Combinational Logic
Ch. 4.13: Behavioral Modeling
Ch. 4.14: Writing a Simple Testbench

Ref. Digital Design, 6th Edition

SOP

- Write Verilog code
- Compile
- Run simulation
- Programmer to FPGA, verify the results
  - Assign board I/O pins => find FPGA I/O pins => assignment editor

HDL Models of Combinational Circuits

Modeling Styles

- **Gate-level modeling:**
  - **Using predefined and user-defined primitive gates**
- **Dataflow modeling:**
  - **Using continuous assignment statements (Boolean function)**
- **Behavior modeling:**
  - **Using procedural assignment statements**
  - **Mostly used to describe sequential circuits**

Review: Gate-level modeling

HDL Example 4.1

```
// Gate-level description of two-to-four-line decoder
// Refer to Fig. 4.19 with symbol E replaced by enable, for clarity.

module decoder_2x4_gates (D, A, B, enable);
  output      [0: 3]     D;
  input                  A, B;
  input                  enable;
  wire                   A_not, B_not, enable_not;

  not
    G1  (A_not, A),
    G2  (B_not, B),
    G3  (enable_not, enable);
  nand
    G4  (D[0], A_not, B_not, enable_not),
    G5  (D[1], A_not, B, enable_not),
    G6  (D[2], A, B_not, enable_not),
    G7  (D[3], A, B, enable_not);
endmodule
```

具體宣告邏輯閘與輸出、輸入

Review: Data-flow modeling

HDL Example 4.3

```
// Dataflow description of two-to-four-line decoder

// See Fig. 4.19. Note: The figure uses symbol E, but the
// Verilog model uses enable to clearly indicate functionality.

module decoder_2x4_df (                          // Verilog 2001, 2005 syntax
  output      [0: 3]      D,
  input                   A, B,
                          enable
);

  assign      D[0] = ~(~A & ~B & ~enable),
              D[1] = ~(~A & B & ~enable),
              D[2] = ~(A & ~B & ~enable),
              D[3] = ~(A & B & ~enable);
endmodule
```

使用 assign 指定輸出的 Boolean function

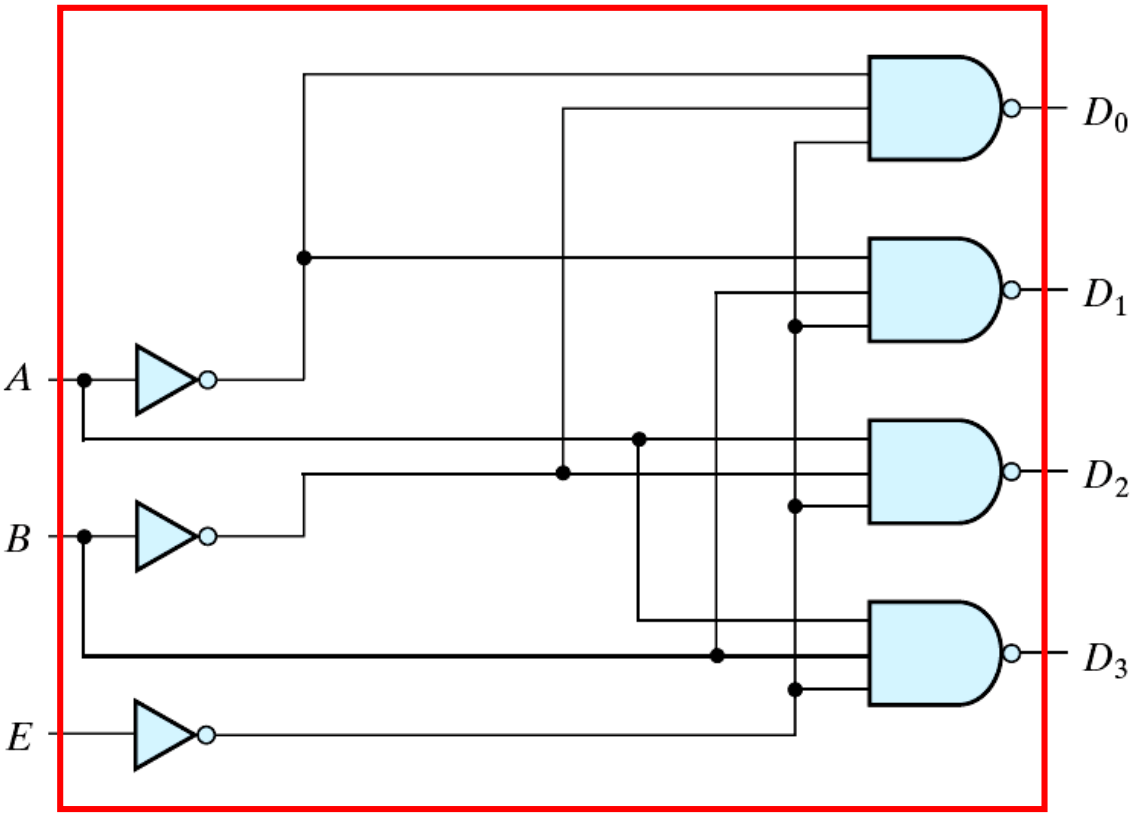
HDL Models of Combinational Circuits

Modeling Styles

- **Gate-level modeling:**
  - **Using predefined and user-defined primitive gates**
- **Dataflow modeling:**
  - **Using continuous assignment statements (Boolean function)**
- **Behavior modeling:**
  - **Using procedural assignment statements**
  - **Mostly used to describe sequential circuits**

What's the difference?

Continuous assignment

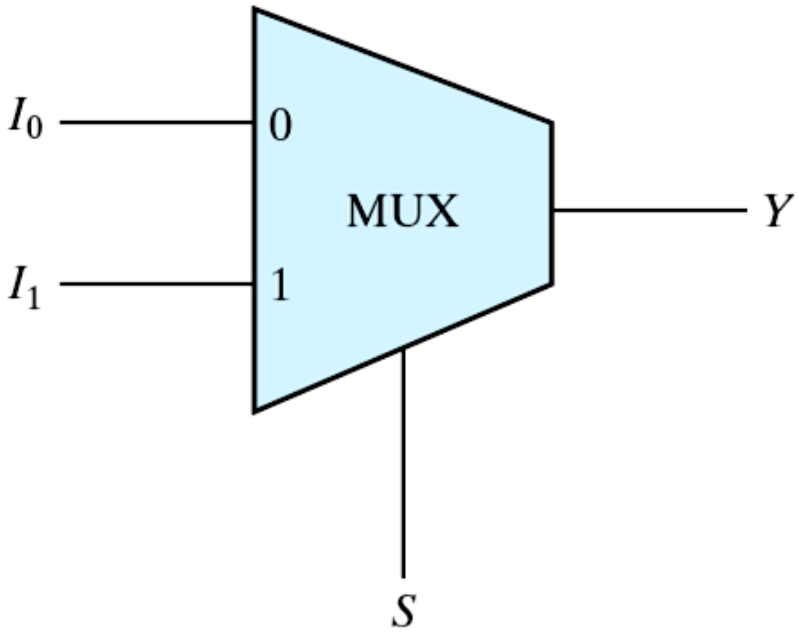


2x4 decoder

assign D[0] = ~(~A & ~B & ~enable)

*** fixed circuit **behavior** at all time

Procedural assignment (1)



Behavior:

if S == 0

$Y = I_0$

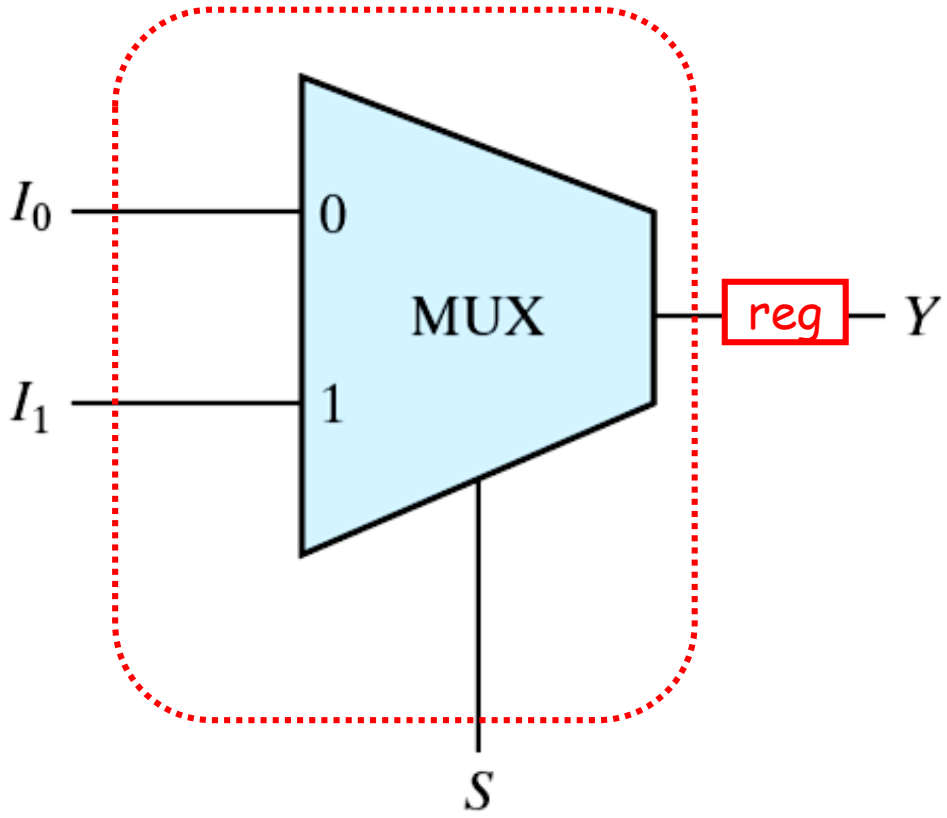
else S == 1

$Y = I_1$

*** circuit behavior depends on specific events

電路的行為不是固定的

Procedural assignment (2)



Procedural assignment
輸出必須是 reg (register) 型態

Outline

- **Behavior modeling**
  - **Example 4-9: 2-to-1-line multiplexer (FPGA)**
  - **Example 4-10: 4-to-1-line multiplexer (FPGA)**
  - **Example 4-11: Test bench for 2x1 MUX (Verilog & Simulation)**
  - **Example 4-13: Test bench for adder**

Behavioral Modeling

- **Procedural assignment:**
  - 關鍵字: always

- Behavioral description of a 2-to-1-line multiplexer

```
// Behavioral description of two-to-one-line multiplexer

module mux_2x1_beh (m_out, A, B, select);
 output      m_out;
 input       A, B, select;
 reg         m_out;

 always      @(A or B or select)
  if (select == 1) m_out = A;
  else m_out = B;
endmodule
```

Procedure assignment 的輸出必須是reg 變數

@ 後的變數有改變時，執行 always 內容

Procedure assignment

比較: HDL Example 4-7

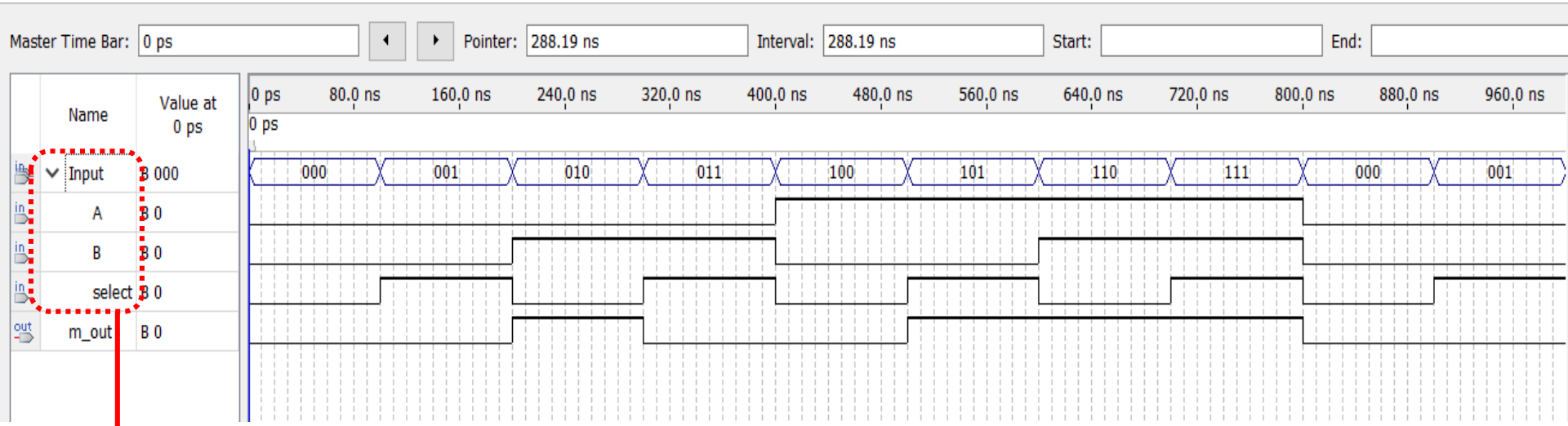
- **Dataflow description** of a 2-to-1-line multiplexer

```
// Dataflow description of two-to-one-line multiplexer

module mux_2x1_df(m_out, A, B, select);
  output        m_out;
  input         A, B;
  input         select;

  assign m_out = (select)? A : B;
endmodule
```

Lab#1-1: Waveform simulation of Example 4.9



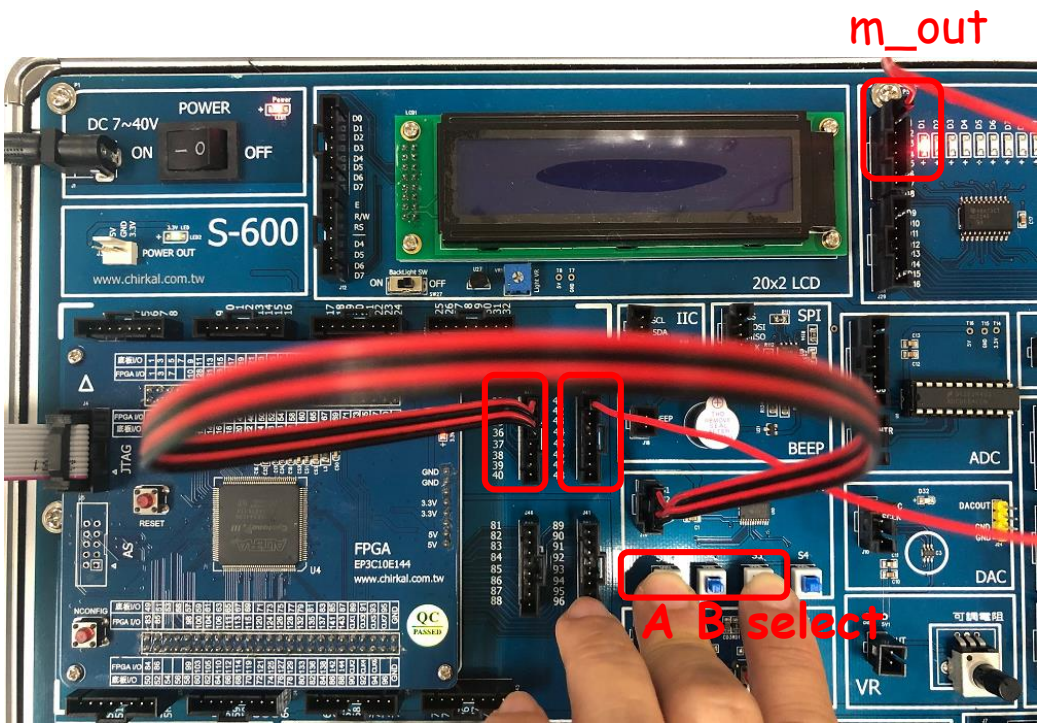
You must generate all possible inputs in this lab.

Lab1-2: FPGA implementation of Example 4.9

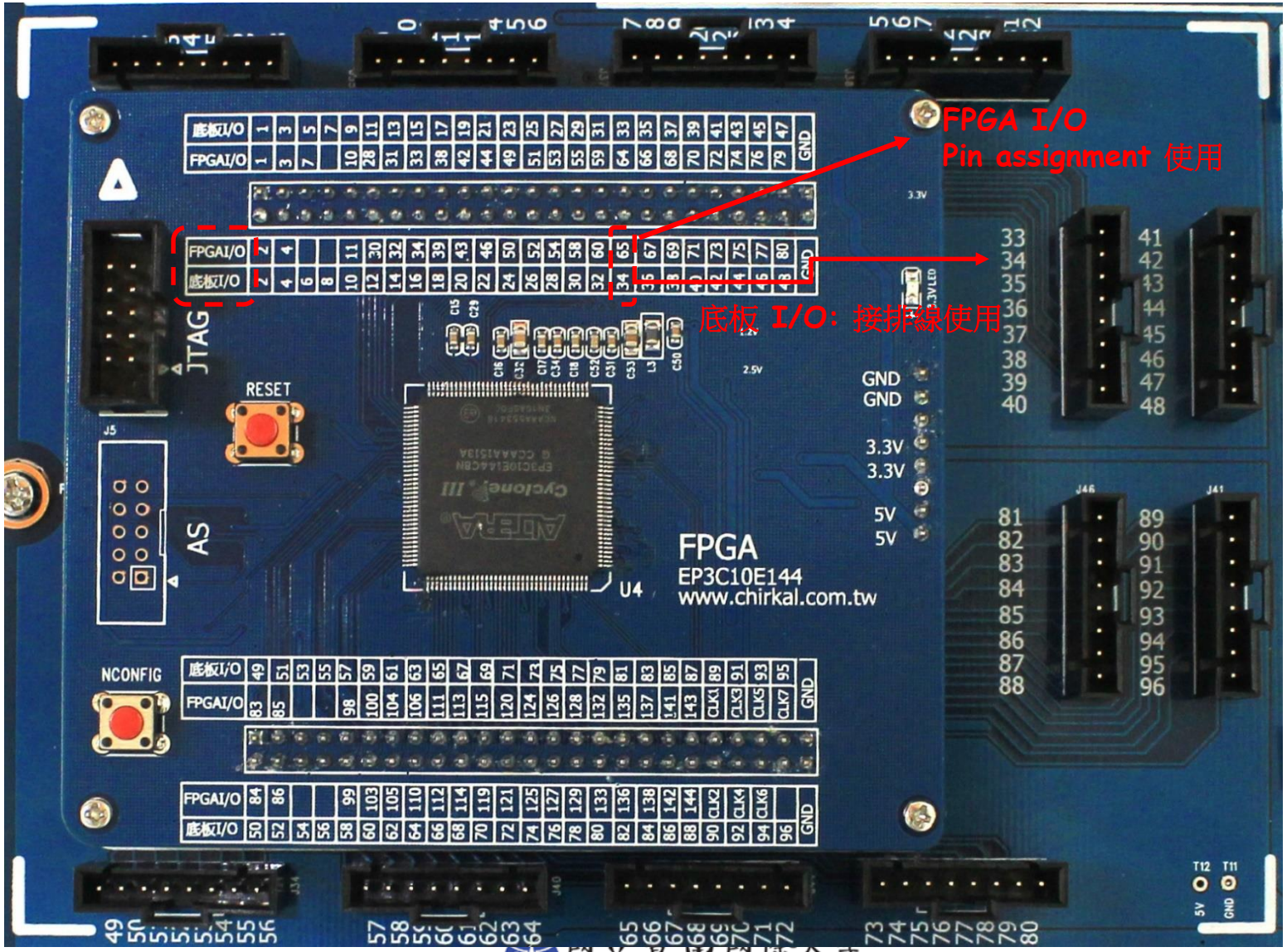
- **Pin assignments (will be different if you use other board pins)**

<<new>> ▾ Filter on node names: *

Statu:	From	To	Assignment Name	Value
✓		in B	Location	PIN_65
✓		out m_out	Location	PIN_72
✓		in select	Location	PIN_66
✓		in A	Location	PIN_64
	<<new>>	<<new>>	<<new>>	



Recall: FPGA pin mapping: 先規劃好 I/O 使用那些 底板 I/O，再找出 FPGA I/O



Outline

- **Behavior modeling**
  - **Example 4-9: 2-to-1-line multiplexer (FPGA)**
  - **Example 4-10: 4-to-1-line multiplexer (FPGA)**
  - **Example 4-11: Test bench for 2x1 MUX (Verilog & Simulation)**
  - **Example 4-13: Test bench for adder**

HDL Example 4-10

- Behavioral description of a 4-to-1-line multiplexer

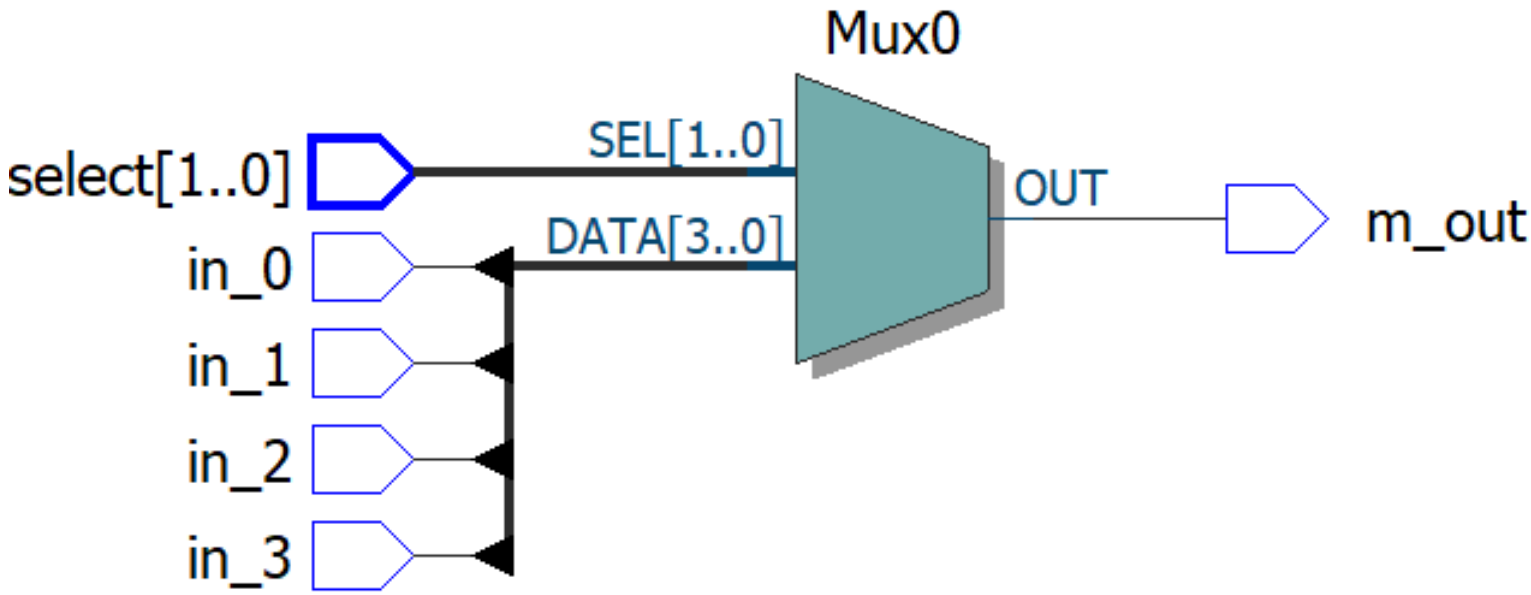
```
// Behavioral description of four-to-one line multiplexer
// Verilog 2001, 2005 port syntax
module mux_4x1_beh
(output reg  m_out,
 input       in_0, in_1, in_2, in_3,
 input [1: 0] select
);
 always @ (in_0, in_1, in_2, in_3, select)      // Verilog 2001, 2005 syntax
  case (select)
    2'b00:          m_out = in_0;
    2'b01:          m_out = in_1;
    2'b10:          m_out = in_2;
    2'b11:          m_out = in_3;
  endcase
endmodule
```

新的參數列寫法

長度 2 的 binary number，值為01

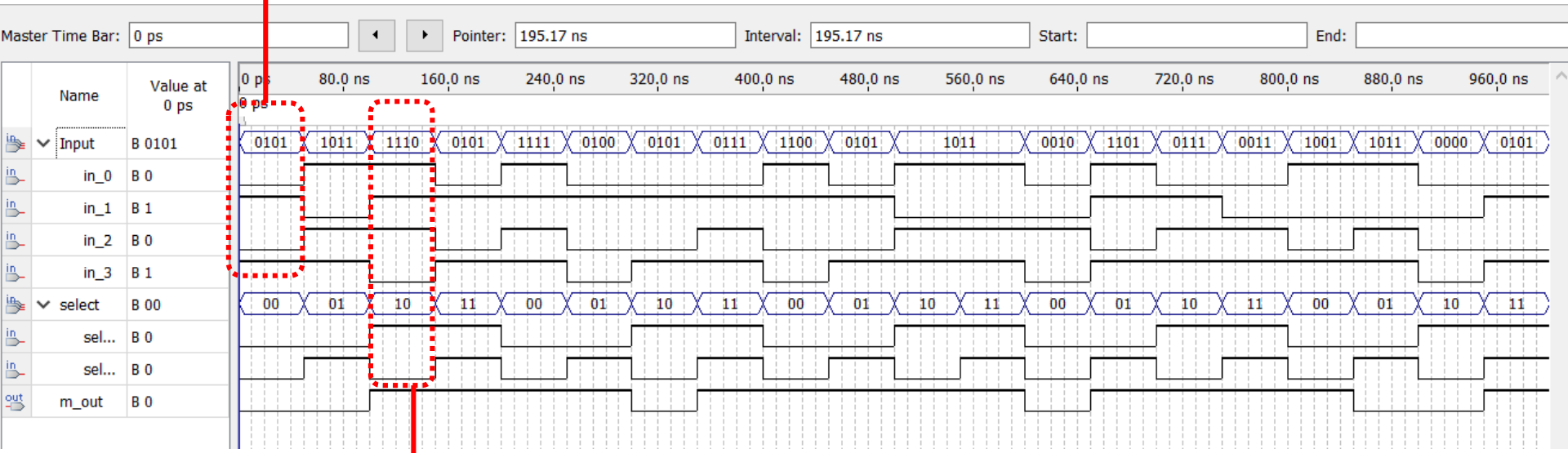
RTL viewer

Tools -> Netlist viewers-> RTL viewer



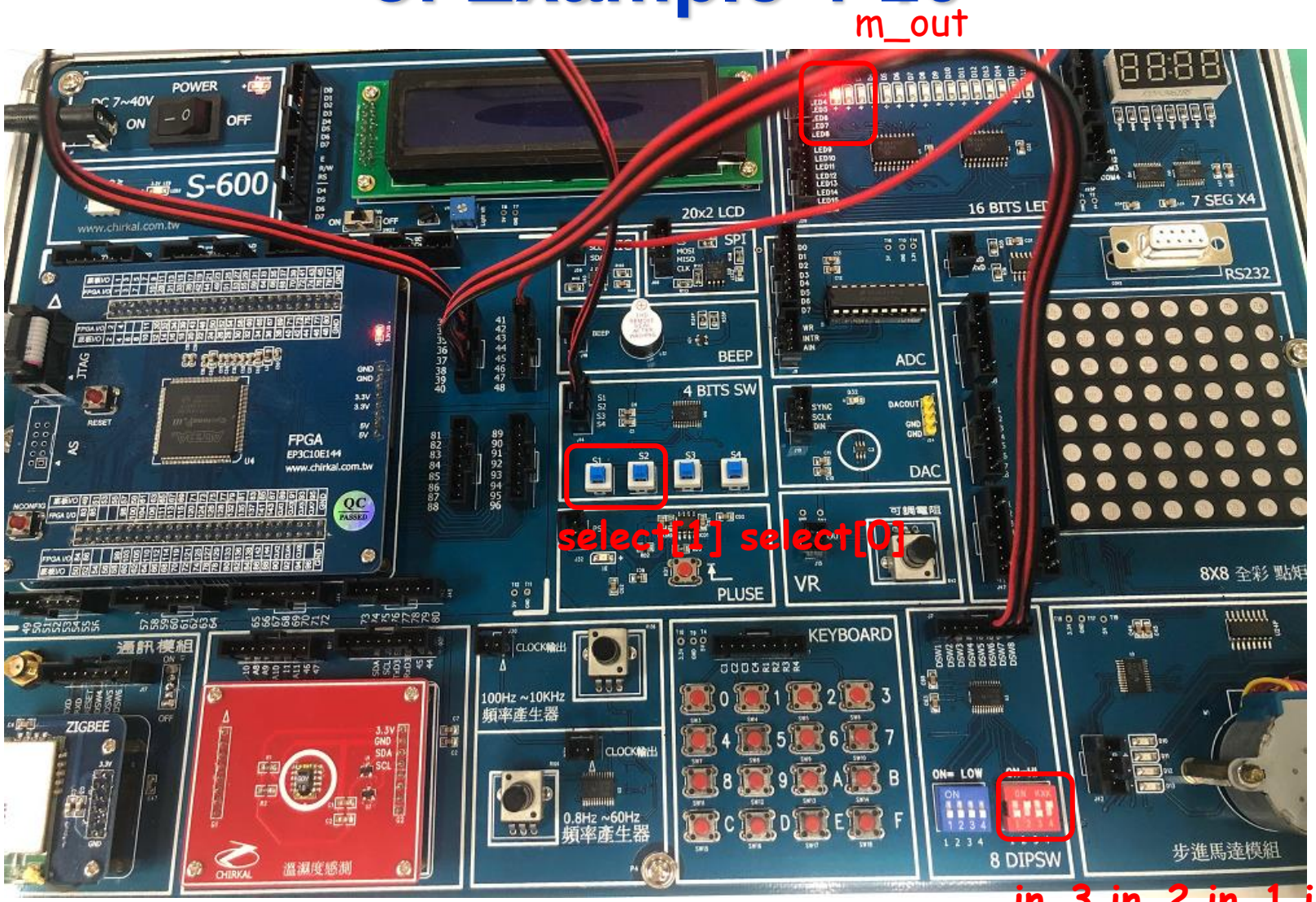
Lab#2-1: Waveform simulation of Example 4-10

隨機產生(R, random)，此 Lab 共有 2^6 種輸入，故隨機產生測試輸入便可



注意將 input 與 select 時間間隔 (interval) 設為一致

Lab2-2: FPGA implementation of Example 4-10



*僅供參考。注意位元順序，要符合左到右，高位元到低位元

Outline

- **Behavior modeling**
  - **Example 4-9: 2-to-1-line multiplexer**
  - **Example 4-10: 4-to-1-line multiplexer**
  - **Example 4-11: Test bench for 2x1 MUX (Verilog & Simulation)**
  - **Example 4-13: Test bench for adder**

Writing a Simple Test Bench

- **initial** block: 只執行一次

```
initial
  begin
          A = 0; B = 0;
    #10   A = 1;
    #20   A = 0; B = 1;
  end
```

Time
0
10
30

```
initial
  begin
    D = 3'b000;
    repeat (7)
      #10  D = D + 3'b001;
  end
```

Three-bit truth table

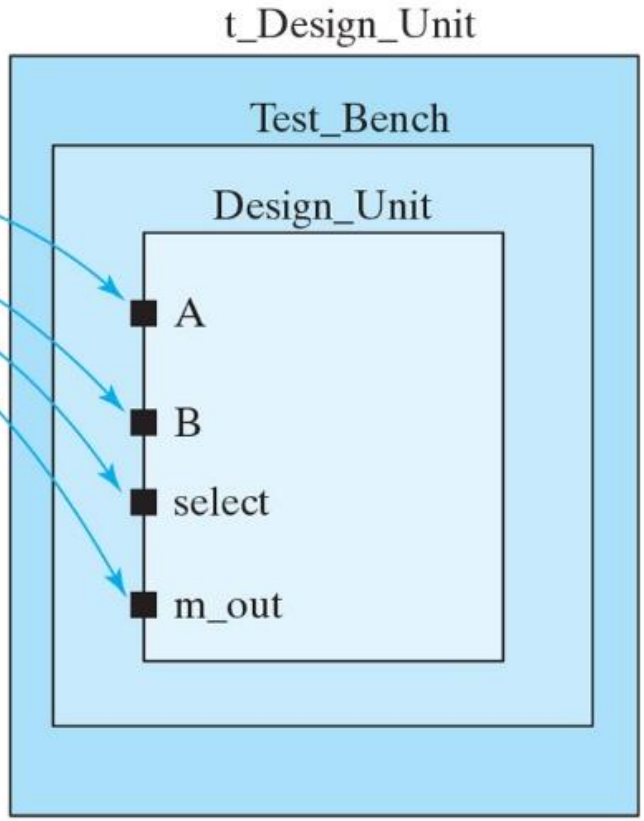
Writing a Simple Test Bench

- Interaction between testbench and design modules

```
modulel t_Design_Unit ();
// Declare stimulus signals
  reg t_A, t_B, t_select;
//Declare reponse signals
  wire : m_out;
--Instantiate unit under test
  Design_Unit UUT
   (.A (t_A), .B(t_B), .select(t_select)t, .m_out(t_m_out));

--Generate stimulus
initial begin
  t_a <= '1';
  t_B <= '0';
end;

initial begin
  select = 1; forever # 10 select = ! select;
end;
end;
endmodule
```



Writing a Simple Test Bench

- Stimulus module

```
module test_module_name;
  // Declare local reg and wire identifiers.
  // Instantiate the design module under test.
  // Specify a stopwatch, using $finish to terminate the simulation.
  // Generate stimulus, using initial and always statements.
  // Display the output response (text or graphics (or both)).
endmodule
```

- System tasks for display

$display—display a one-time value of variables or strings with an end-of-line return,
$write**—same as **$display, but without going to next line,
$monitor—display variables whenever a value changes during a simulation run,
$time—display the simulation time,
$finish—terminate the simulation.

- Syntax for **$display**, **$write**, and **$monitor**:

Task-name (format specification, argument list);

Example:

```
$display ("%d %b %b", C, A, B);
```

Example:

```
$display ("time = %0d A = %b B = %b", $time, A, B);
```

```
time = 3  A = 10  B = 1
```

HDL Example 4-11

- Stimulus module

```
// Test bench with stimulus for mux_2x1_df

module t_mux_2x1_df;
  wire        t_mux_out;
  reg         t_A, t_B;
  reg         t_select;
  parameter   stop_time = 50;
  mux_2x1_df  M1 (t_mux_out, t_A, t_B, t_select);     // Instantiation of circuit to be tested

initial # stop_time $finish;

initial begin                                         // Stimulus generator
        t_select = 1; t_A = 0; t_B = 1;
  #10   t_A = 1; t_B = 0;
  #10   t_select = 0;
  #10   t_A = 0; t_B = 1;
end
```

Test module 通常沒有參數

parameter, 常數的變數

我們這裡改成使用前面寫的 mux_2x1_beh

一個程式裡面可以有多個 initial

HDL Example 4-11 (Continued)

```
initial begin                                          // Response monitor
  // $display ("    time   Select  A    B    m_out");
  // $monitor ($time,, "   %b    %b    %b    %b", t_select, t_A, t_B, t_m_out);
  $monitor ("time=", $time,, "select = %b A = %b B = %b OUT = %b",
  t_select, t_A, t_B, t_mux_out);
end
endmodule
```

課本印刷錯誤, 應為 $time, ", select ...

```
// Dataflow description of two-to-one-line multiplexer

// from Example 4.6
module mux_2x1_df (m_out, A, B, select);
  output        m_out;
  input         A, B;
  input         select;

  assign m_out = (select)? A : B;
endmodule
```

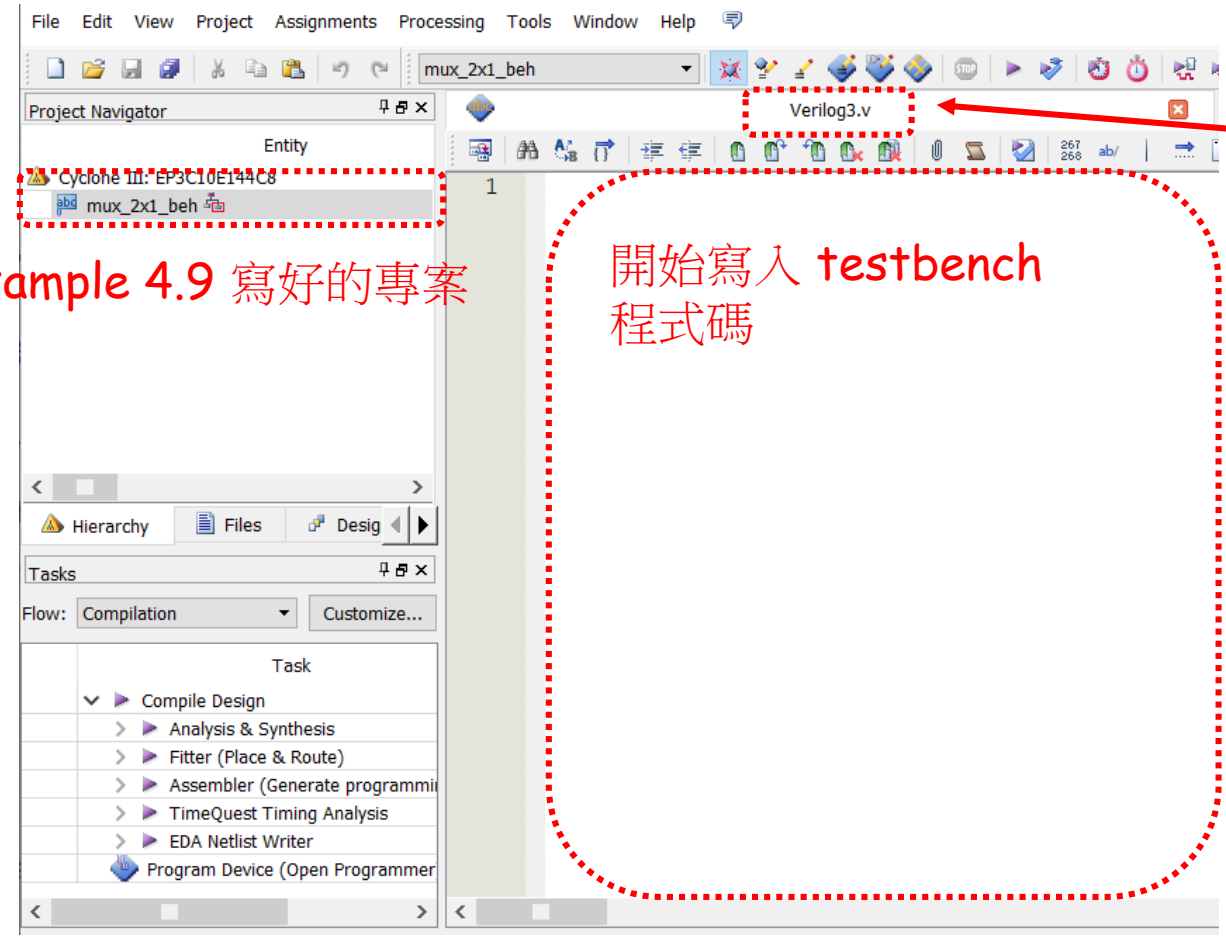
這裡不用寫，我們使用前面寫好的 mux_2x1_beh 專案

```
Simulation log:
select = 1 A = 0 B = 1 OUT = 0 time = 0
select = 1 A = 1 B = 0 OUT = 1 time = 10
select = 0 A = 1 B = 0 OUT = 0 time = 20
select = 0 A = 0 B = 1 OUT = 1 time = 30
```

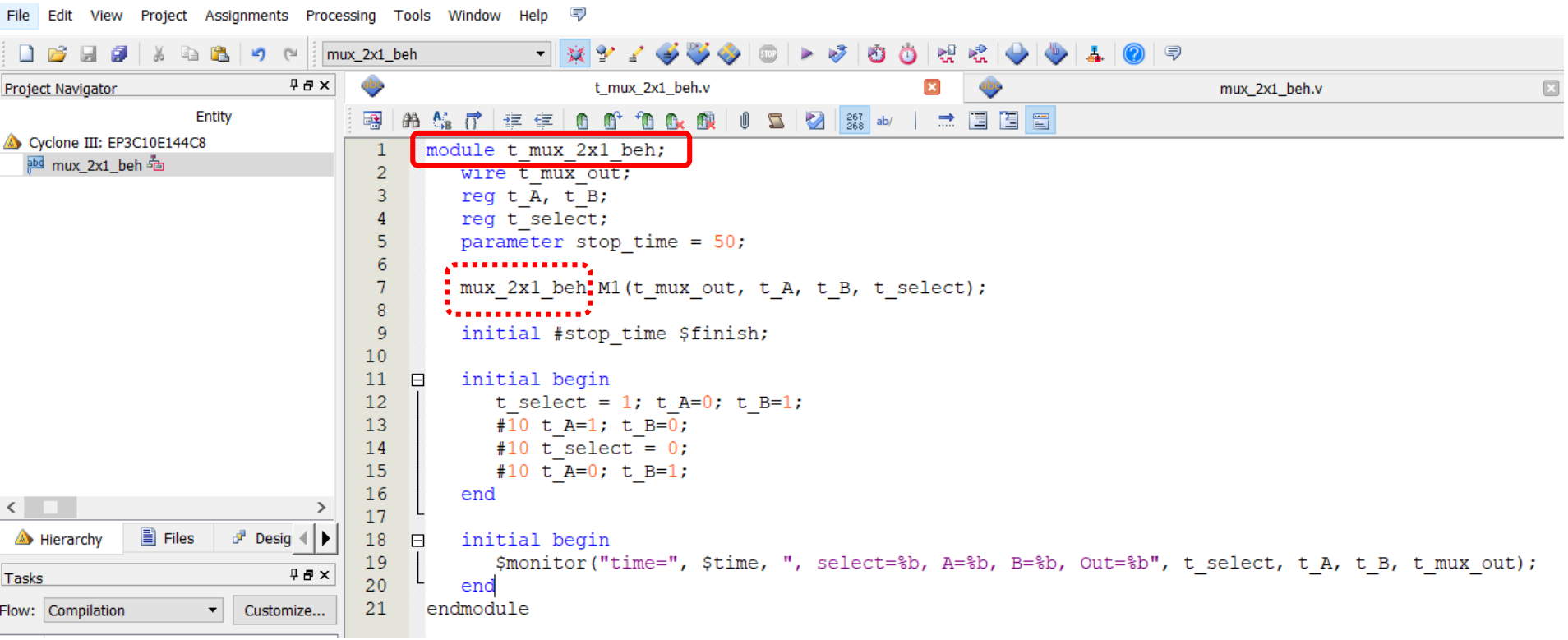
模擬的輸出結果
不是寫在 test bench 裡面的

Write test bench steps

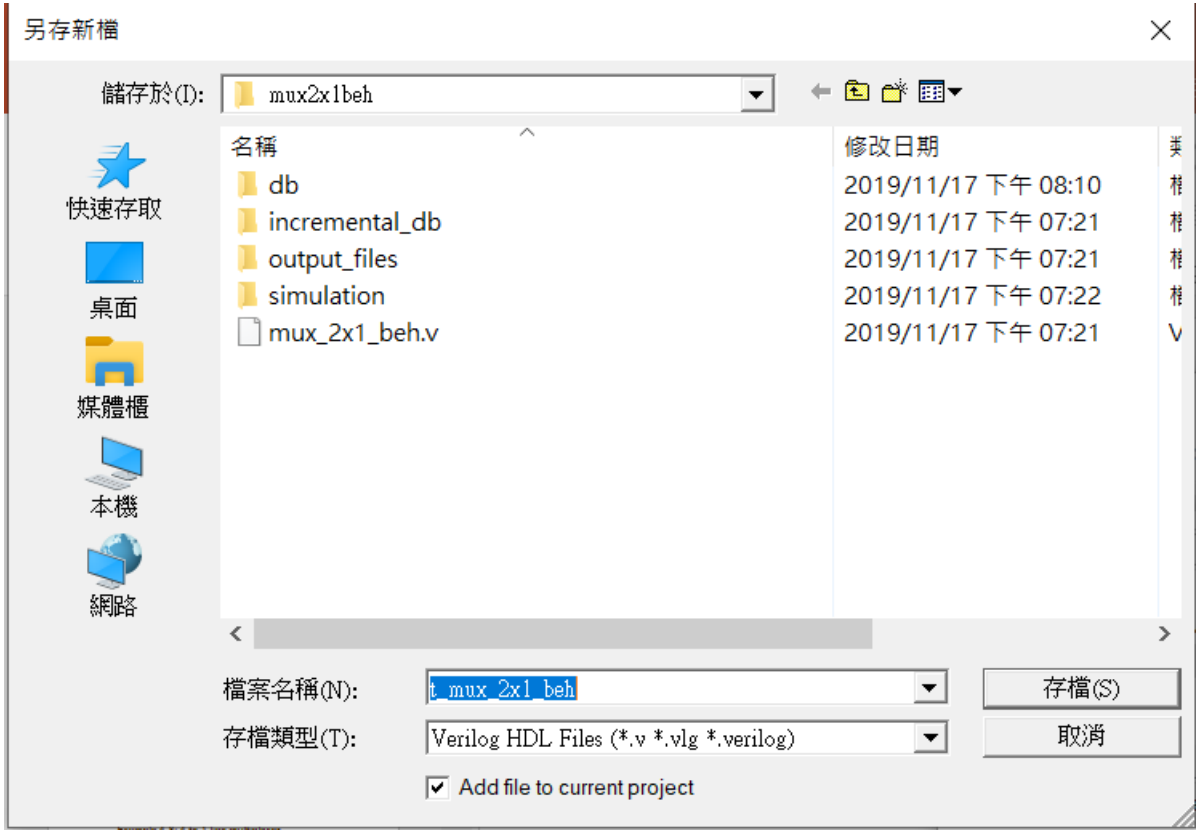
- 開啟剛才在 **Example 4.9** 寫的 **mux_2x1_beh project**, and **new** a **Verilog HDL file**



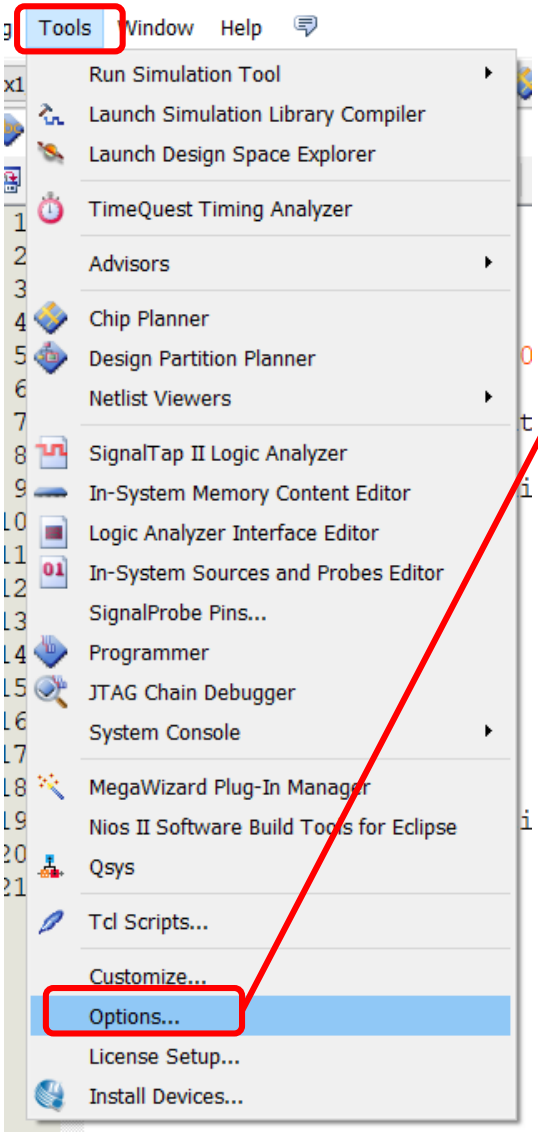
Enter test bench code

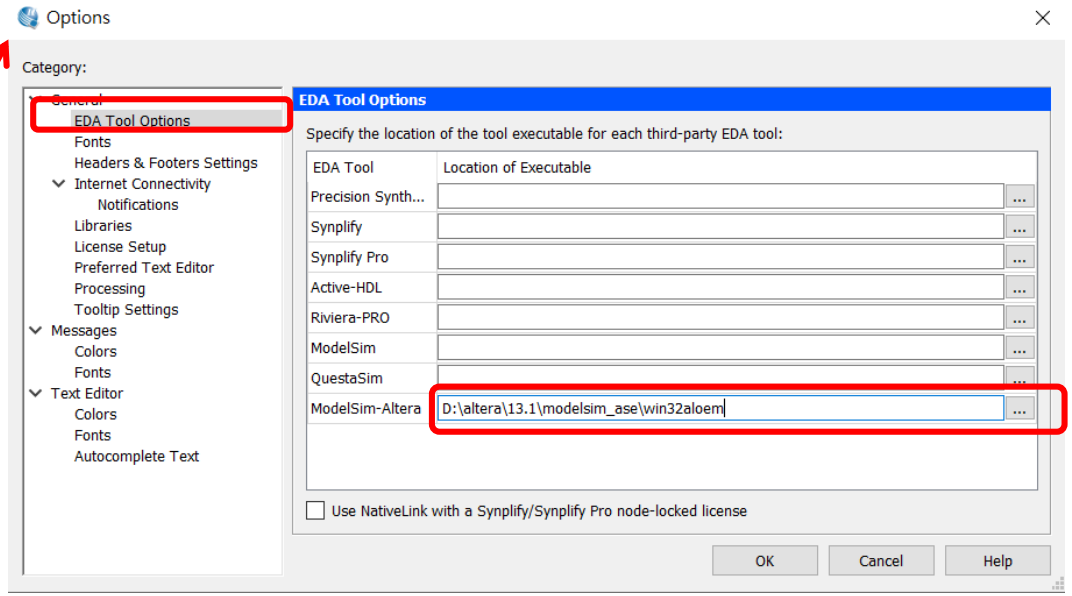


存檔，跟 Verilog module 名稱相同



Setting for simulation (1)





安裝 modelsim_ase 時應該會設定好路徑
如果後面執行時，找不到 modelsim_ase 時再來檢查

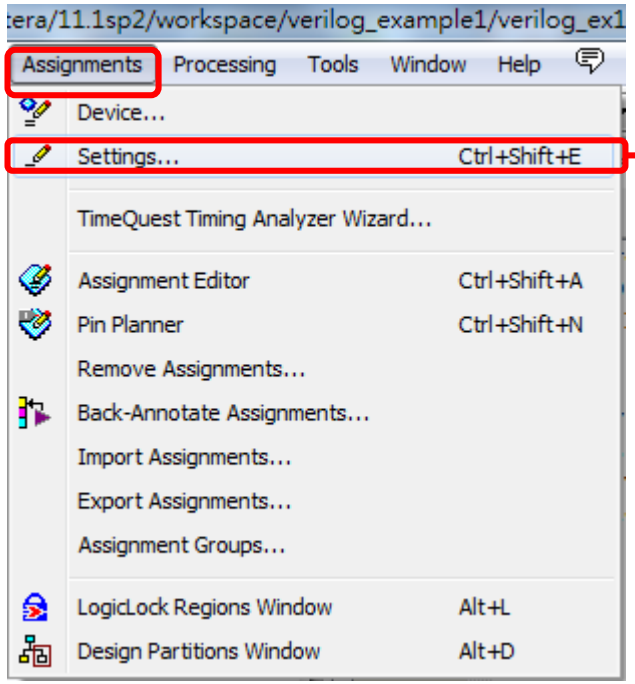
D:\altera\13.1\modelsim_ase\win32aloem

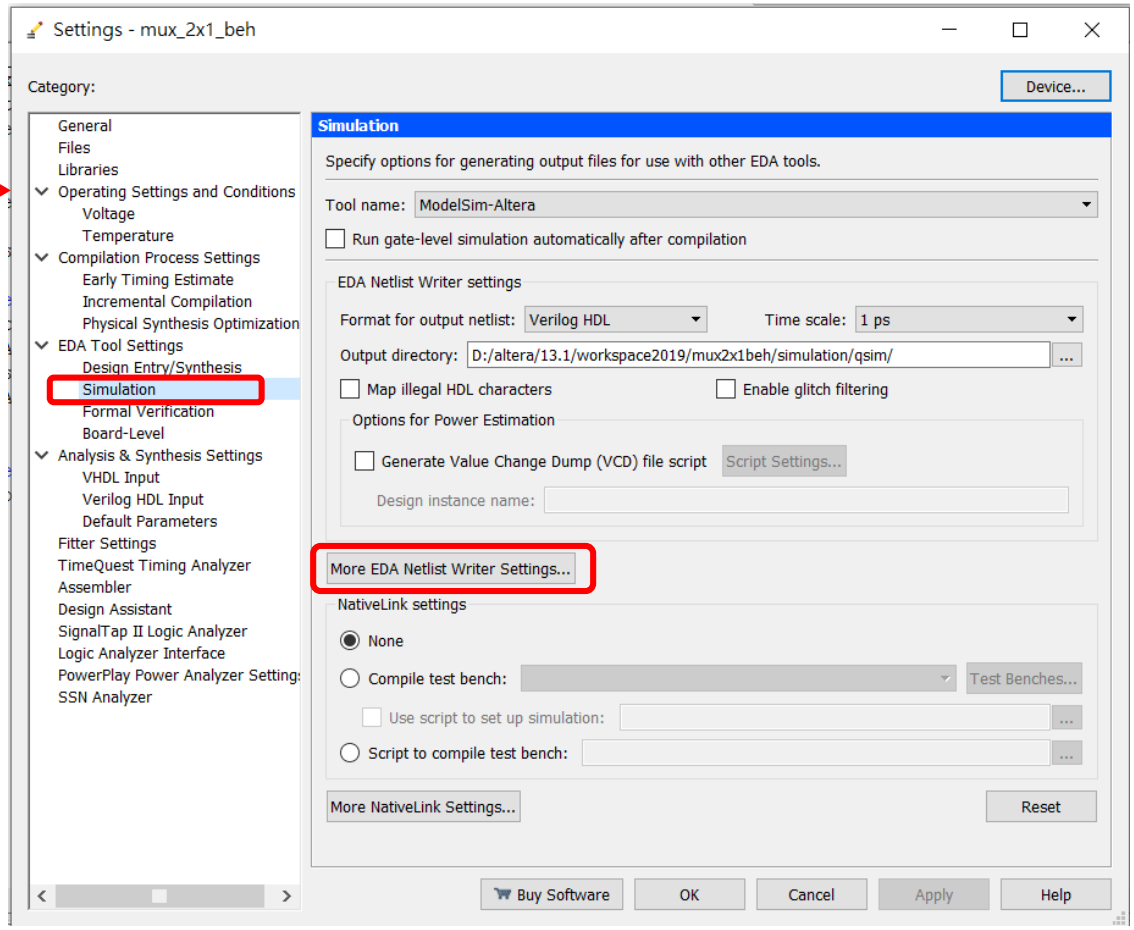
或

D:\altera\13.1\modelsim_ase\win32aloem\

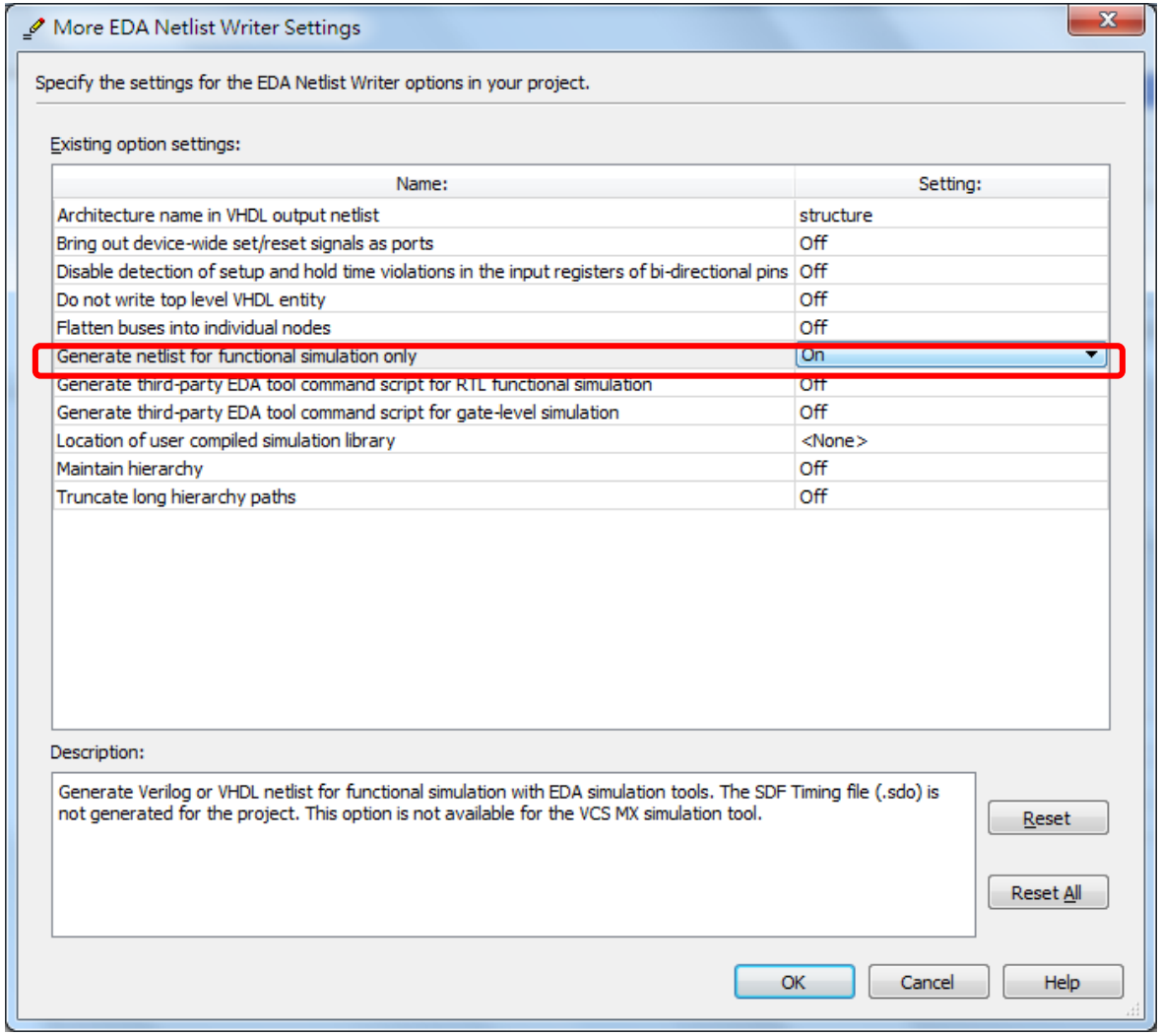
最後的 \, 以前有同學明明路徑有 modelsim, 還是說找不到，這時可以試試看加或不加 \

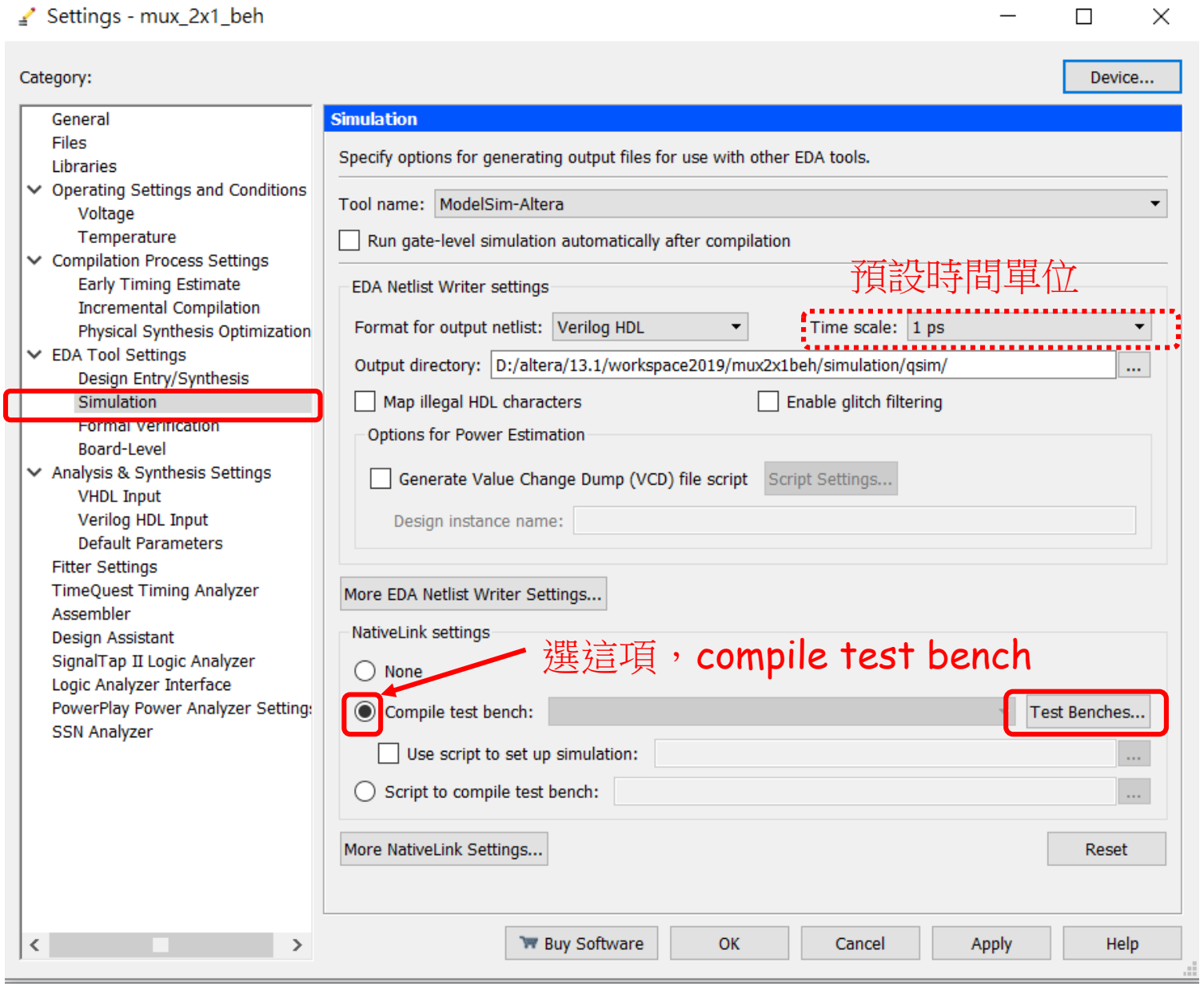
Setting for simulation (2)

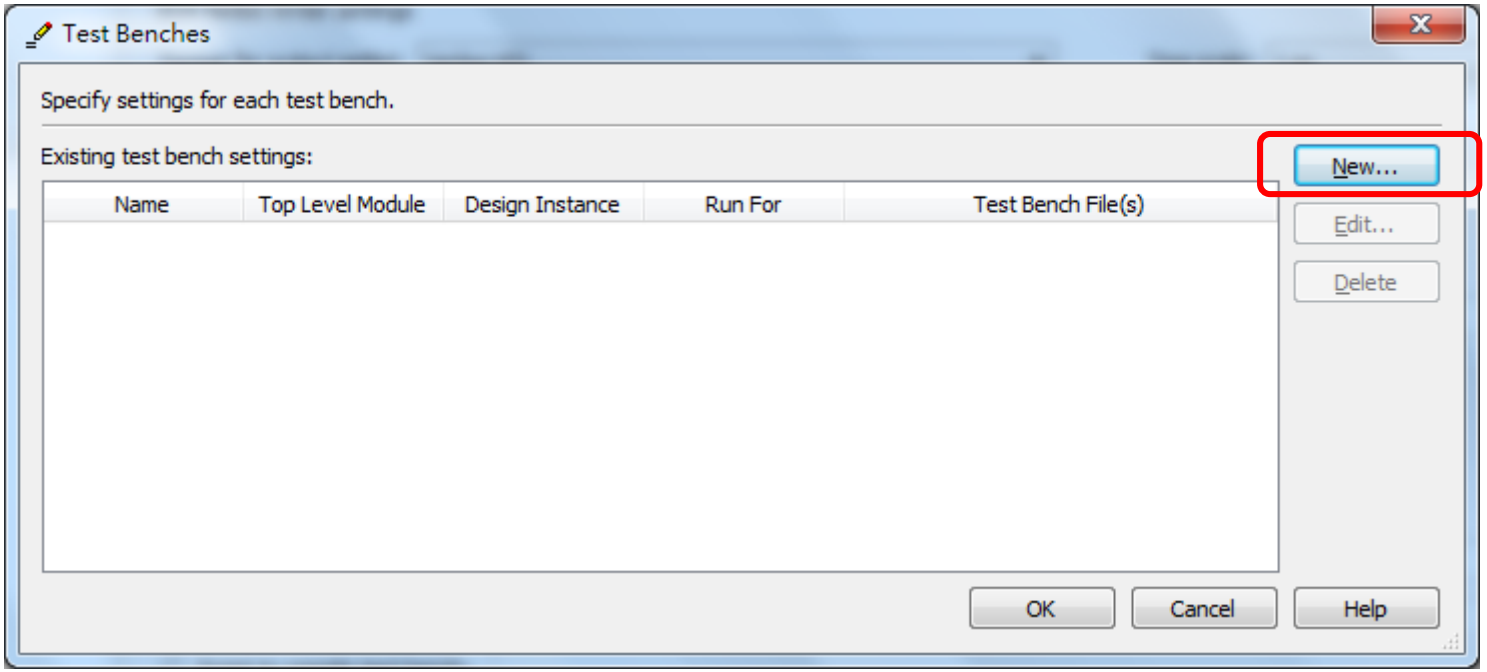


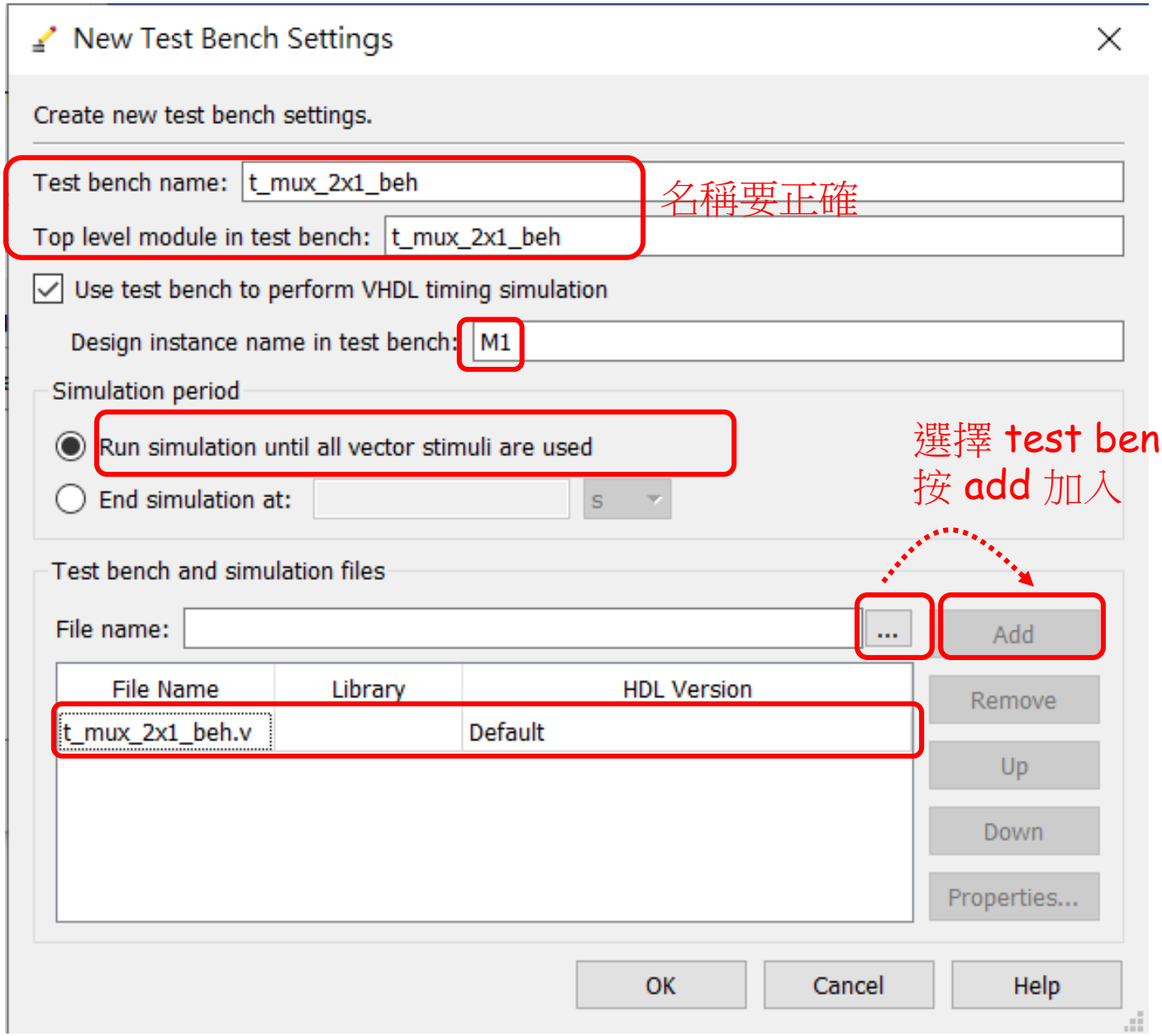


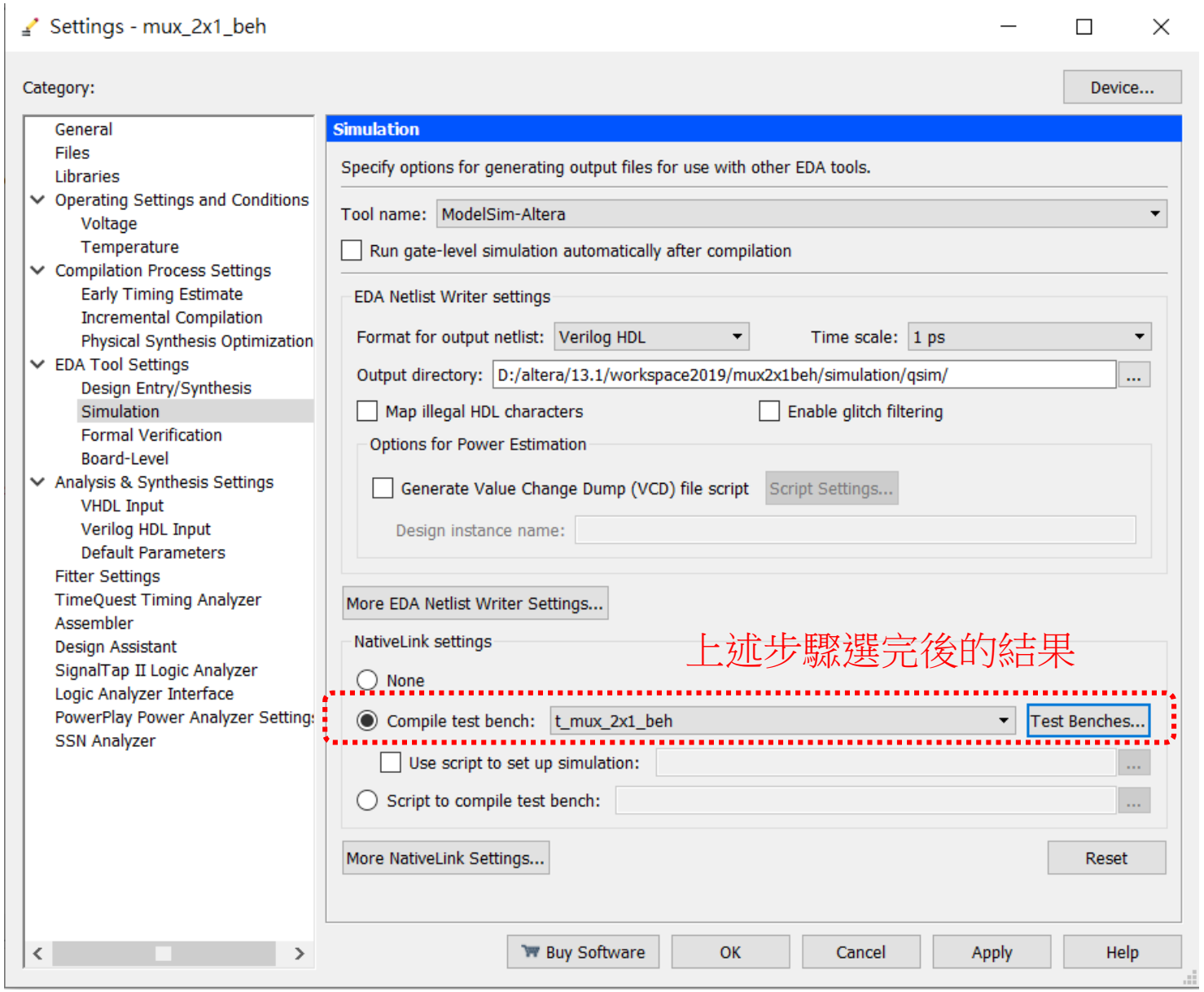
EDA netlist writer setting



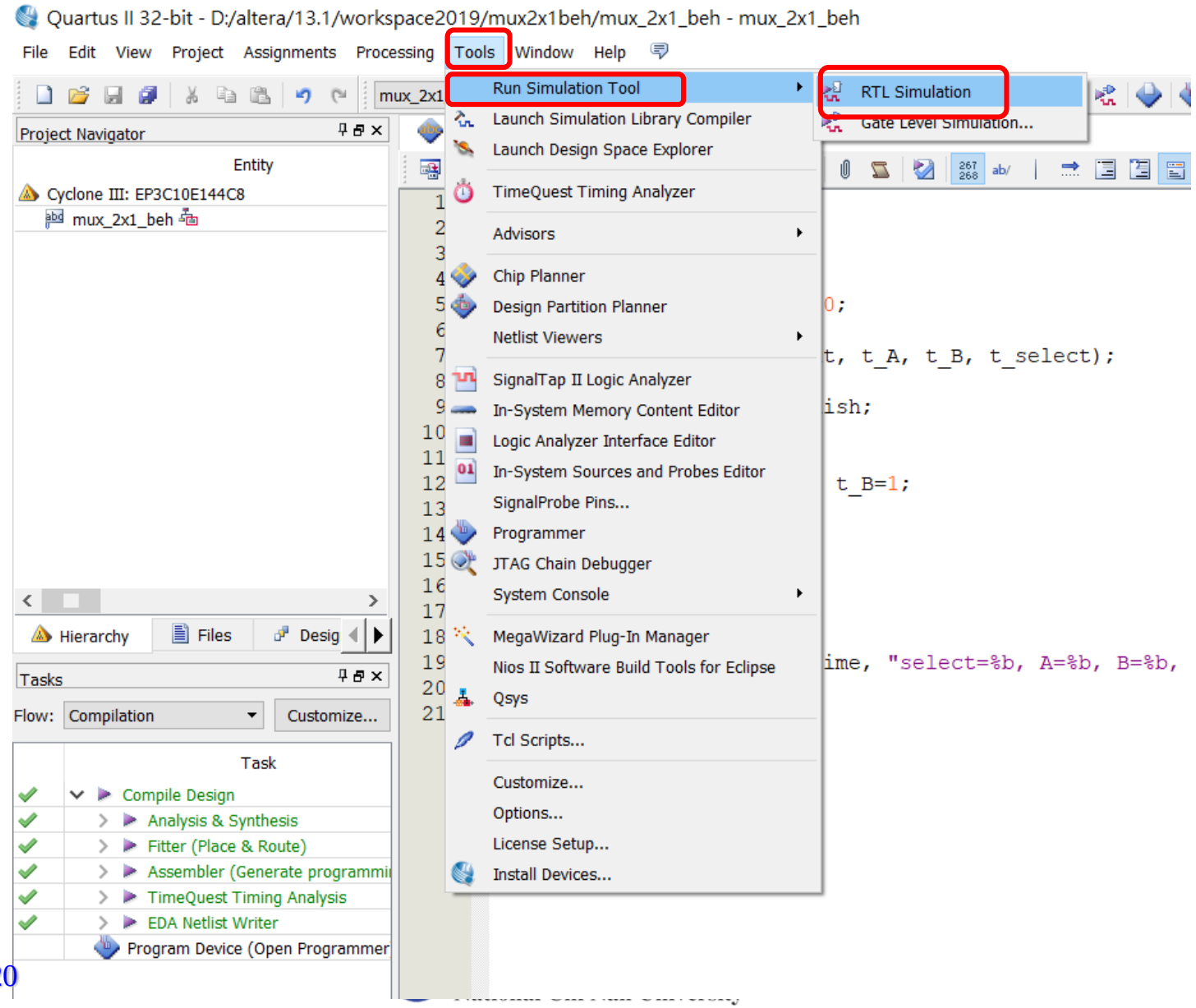

Settings - mux_2x1_beh
Category:
Device...
General
Files
Libraries
Operating Settings and Conditions
Voltage
Temperature
Compilation Process Settings
Early Timing Estimate
Incremental Compilation
Physical Synthesis Optimization
EDA Tool Settings
Design Entry/Synthesis
Simulation
Formal Verification
Board-Level
Analysis & Synthesis Settings
VHDL Input
Verilog HDL Input
Default Parameters
Fitter Settings
TimeQuest Timing Analyzer
Assembler
Design Assistant
SignalTap II Logic Analyzer
Logic Analyzer Interface
PowerPlay Power Analyzer Setting
SSN Analyzer
Simulation
Specify options for generating output files for use with other EDA tools.
Tool name: ModelSim-Altera
Run gate-level simulation automatically after compilation
預設時間單位
EDA Netlist Writer settings
Format for output netlist: Verilog HDL
Time scale: 1 ps
Output directory: D:/altera/13.1/workspace2019/mux2x1beh/simulation/qsim/
Map illegal HDL characters
Enable glitch filtering
Options for Power Estimation
Generate Value Change Dump (VCD) file script
Script Settings...
Design instance name:
More EDA Netlist Writer Settings...
NativeLink settings
選這項，compile test bench
None
Compile test bench:
Test Benches...
Use script to set up simulation:
Script to compile test bench:
More NativeLink Settings...
Reset
Buy Software
OK
Cancel
Apply
Help


Test Benches
Specify settings for each test bench.
Existing test bench settings:
New...
Edit...
Delete
Name
Top Level Module
Design Instance
Run For
Test Bench File(s)
OK
Cancel
Help

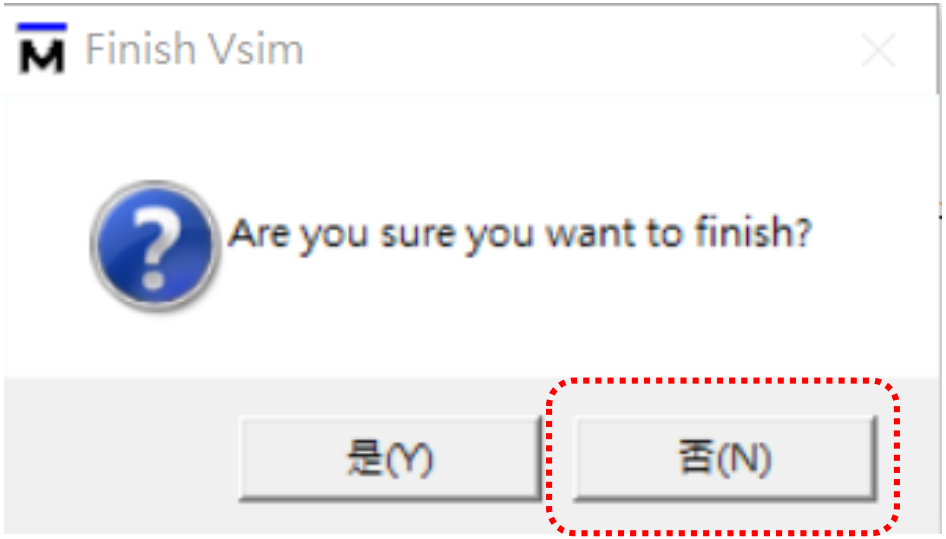

New Test Bench Settings
Create new test bench settings.
Test bench name: t_mux_2x1_beh
Top level module in test bench: t_mux_2x1_beh
名稱要正確
Use test bench to perform VHDL timing simulation
Design instance name in test bench: M1
Simulation period
Run simulation until all vector stimuli are used
End simulation at:
s
選擇 test bench 檔案後
按 add 加入
Test bench and simulation files
File name:
...
Add
File Name
Library
HDL Version
t_mux_2x1_beh.v
Default
Remove
Up
Down
Properties...
OK
Cancel
Help


Settings - mux_2x1_beh
Category:
Device...
General
Files
Libraries
Operating Settings and Conditions
Voltage
Temperature
Compilation Process Settings
Early Timing Estimate
Incremental Compilation
Physical Synthesis Optimization
EDA Tool Settings
Design Entry/Synthesis
Simulation
Formal Verification
Board-Level
Analysis & Synthesis Settings
VHDL Input
Verilog HDL Input
Default Parameters
Fitter Settings
TimeQuest Timing Analyzer
Assembler
Design Assistant
SignalTap II Logic Analyzer
Logic Analyzer Interface
PowerPlay Power Analyzer Settings
SSN Analyzer
Simulation
Specify options for generating output files for use with other EDA tools.
Tool name: ModelSim-Altera
Run gate-level simulation automatically after compilation
EDA Netlist Writer settings
Format for output netlist: Verilog HDL
Time scale: 1 ps
Output directory: D:/altera/13.1/workspace2019/mux2x1beh/simulation/qsim/
Map illegal HDL characters
Enable glitch filtering
Options for Power Estimation
Generate Value Change Dump (VCD) file script
Script Settings...
Design instance name:
More EDA Netlist Writer Settings...
NativeLink settings
上述步驟選完後的結果
None
Compile test bench: t_mux_2x1_beh
Test Benches...
Use script to set up simulation:
Script to compile test bench:
More NativeLink Settings...
Reset
Buy Software
OK
Cancel
Apply
Help

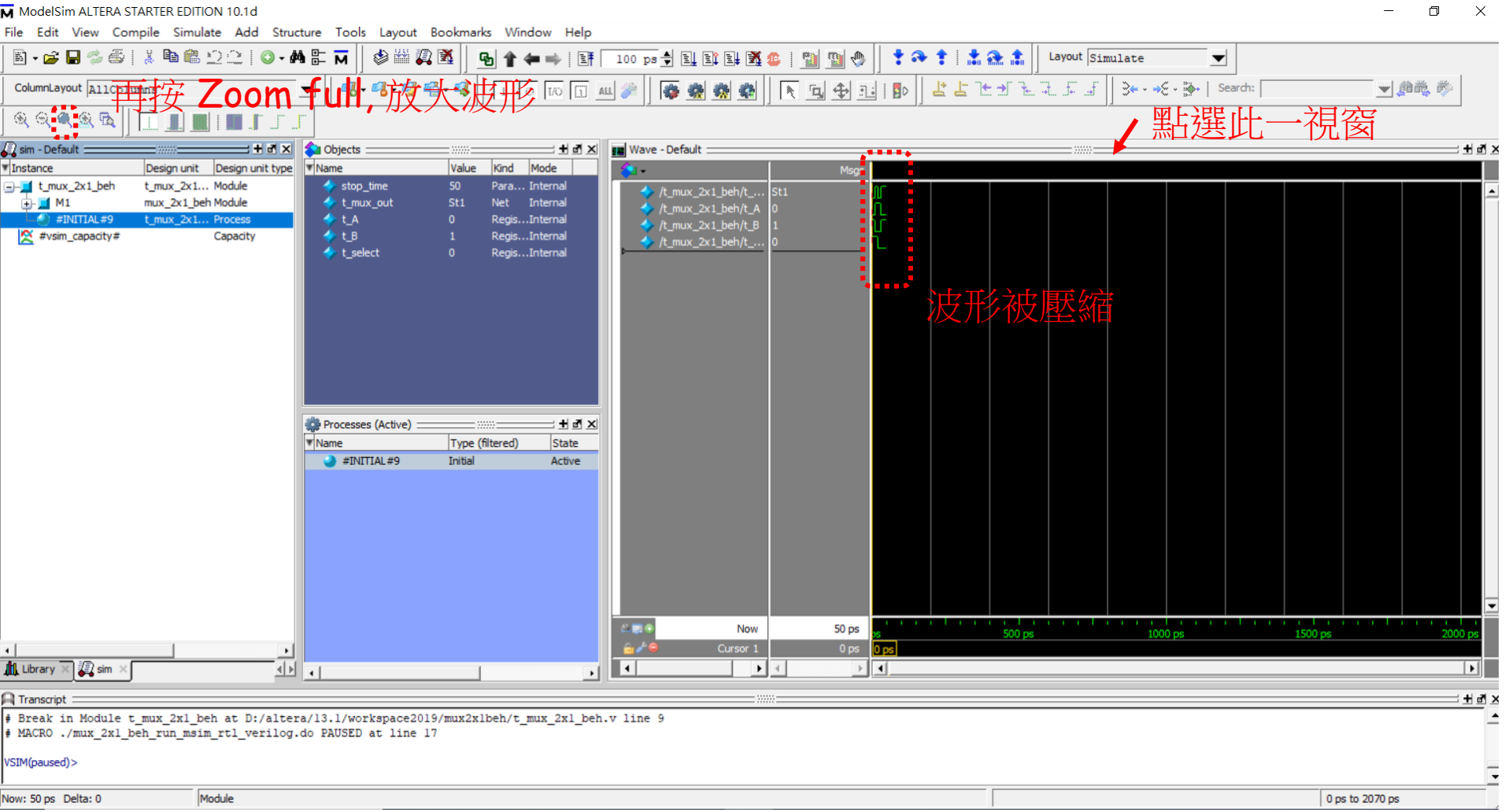
Run RTL simulation



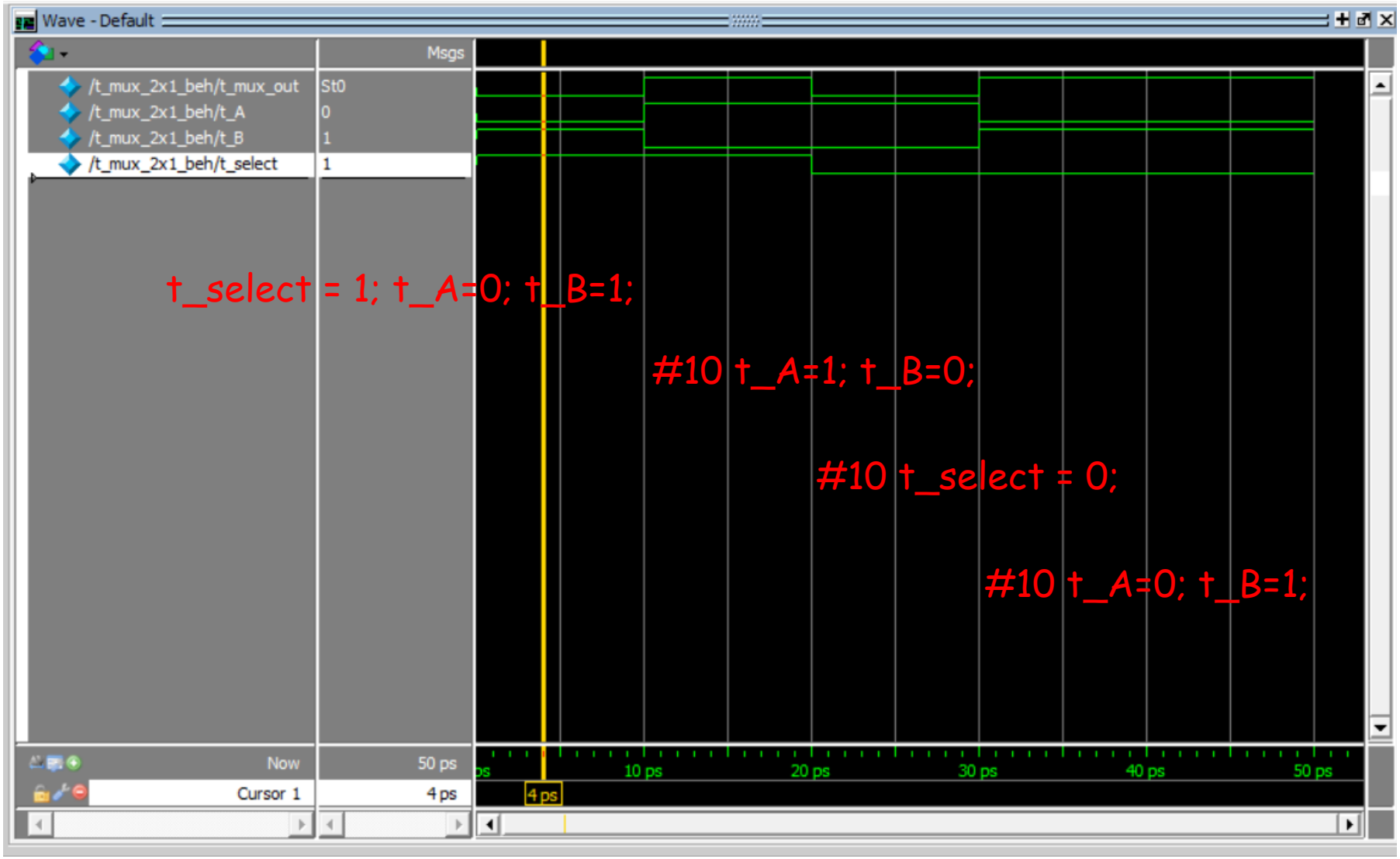
不要終止 Simulator



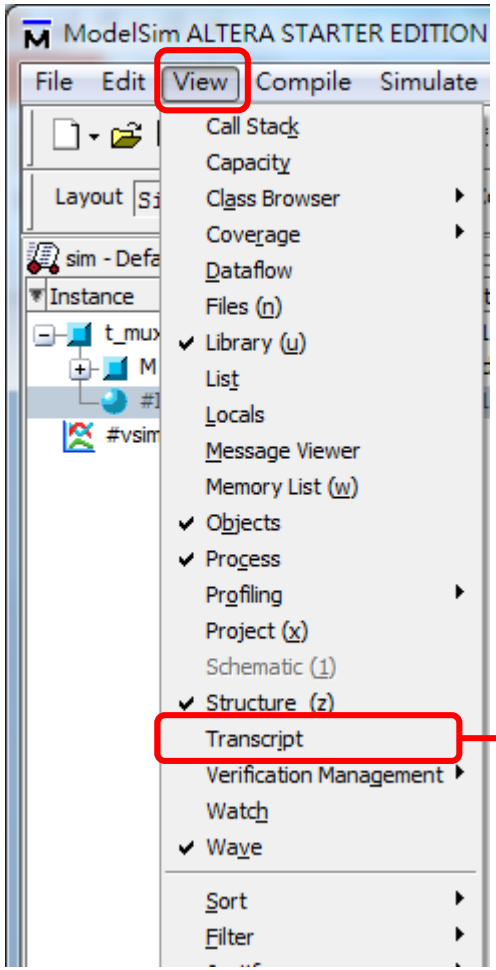
ModelSim Altera 執行結果



Wave window


Wave - Default
Msgs
/t_mux_2x1_beh/t_mux_out
/t_mux_2x1_beh/t_A
/t_mux_2x1_beh/t_B
/t_mux_2x1_beh/t_select
St0
0
1
1
t_select = 1; t_A=0; t_B=1;
#10 t_A=1; t_B=0;
#10 t_select = 0;
#10 t_A=0; t_B=1;
Now
50 ps
Cursor 1
4 ps
10 ps
20 ps
30 ps
40 ps
50 ps
4 ps

Transcript window



Transcript

```
# -- Compiling module t_mux_2x1_beh
#
# Top level modules:
#       t_mux_2x1_beh
#
# vsim -t 1ps -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -L cycloneiii_ver -L
# vsim -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -L cycloneiii_ver -L rtl_wor
# Loading work.t_mux_2x1_beh
# Loading work.mux_2x1_beh
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# time=                    0, select=1, A=0, B=1, Out=0
# time=                   10, select=1, A=1, B=0, Out=1
# time=                   20, select=0, A=1, B=0, Out=0
# time=                   30, select=0, A=0, B=1, Out=1
# ** Note: $finish    : D:/altera/13.1/workspace2019/mux2x1beh/t_mux_2x1_beh.v(9)
#    Time: 50 ps  Iteration: 0  Instance: /t_mux_2x1_beh
# 1
# Break in Module t_mux_2x1_beh at D:/altera/13.1/workspace2019/mux2x1beh/t_mux_2x1_beh.v line 9
# Simulation Breakpoint: 1
# Break in Module t_mux_2x1_beh at D:/altera/13.1/workspace2019/mux2x1beh/t_mux_2x1_beh.v line 9
# MACRO ./mux_2x1_beh_run_msim_rtl_verilog.do PAUSED at line 17
```

$monitor 輸出結果

Lab#3

- **Write test bench for Lab#2 (Example 4-10)**

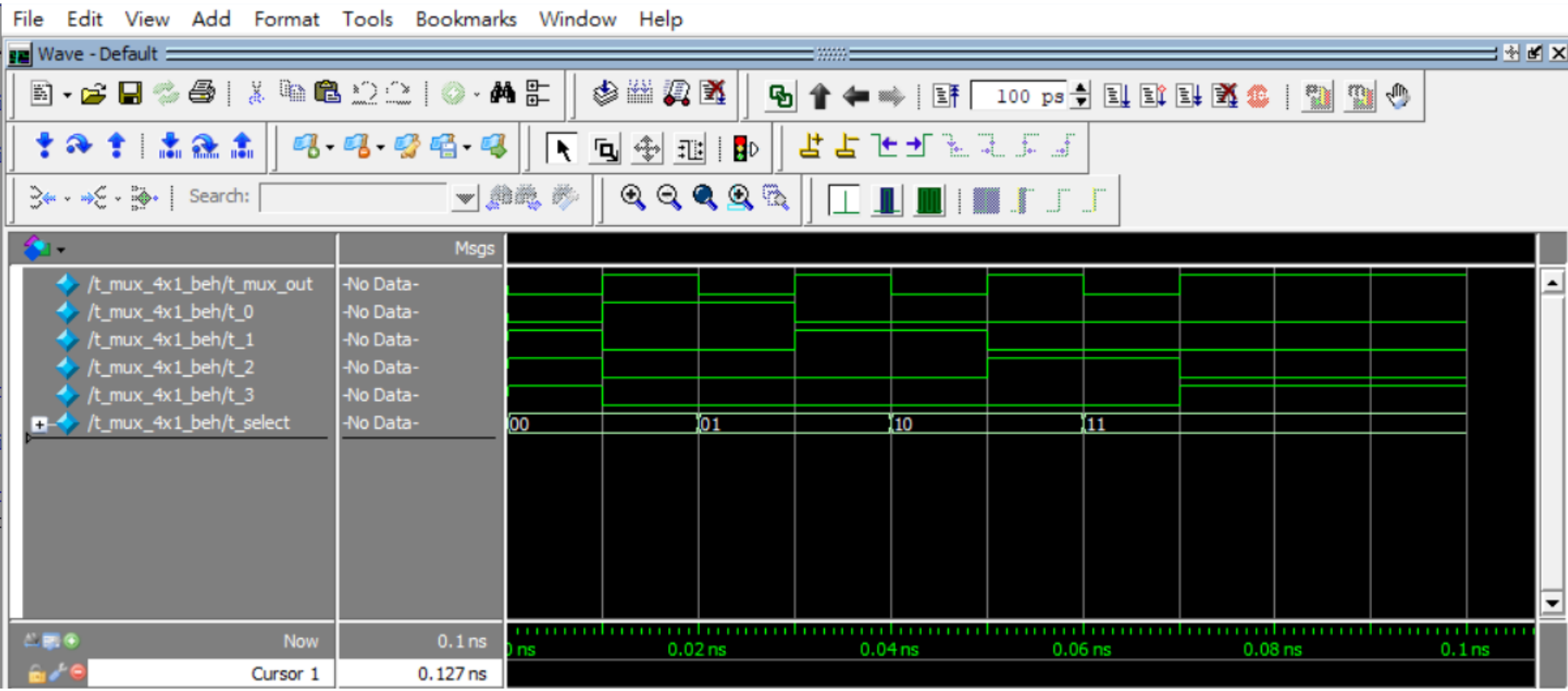
```
1   module t_mux_4x1_beh;
2      wire t_mux_out;
3      reg t_0, t_1, t_2, t_3;
4      reg [1:0] t_select;
5      parameter stop_time = 100;
6
7      mux_4x1_beh M1(t_mux_out, t_0, t_1, t_2, t_3, t_select);
8
9      initial #stop_time $finish;
10
```

Generate test signal as the following slides

Use $monitor to generate output to transcript

endmodule

輸出結果: Wave windows



輸出結果: Transcript window

Transcript

```
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2012
# -- Compiling module t_mux_4x1_beh
#
# Top level modules:
#       t_mux_4x1_beh
#
# vsim -t 1ps -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -
# vsim -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -L cyclo
# Loading work.t_mux_4x1_beh
# Loading work.mux_4x1_beh
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# time=                    0, select=00, t0=0, t1=1, t2=1, t3=1, Out=0
# time=                   10, select=00, t0=1, t1=0, t2=0, t3=0, Out=1
# time=                   20, select=01, t0=1, t1=0, t2=0, t3=0, Out=0
# time=                   30, select=01, t0=0, t1=1, t2=0, t3=0, Out=1
# time=                   40, select=10, t0=0, t1=1, t2=0, t3=0, Out=0
# time=                   50, select=10, t0=0, t1=0, t2=1, t3=0, Out=1
# time=                   60, select=11, t0=0, t1=0, t2=1, t3=0, Out=0
# time=                   70, select=11, t0=0, t1=0, t2=0, t3=1, Out=1
# ** Note: $finish    : D:/altera/13.1/workspace2019/mux4x1beh/t_mux_4x1_beh.v(9)
#    Time: 100 ps  Iteration: 0  Instance: /t_mux_4x1_beh
```

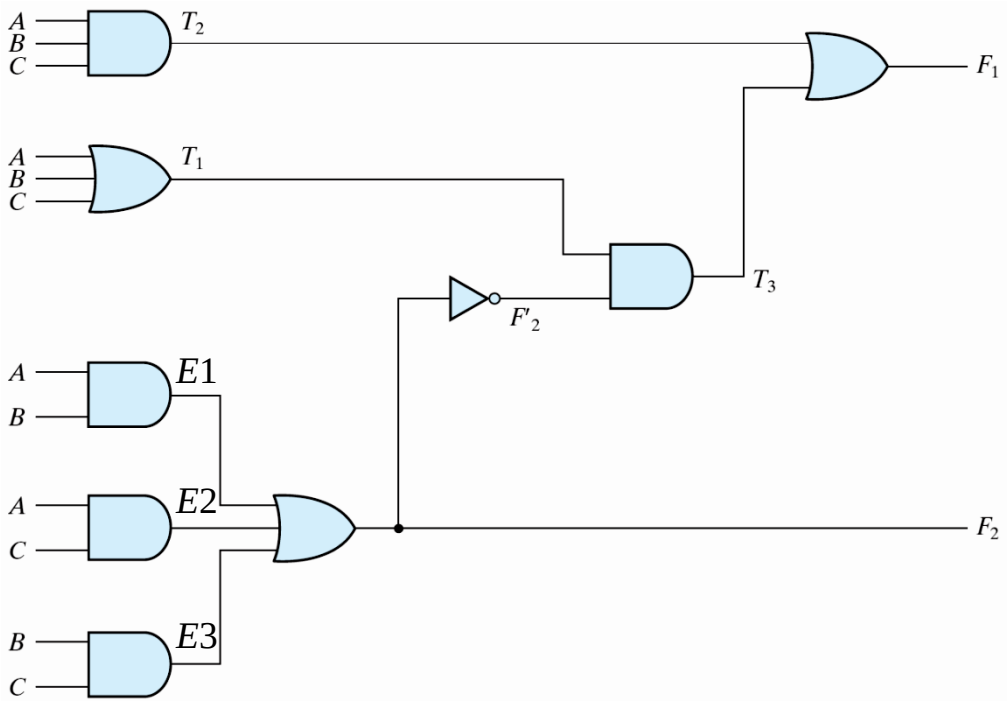
Outline

- **Behavior modeling**
  - **Example 4-9: 2-to-1-line multiplexer**
  - **Example 4-10: 4-to-1-line multiplexer**
  - **Example 4-11: Test bench for 2x1 MUX**
  - **Example 4-13: Test bench for adder**

HDL Example 4-13

- Gate-level description of a full adder

```
// Gate-level description of circuit of Fig. 4.2
module Circuit_of_Fig_4_2 (A, B, C, F1, F2);
  input   A, B, C;
  output  F1, F2;
  wire    T1, T2, T3, F2_b, E1, E2, E3;
  or    g1 (T1, A, B, C);
  and g2 (T2, A, B, C);
  and g3 (E1, A, B);
  and g4 (E2, A, C);
  and g5 (E3, B, C);
  or    g6 (F2, E1, E2, E3);
  not   g7 (F2_b, F2);
  and g8 (T3, T1, F2_b);
  or    g9 (F1, T2, T3);
endmodule
```

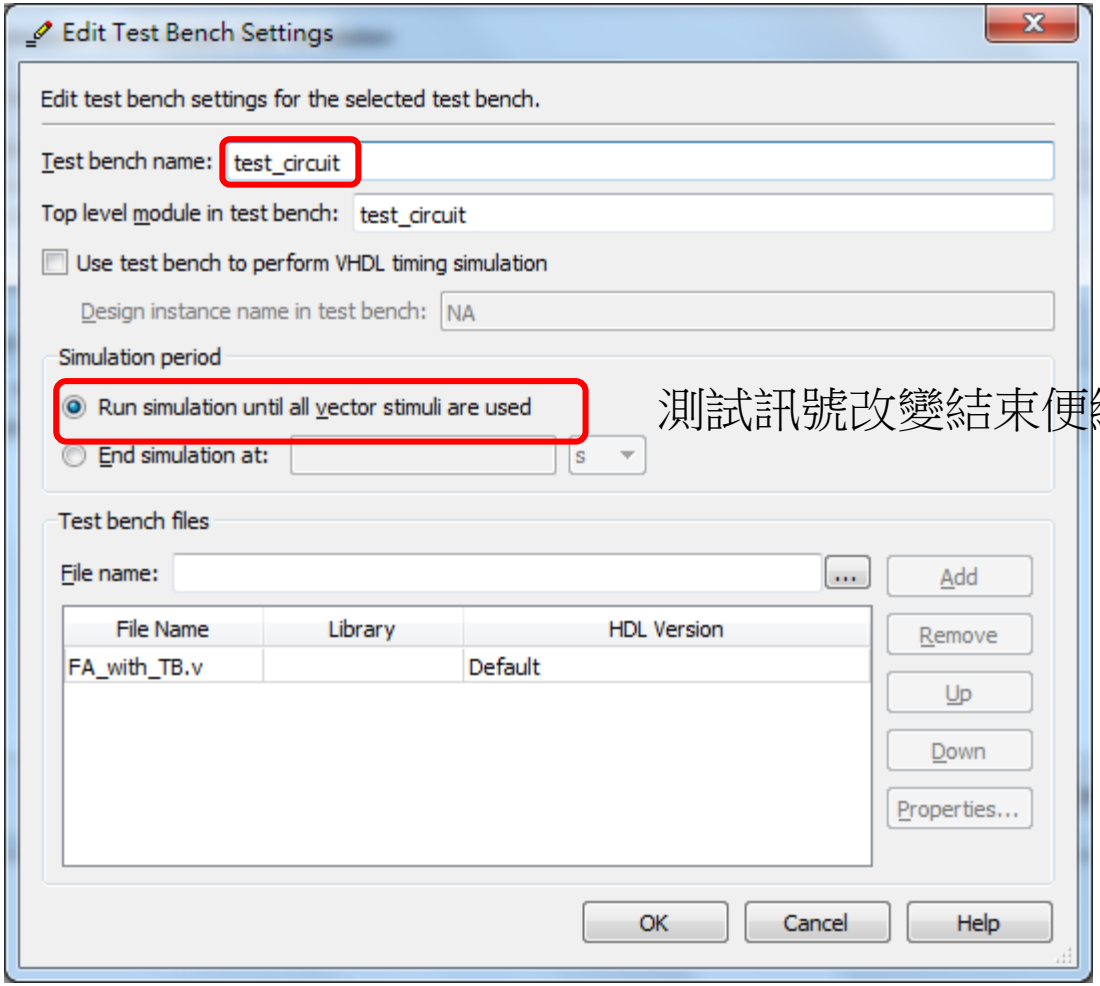


HDL Example 4-10 (Continued)

```
// Stimulus to analyze the circuit
module test_circuit;
  reg [2: 0] D;
  wire F1, F2;
  Circuit_of_Fig_4_2   M_F4_32 (D[2], D[1], D[0], F1, F2);
  initial
    begin
      D = 3'b000;
      repeat (7) #10 D = D + 1'b1;
    end
  initial
  $monitor ("ABC = %b F1 = %b F2 =%b ",  D, F1, F2);
endmodule
```

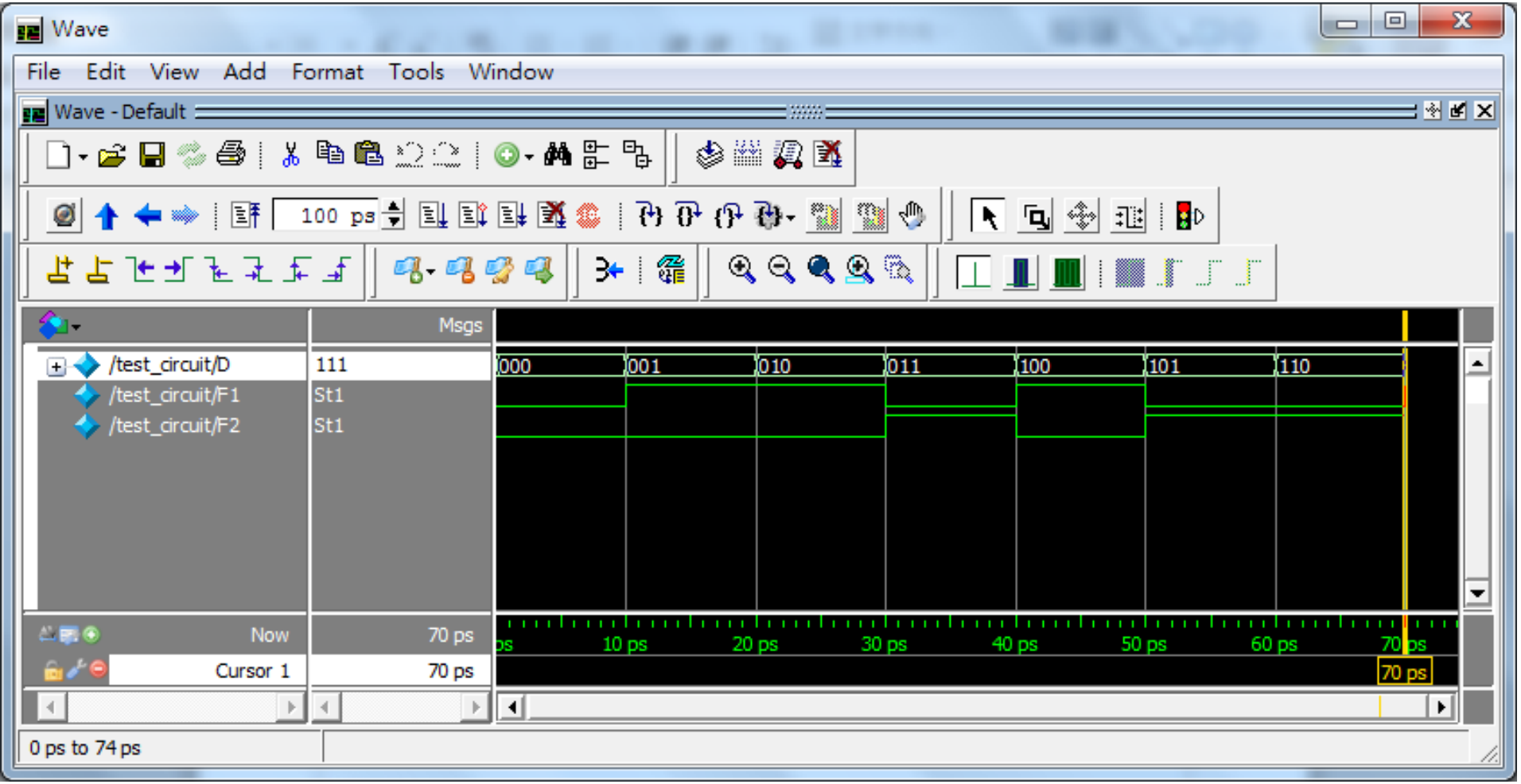
Simulation log: ABC = 000 F1 = 0 F2 =0
ABC = 001 F1 = 1 F2 =0 ABC = 010 F1 = 1 F2 =0
ABC = 011 F1 = 0 F2 =1 ABC = 100 F1 = 1 F2 =0
ABC = 101 F1 = 0 F2 =1 ABC = 110 F1 = 0 F2 =1
ABC = 111 F1 = 1 F2 =1

Assignment -> setting -> simulation -> test bench



測試訊號改變結束便終止

RTL simulation => waveform



RTL simulation => transcript