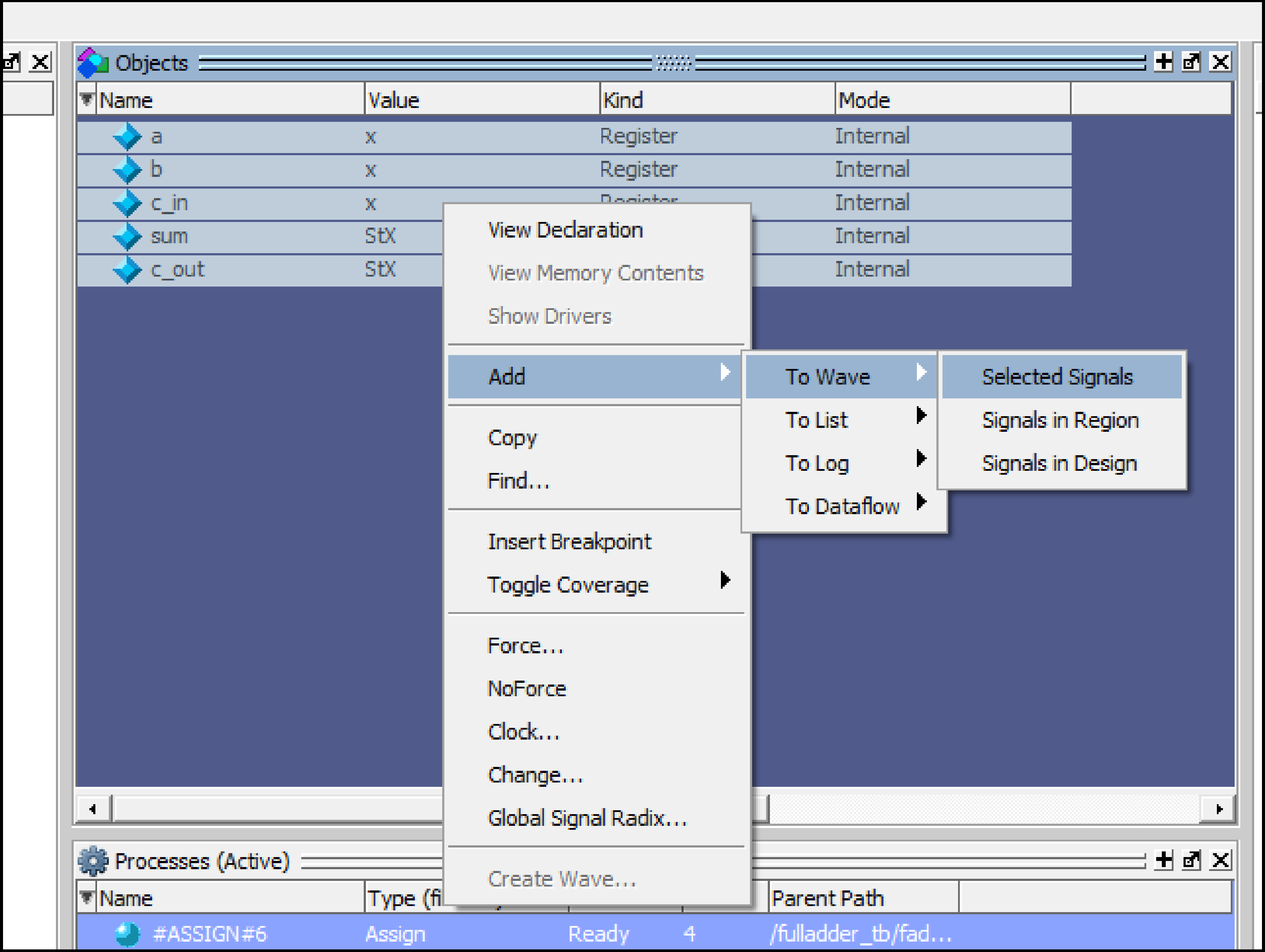


Name	Type	Path
work	Library	C:/altera/10.1/tb/fulladder_testbench...
fulladder	Module	C:/altera/10.1/tb/fulladder_testbench...
fulladder_tb	Module	C:/altera/10.1/tb/fulladder_testbench...
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220m...
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_l...
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera...
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf

Design Unit(s): work.fulladder_tb

Resolution: default

Monitor



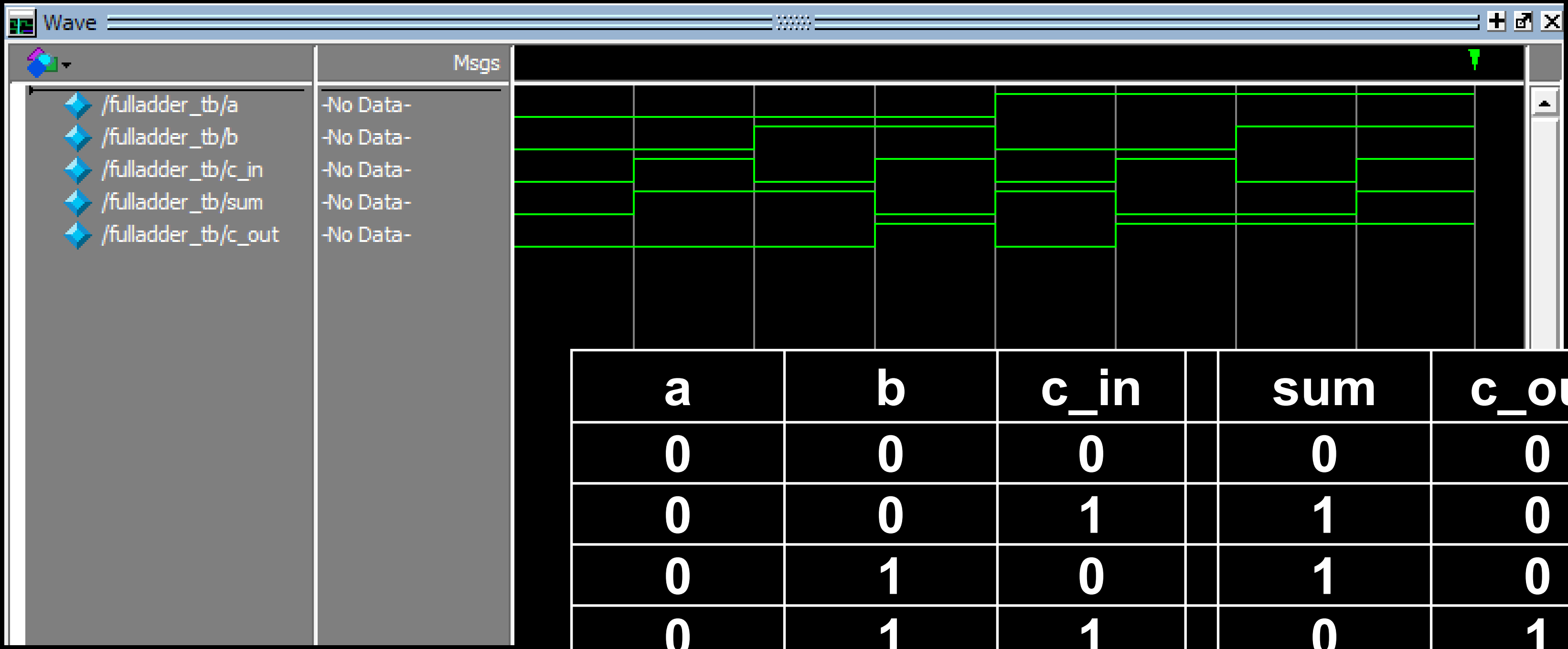
Objects

Name	Value	Kind	Mode
a	x	Register	Internal
b	x	Register	Internal
c_in	x	Register	Internal
sum	StX		Internal
c_out	StX		Internal

Processes (Active)

Name	Type (fi			Parent Path
#ASSIGN#6	Assign	Ready	4	/fulladder_tb/fad...

Run Simuation

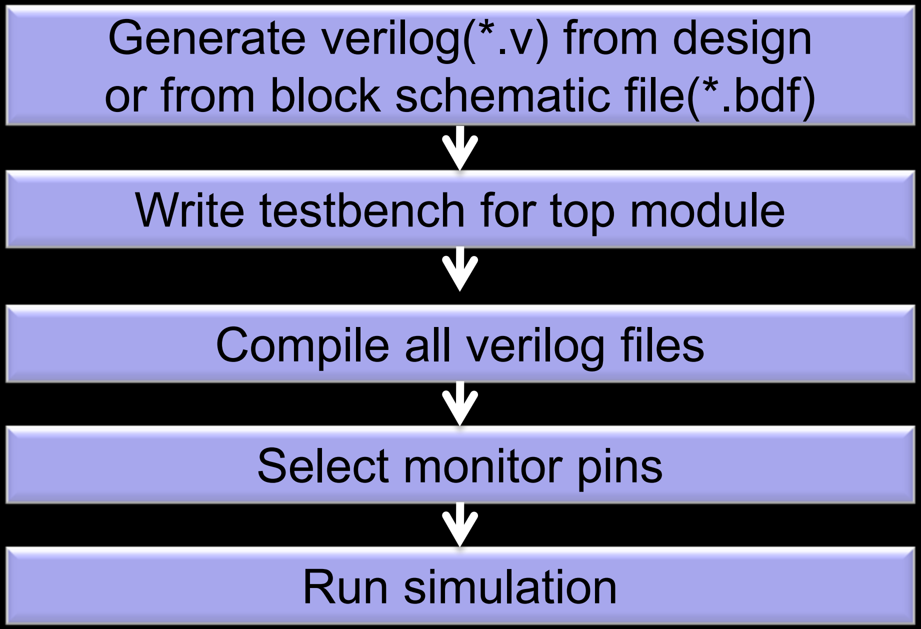


Wave

	Msgs
/fulladder_tb/a	-No Data-
/fulladder_tb/b	-No Data-
/fulladder_tb/c_in	-No Data-
/fulladder_tb/sum	-No Data-
/fulladder_tb/c_out	-No Data-

a	b	c_in		sum	c_out
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1

Summary for Test Flow

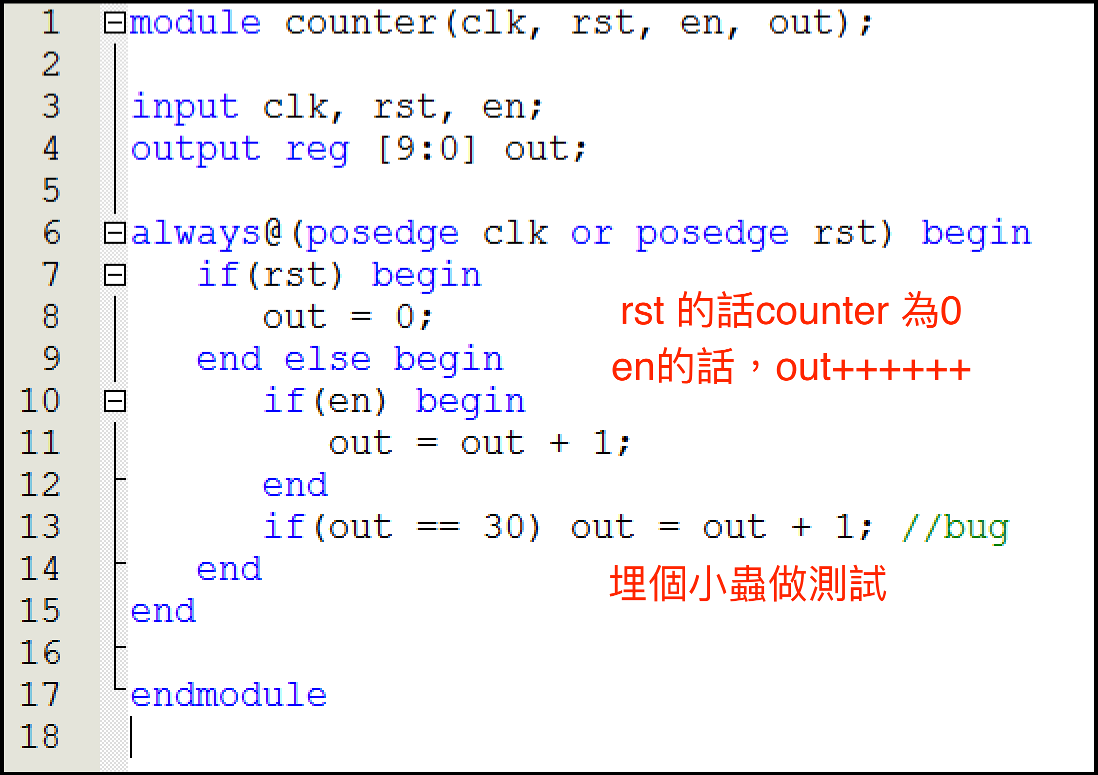

Generate verilog(*.v) from design or from block schematic file(*.bdf)
Write testbench for top module
Compile all verilog files
Select monitor pins
Run simulation

Outline

- **Introduction**
- **Hello ModelSim**
- **Testbench Tutorial**
  - Lab 1: 1-bit Full Adder
  - Lab 2: Counter
- **Exercise**
  - Lab 3: Seg7viewer

Lab 2: Counter

- **counter.v**



```
module counter(clk, rst, en, out);

input clk, rst, en;
output reg [9:0] out;

always@(posedge clk or posedge rst) begin
    if(rst) begin
        out = 0;
    end else begin
        if(en) begin
            out = out + 1;
        end
        if(out == 30) out = out + 1; //bug
    end
end

endmodule
```

rst 的話counter 為0

en的話，out++++++

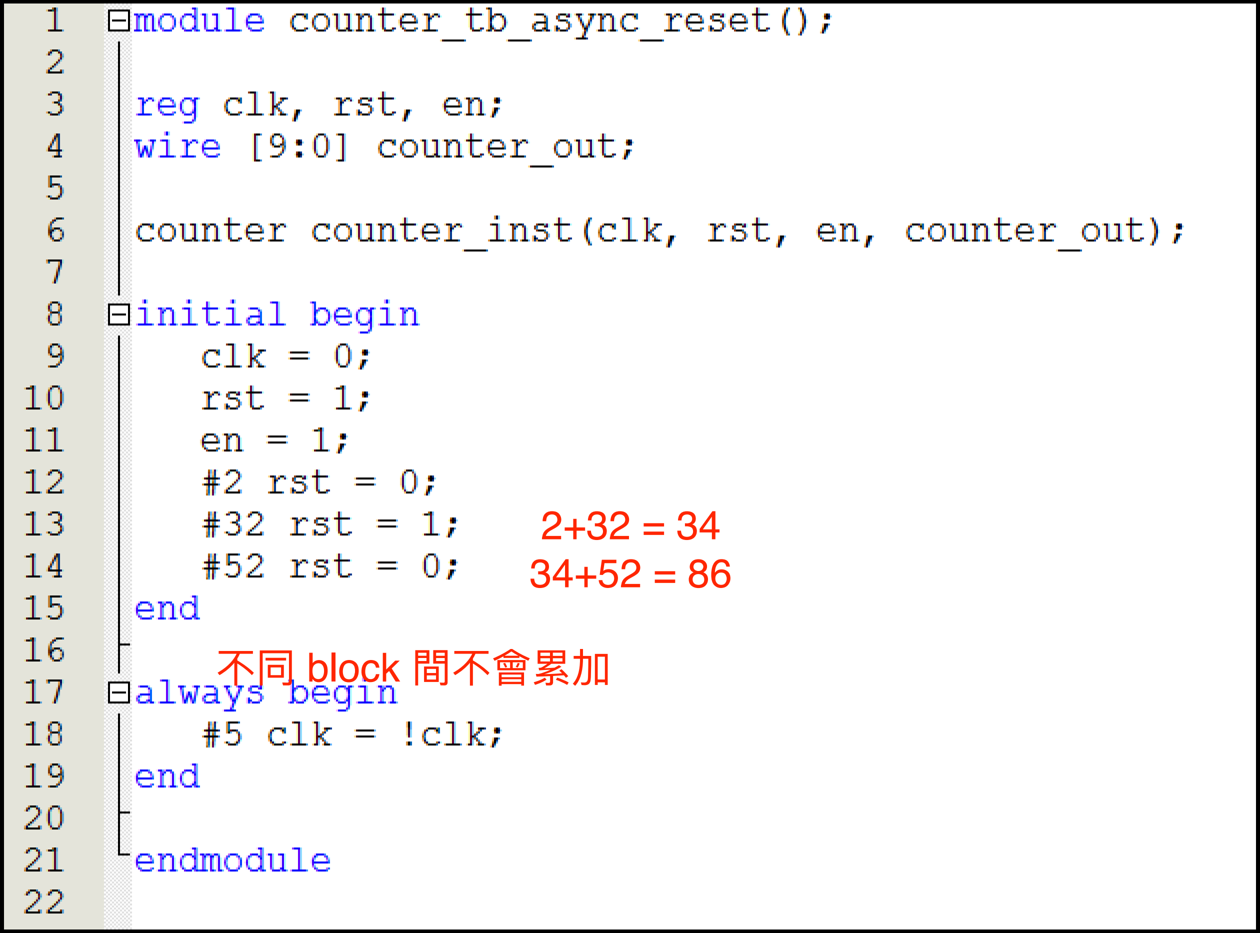
埋個小蟲做測試

Test Plan

- **Test case:**
  1. Functionality of reset
  2. Under irregular clock 打不規律的頻率看會怎樣
  3. Self-checking test 自動化測試

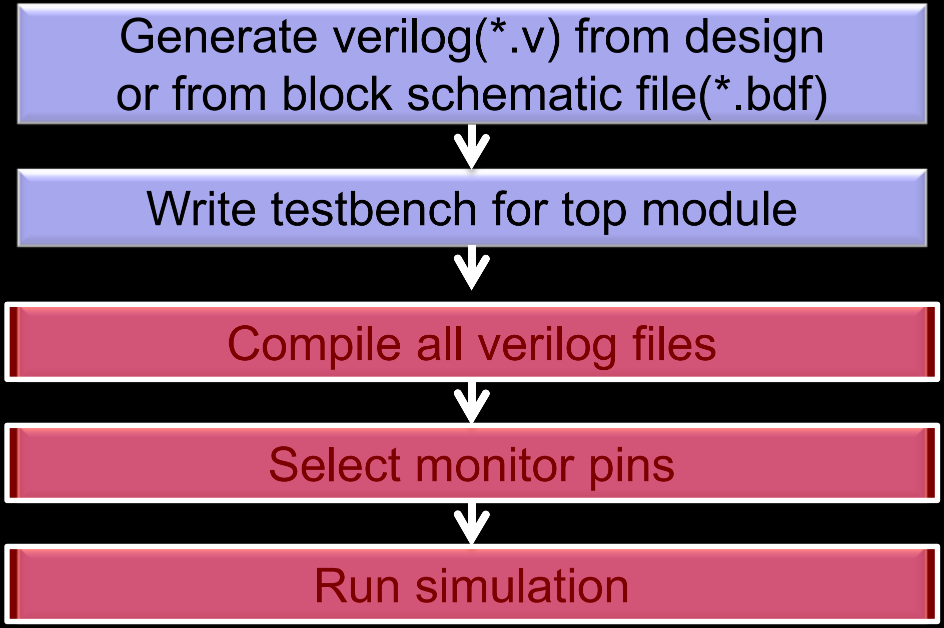
Test Case 1

- **Test the functionality of async-reset**

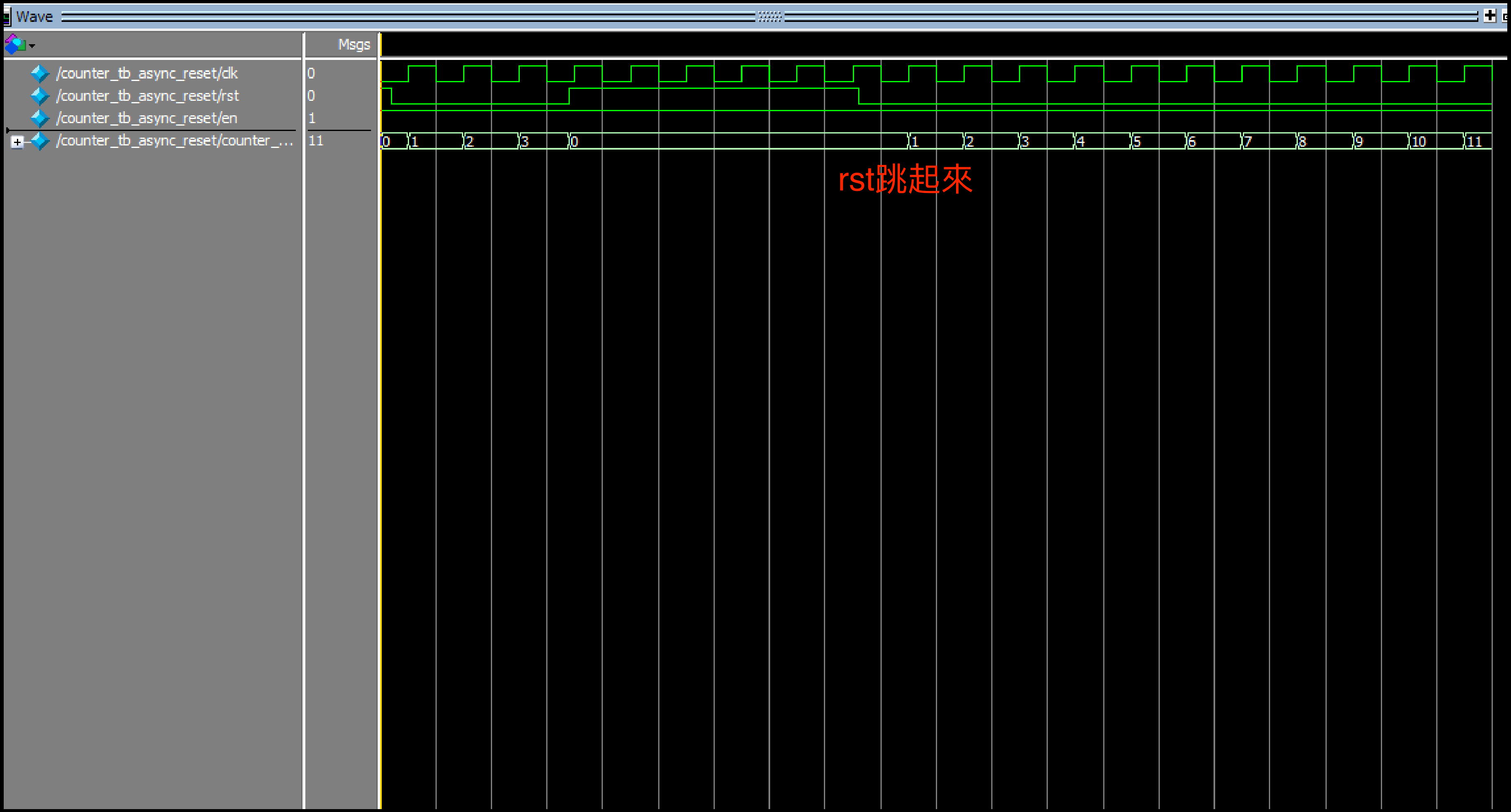


```
module counter_tb_async_reset();

reg clk, rst, en;
wire [9:0] counter_out;

counter counter_inst(clk, rst, en, counter_out);

initial begin
    clk = 0;
    rst = 1;
    en = 1;
    #2 rst = 0;
    #32 rst = 1;
    #52 rst = 0;
end

always begin
    #5 clk = !clk;
end

endmodule
```

Test Flow

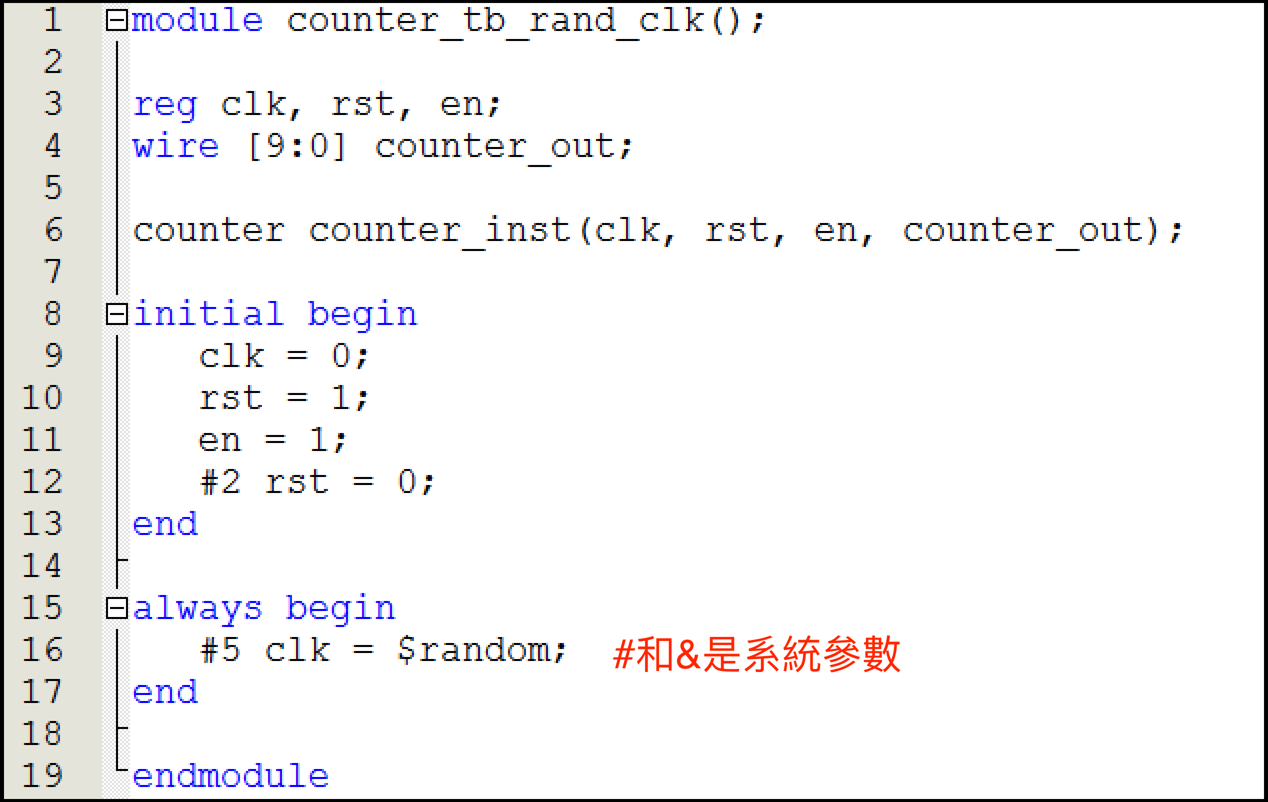

Generate verilog(*.v) from design or from block schematic file(*.bdf)
Write testbench for top module
Compile all verilog files
Select monitor pins
Run simulation

Simulation Result


Wave
Msgs
/counter_tb_async_reset/clk 0
/counter_tb_async_reset/rst 0
/counter_tb_async_reset/en 1
/counter_tb_async_reset/counter_... 11
0 1 2 3 0 1 2 3 4 5 6 7 8 9 10 11
rst跳起來

Test Case 2

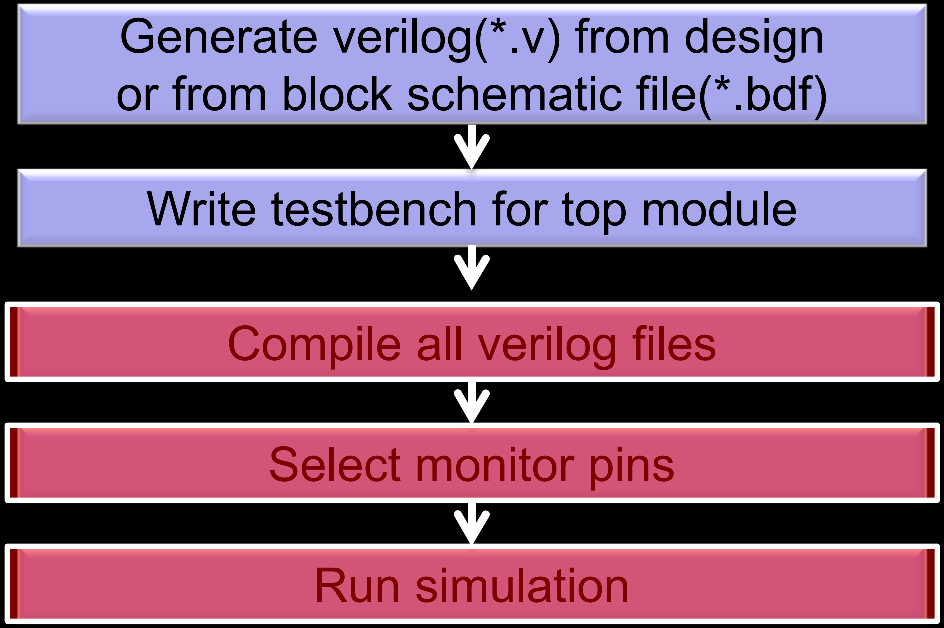
- Add some random factors
- Test the irregular clock



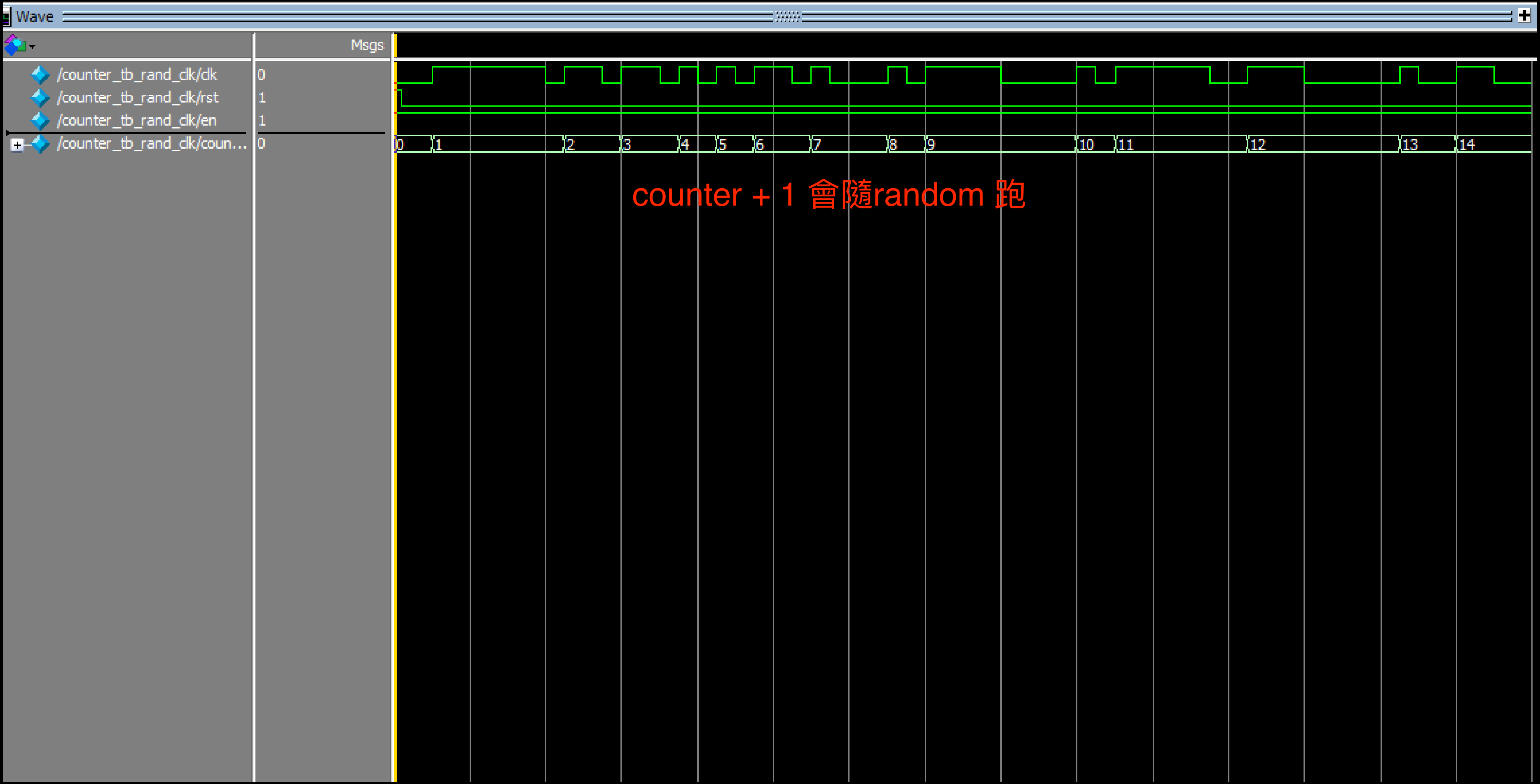
```
module counter_tb_rand_clk();

reg clk, rst, en;
wire [9:0] counter_out;

counter counter_inst(clk, rst, en, counter_out);

initial begin
   clk = 0;
   rst = 1;
   en = 1;
   #2 rst = 0;
end

always begin
   #5 clk = $random;
end

endmodule
```

#和&是系統參數

Test Flow


Generate verilog(*.v) from design or from block schematic file(*.bdf)
Write testbench for top module
Compile all verilog files
Select monitor pins
Run simulation

Simulation Result



```
/counter_tb_rand_clk/clk     0
/counter_tb_rand_clk/rst     1
/counter_tb_rand_clk/en      1
/counter_tb_rand_clk/coun... 0
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

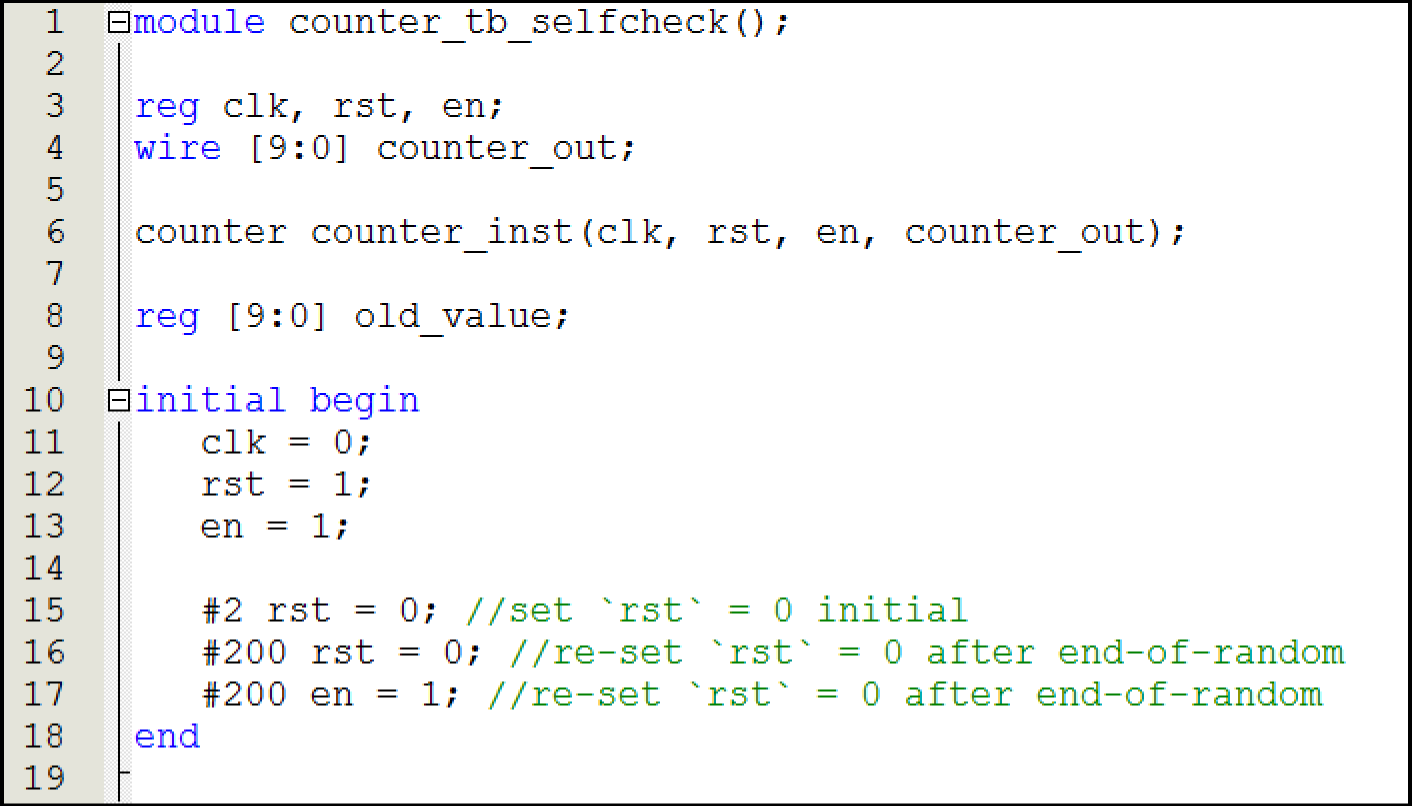
Test Case 3

有些時候肉眼看不出來，所以請機器幫我們測試

- Make testbench to be automated

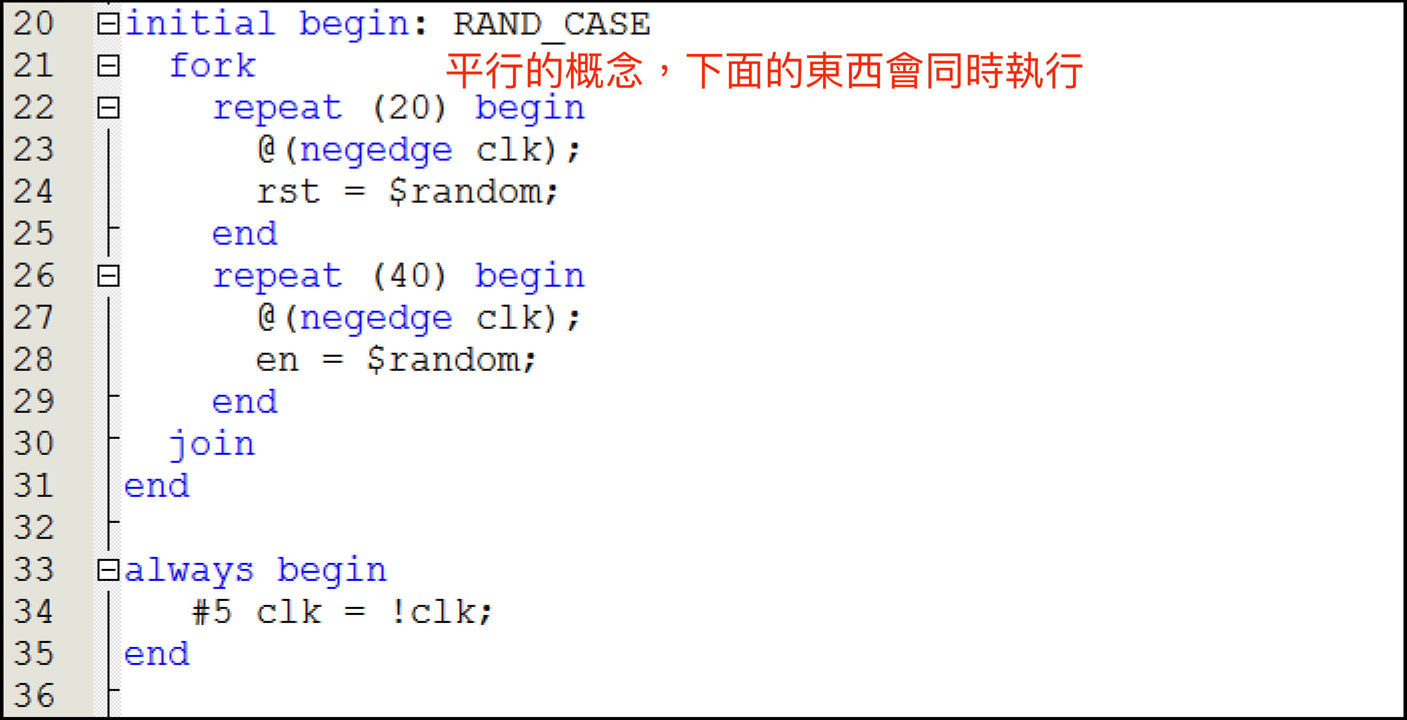
- Check Rules (pseudo code):
  - if reset: count is 0
  - if not enable: count is count_old
  - if not reset and enable: count is (count_old + 1)

Counter_tb_selfcheck.v - 1



```
module counter_tb_selfcheck();

reg clk, rst, en;
wire [9:0] counter_out;

counter counter_inst(clk, rst, en, counter_out);

reg [9:0] old_value;

initial begin
   clk = 0;
   rst = 1;
   en = 1;

   #2 rst = 0; //set `rst` = 0 initial
   #200 rst = 0; //re-set `rst` = 0 after end-of-random
   #200 en = 1; //re-set `rst` = 0 after end-of-random
end

```

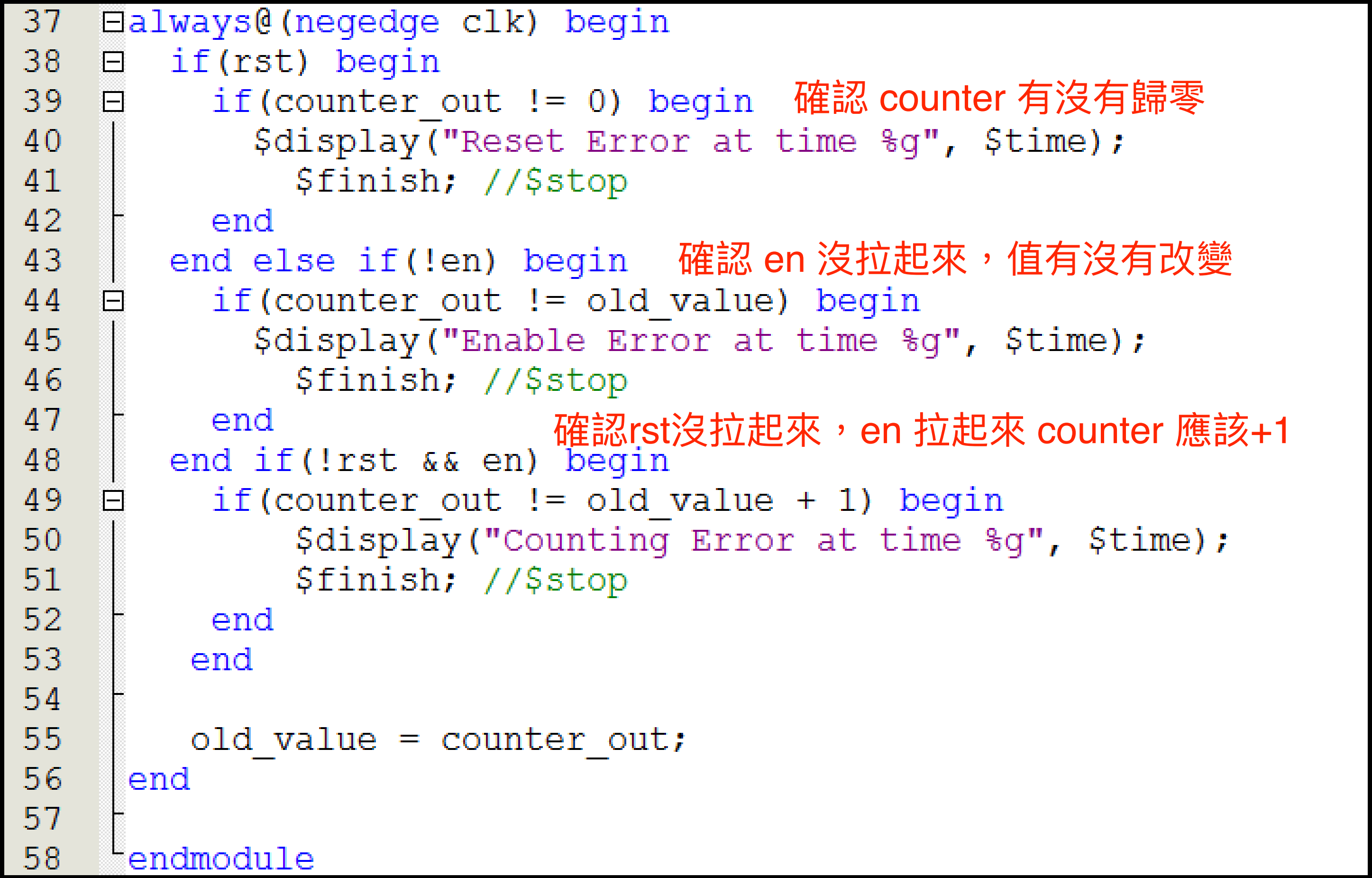
Counter_tb_selfcheck.v - 2



```
initial begin: RAND_CASE
  fork
    repeat (20) begin
      @(negedge clk);
      rst = $random;
    end
    repeat (40) begin
      @(negedge clk);
      en = $random;
    end
  join
end

always begin
   #5 clk = !clk;
end
```

平行的概念，下面的東西會同時執行

Counter_tb_selfcheck.v - 3



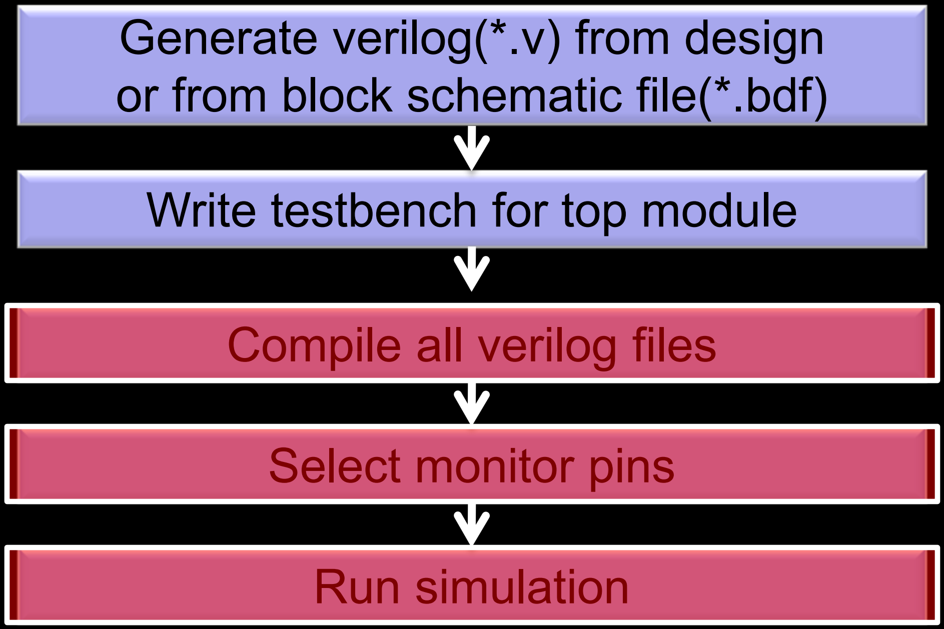
```verilog
always@(negedge clk) begin
  if(rst) begin
    if(counter_out != 0) begin
      $display("Reset Error at time %g", $time);
        $finish; //$stop
    end
  end else if(!en) begin
    if(counter_out != old_value) begin
      $display("Enable Error at time %g", $time);
        $finish; //$stop
    end
  end if(!rst && en) begin
    if(counter_out != old_value + 1) begin
        $display("Counting Error at time %g", $time);
        $finish; //$stop
    end
   end

   old_value = counter_out;
end

endmodule
```

確認 counter 有沒有歸零

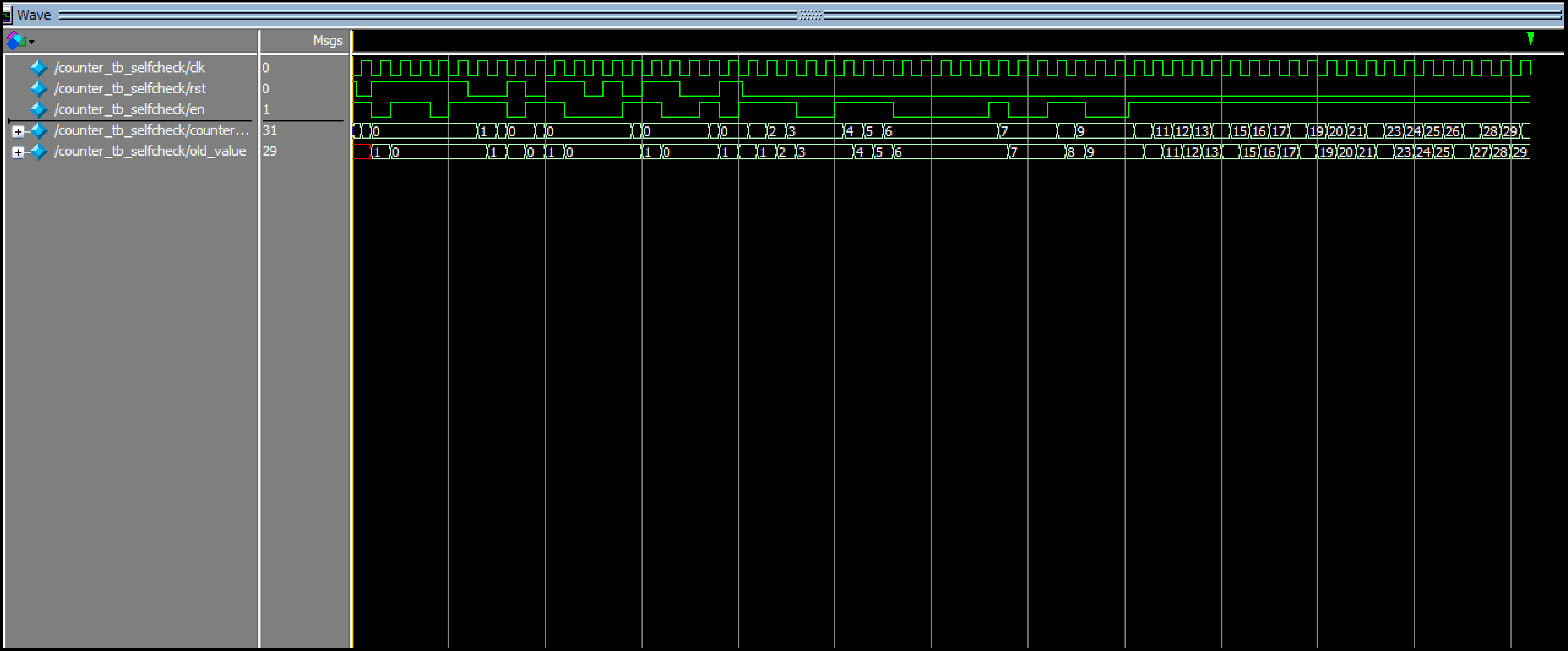
確認 en 沒拉起來，值有沒有改變

確認rst沒拉起來，en 拉起來 counter 應該+1

Simulation Flow


Generate verilog(*.v) from design or from block schematic file(*.bdf)
Write testbench for top module
Compile all verilog files
Select monitor pins
Run simulation

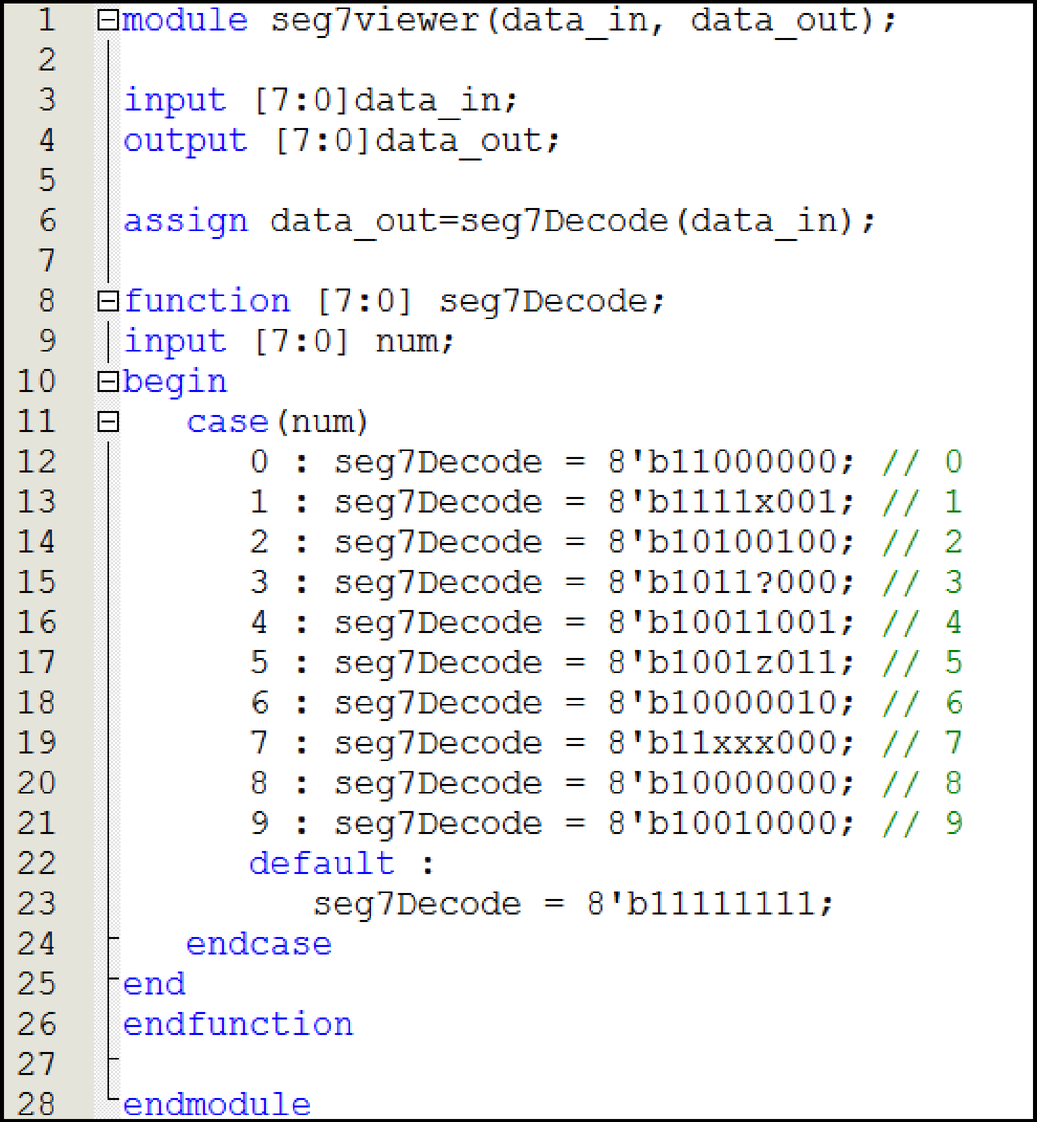
Simulation Result


Wave
Msgs
/counter_tb_selfcheck/clk 0
/counter_tb_selfcheck/rst 0
/counter_tb_selfcheck/en 1
/counter_tb_selfcheck/counter... 31
/counter_tb_selfcheck/old_value 29

Outline

- **Introduction**
- **Hello ModelSim**
- **Testbench Tutorial**
  - Lab 1: 1-bit Full Adder
  - Lab 2: Counter
- **Exercise**
  - Lab 3: Seg7viewer

Lab 3: Seg7viewer



```verilog
module seg7viewer(data_in, data_out);

input [7:0]data_in;
output [7:0]data_out;

assign data_out=seg7Decode(data_in);

function [7:0] seg7Decode;
input [7:0] num;
begin
    case(num)
        0 : seg7Decode = 8'b11000000; // 0
        1 : seg7Decode = 8'b1111x001; // 1
        2 : seg7Decode = 8'b10100100; // 2
        3 : seg7Decode = 8'b1011?000; // 3
        4 : seg7Decode = 8'b10011001; // 4
        5 : seg7Decode = 8'b1001z011; // 5
        6 : seg7Decode = 8'b10000010; // 6
        7 : seg7Decode = 8'b11xxx000; // 7
        8 : seg7Decode = 8'b10000000; // 8
        9 : seg7Decode = 8'b10010000; // 9
        default :
            seg7Decode = 8'b11111111;
    endcase
end
endfunction

endmodule
```

Test Plan

- Is the design reliable?
- What is the scenario?
- Are the faults under coverage?
- How to generate test case?
- How to write a self-checking testbench?
- etc

Reference

- **Art of Writing TestBenches**
  - http://www.asic-world.com/verilog/art_testbench_writing.html
- **Unit Testing Guidelines**
  - http://geosoft.no/development/unittesting.html