



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

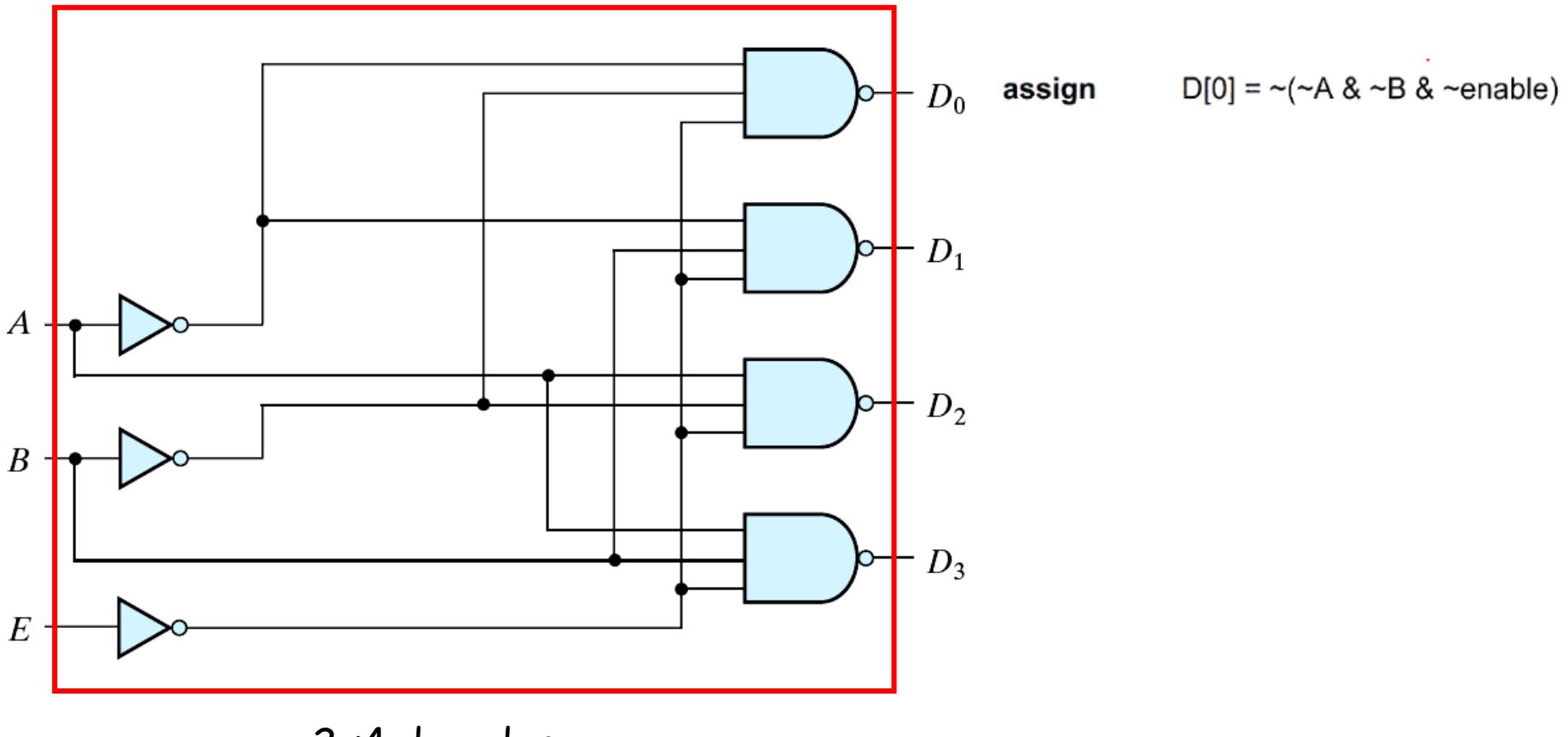


Modeling Styles

- **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?

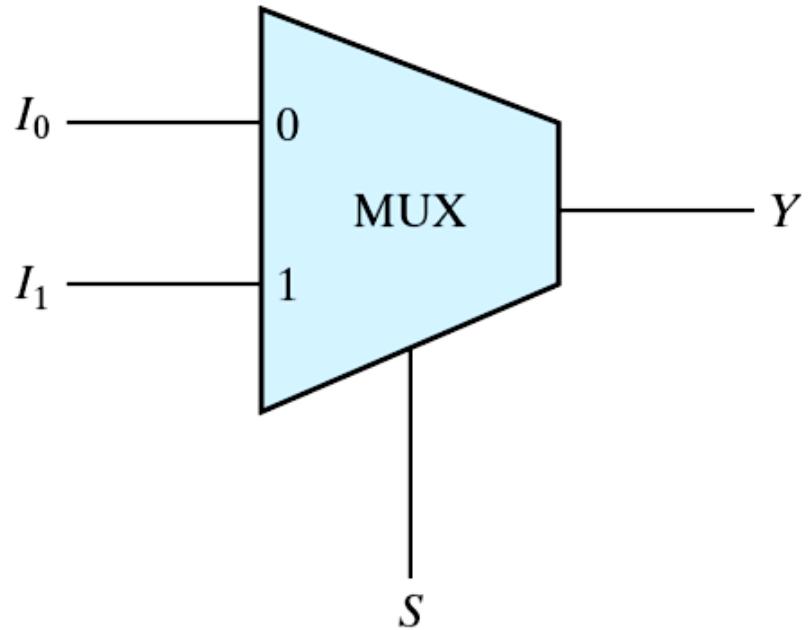


Continuous assignment





Procedural assignment (1)



Behavior:

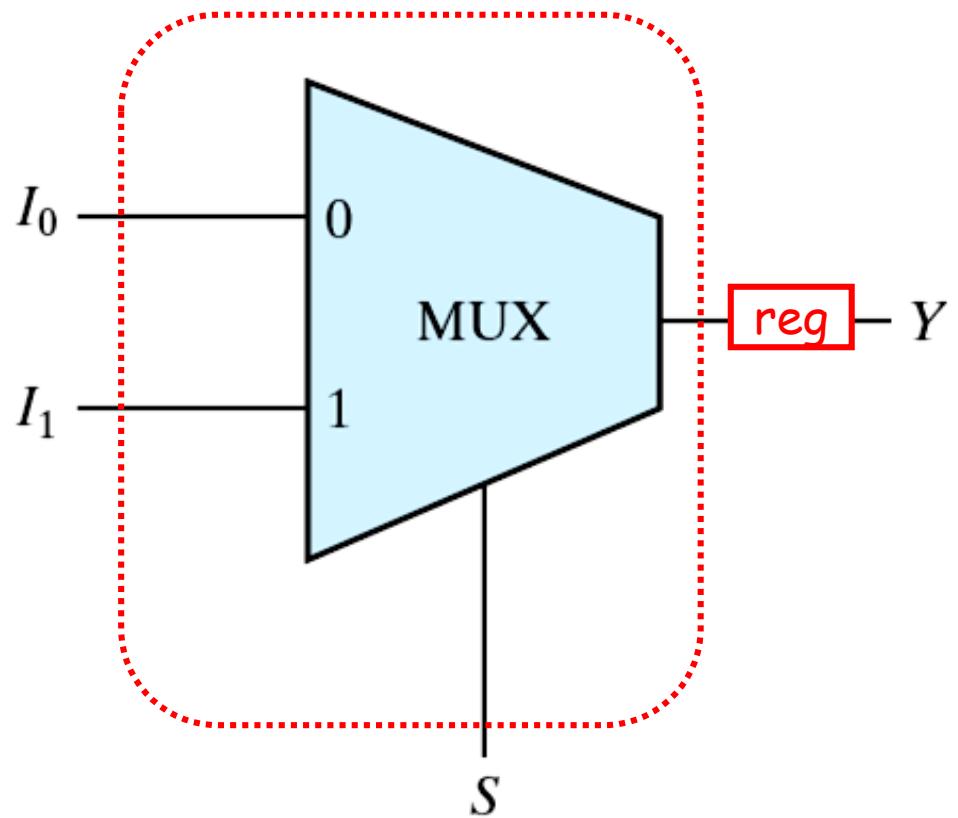
if $S == 0$
 $Y = I_0$

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;
```

Procedure assignment 的輸出必須是 reg 變數

```
always @ (A or B or select)
```

@ 後的變數有改變時，執行 always 內容

```
if (select == 1) m_out = A;
else m_out = B;
```

Procedure assignment

```
endmodule
```



比較: HDL Example 4-7

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

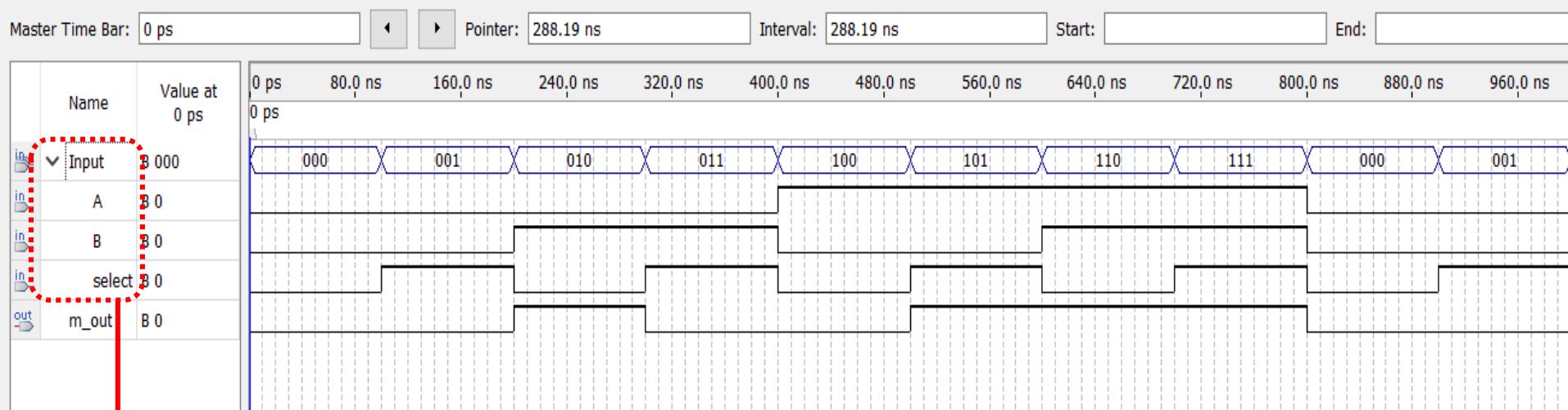
```
// Dataflow description of two-to-one-line multiplexer

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.

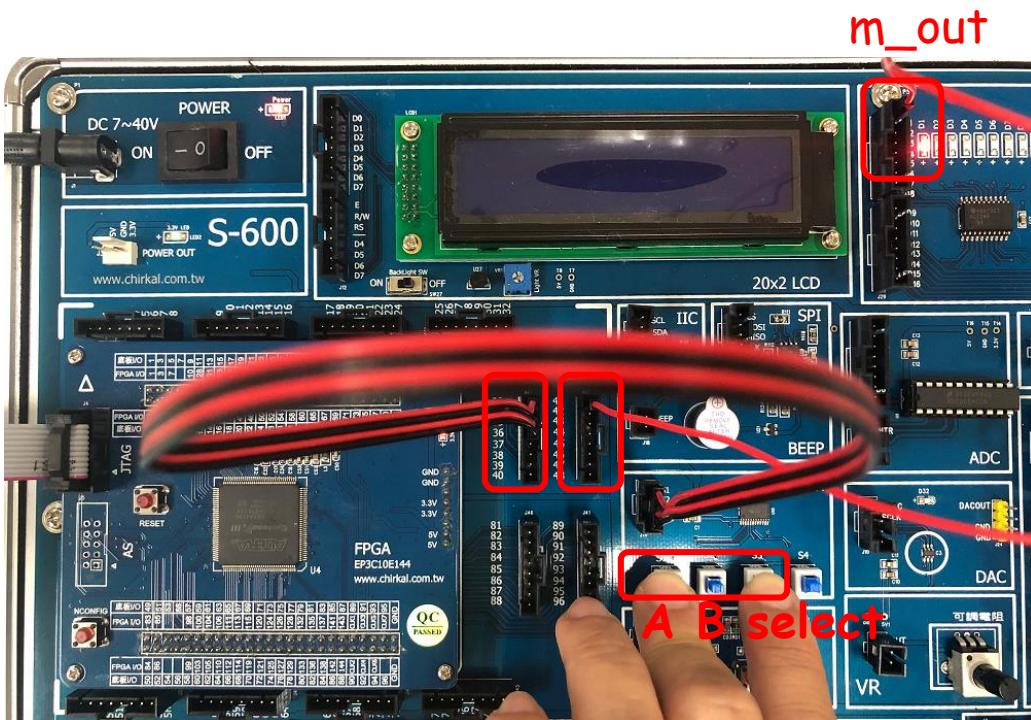


Lab1-2: FPGA implementation of Example 4.9

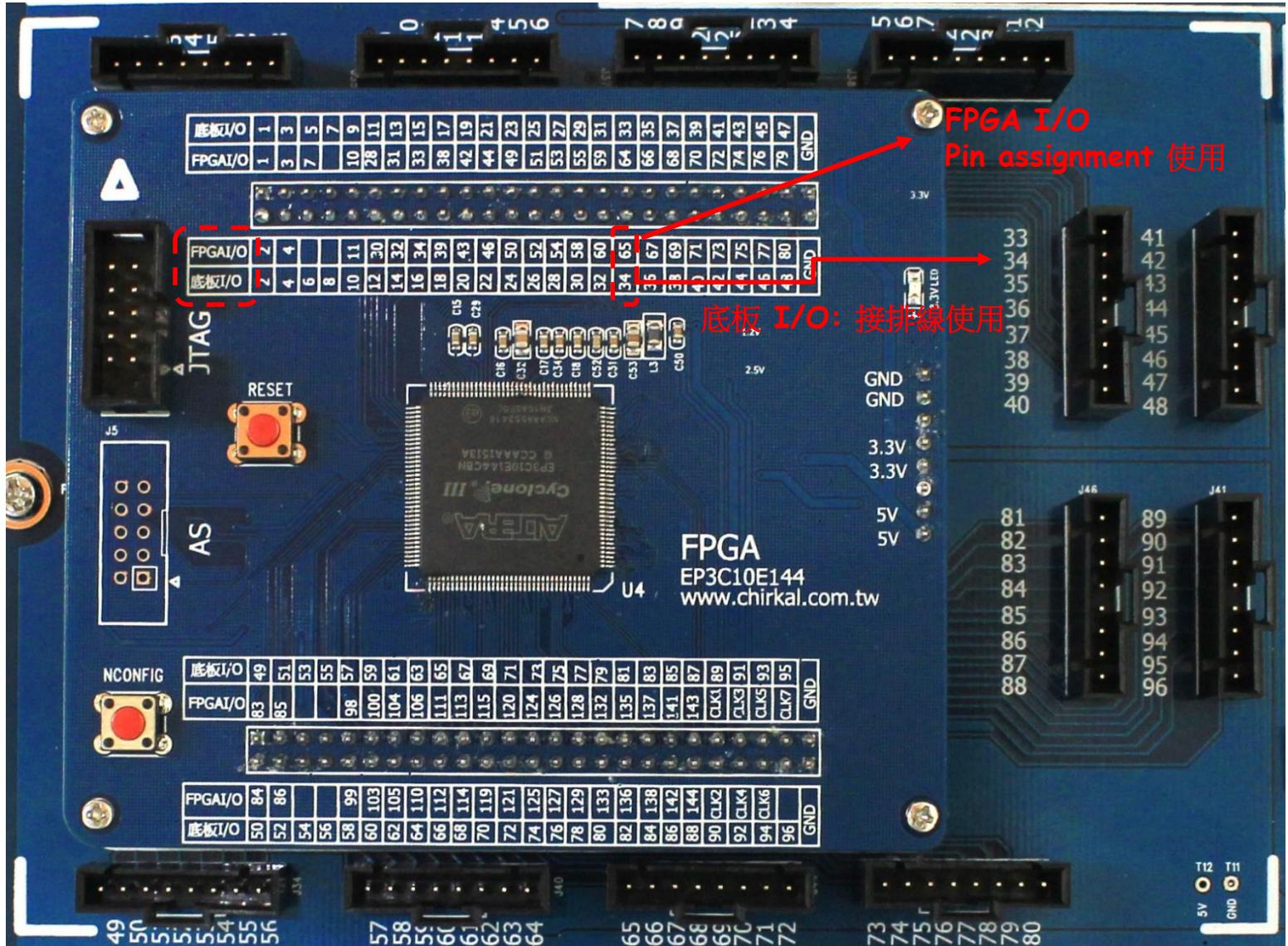
- Pin assignments (will be different if you use other board pins)

<<new>> Filter on node names: *

Status:	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; → 長度 2 的 binary number，值為 01
            2'b10:          m_out = in_2;
            2'b11:          m_out = in_3;
    endcase

```

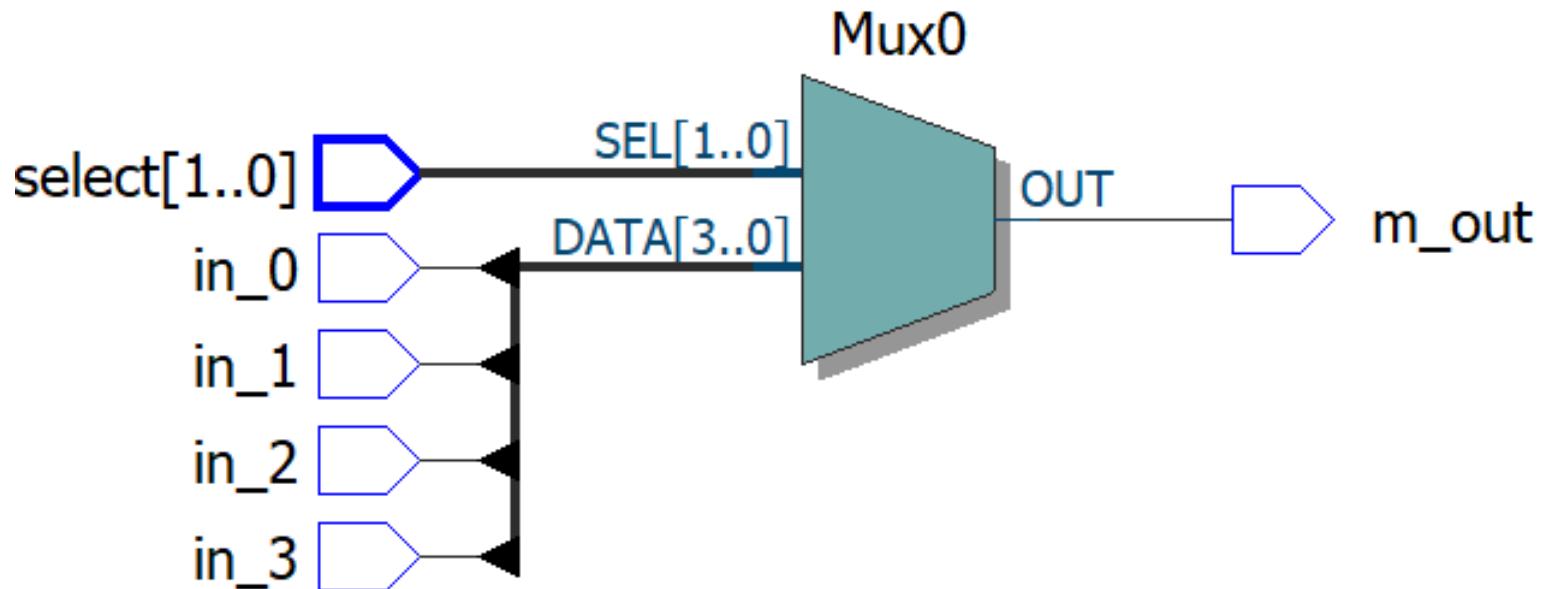
新的參數列寫法

```
endmodule
```



RTL viewer

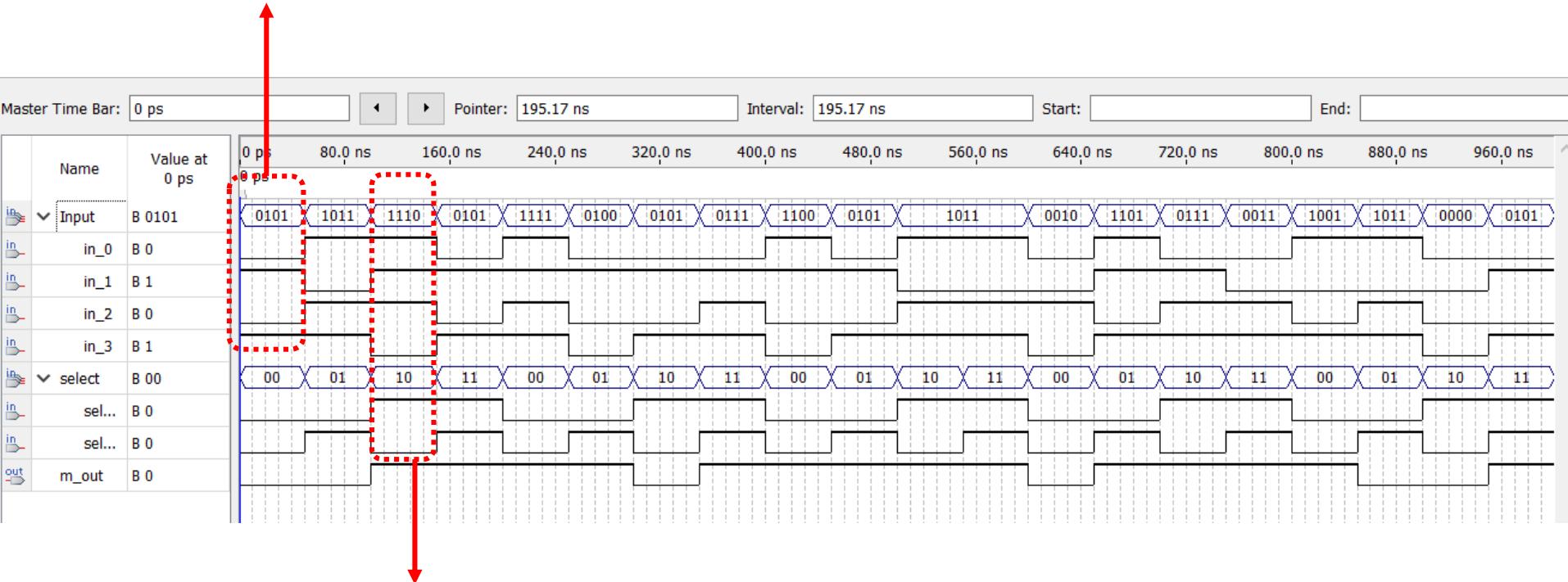
Tools -> Netlist viewers-> RTL viewer



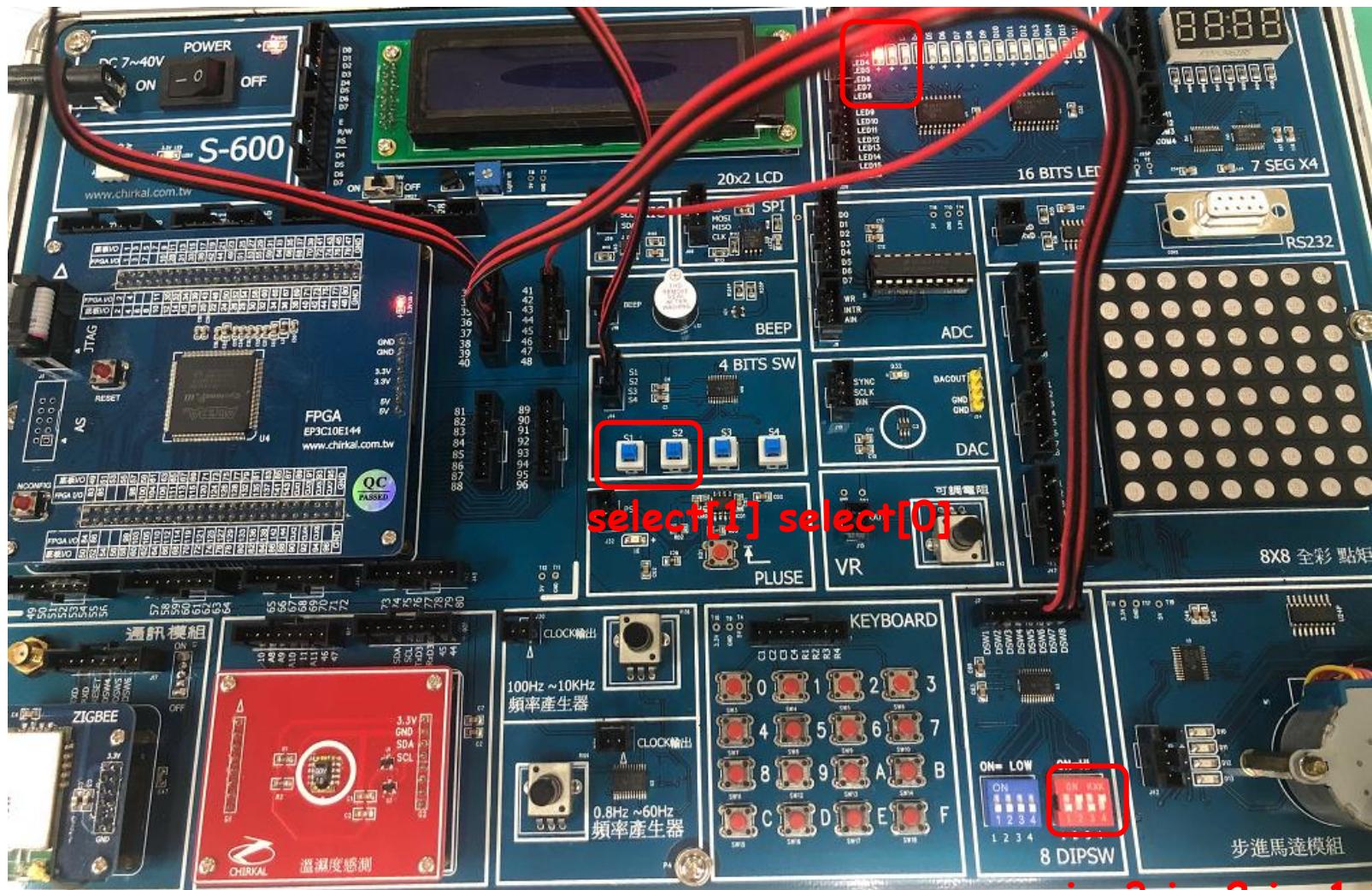


Lab#2-1: Waveform simulation of Example 4-10

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



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
```



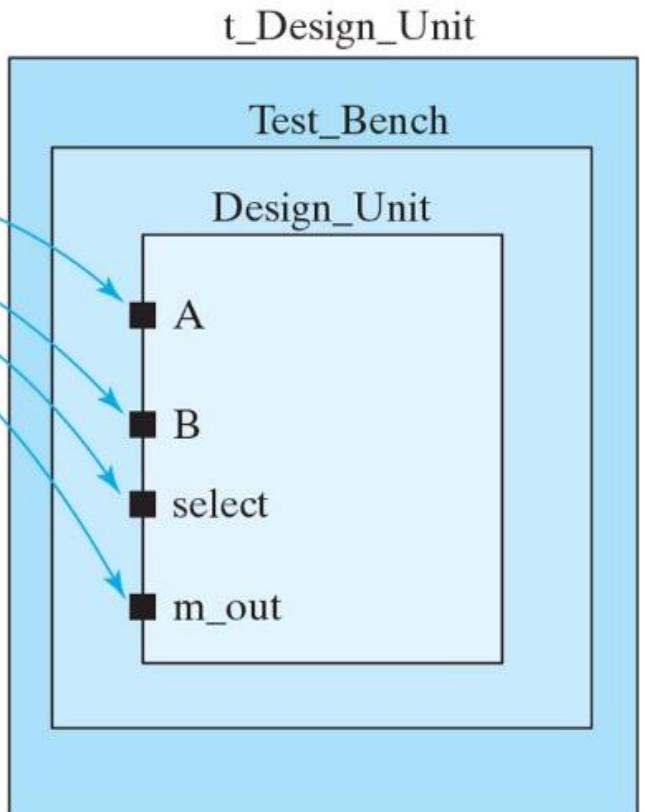
Three-bit truth table



Writing a Simple Test Bench

- Interaction between **testbench** and **design modules**

```
module 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),.m_out(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;           ← parameter, 常數的變數
    mux_2x1_df M1 (t_mux_out, t_A, t_B, t_select);   // Instantiation of circuit to be tested
    initial # stop_time $finish;        ← 一個程式裡面可以有多個 initial
    initial begin
        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
我們這裡改成使用前面寫的 mux_2x1_beh



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); 課本印刷錯誤，應為 $time, ", select ...
end
endmodule
```

```
// 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
```

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
```

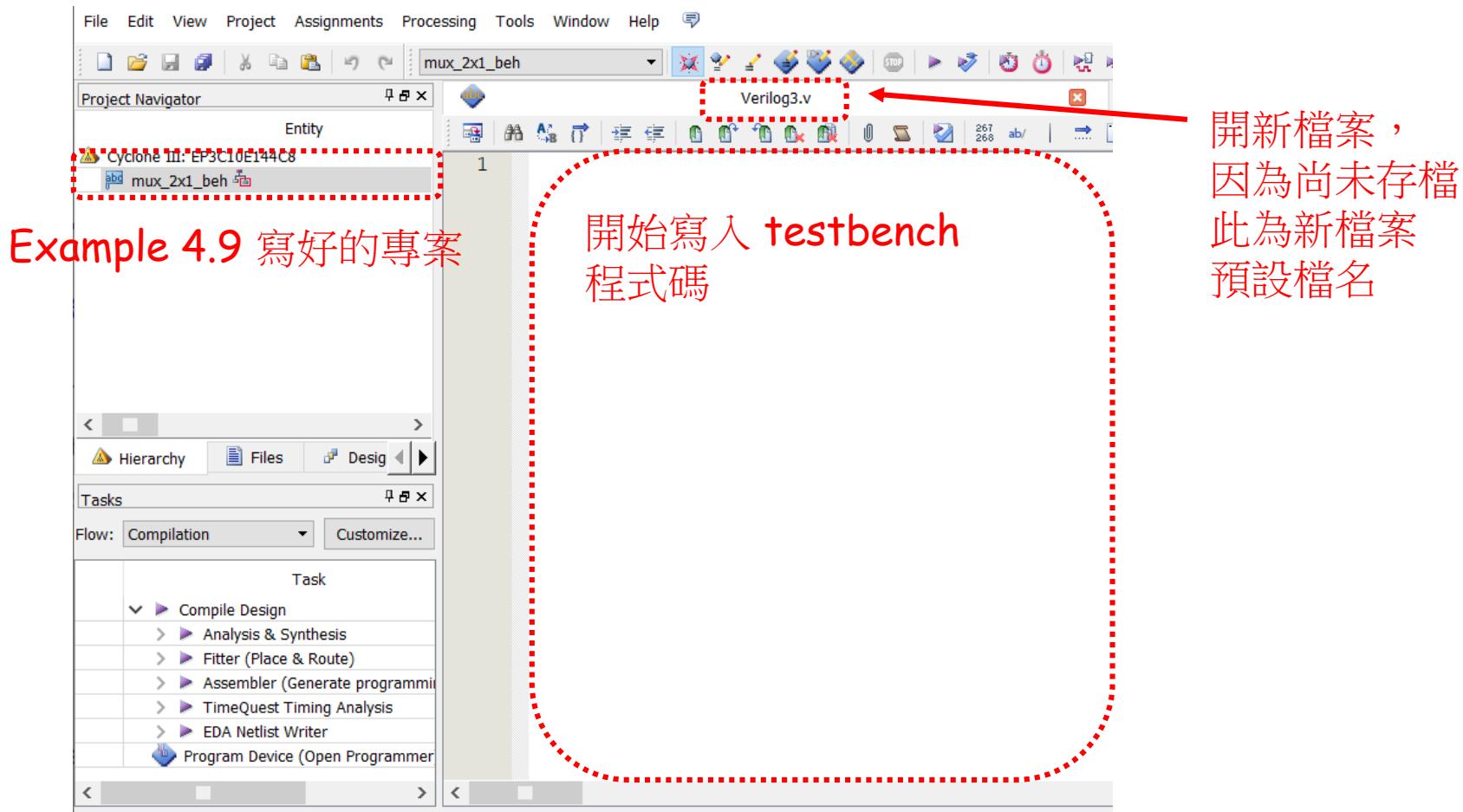
這裡不用寫，我們使用前面
寫好的 mux_2x1_beh 專案

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





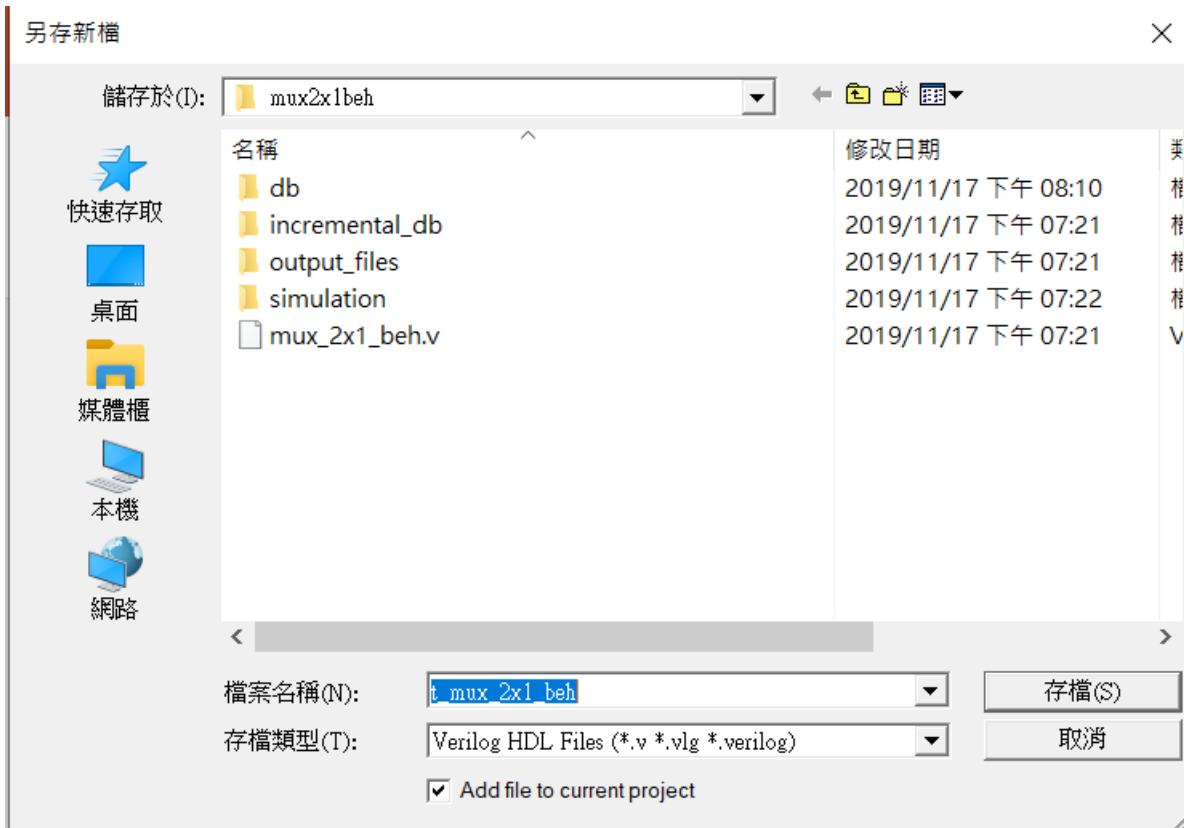
Enter test bench code

Screenshot of a VHDL editor showing the code for a test bench module. The code defines a module t_mux_2x1_beh with various ports and internal logic. A red box highlights the module declaration, and another red box highlights the instantiation of the module M1.

```
File Edit View Project Assignments Processing Tools Window Help
mux_2x1_beh t_mux_2x1_beh.v mux_2x1_beh.v
Project Navigator Entity
Cyclone III: EP3C10E144C8
mux_2x1_beh
1 module t_mux_2x1_beh;
2   wire t_mux_out;
3   reg t_A, t_B;
4   reg t_select;
5   parameter stop_time = 50;
6
7   mux_2x1_beh M1(t_mux_out, t_A, t_B, t_select);
8   .....initial #stop_time $finish;
9
10  initial begin
11    t_select = 1; t_A=0; t_B=1;
12    #10 t_A=1; t_B=0;
13    #10 t_select = 0;
14    #10 t_A=0; t_B=1;
15  end
16
17  initial begin
18    $monitor("time=", $time, ", select=%b, A=%b, B=%b, Out=%b", t_select, t_A, t_B, t_mux_out);
19  end
20
21 endmodule
```

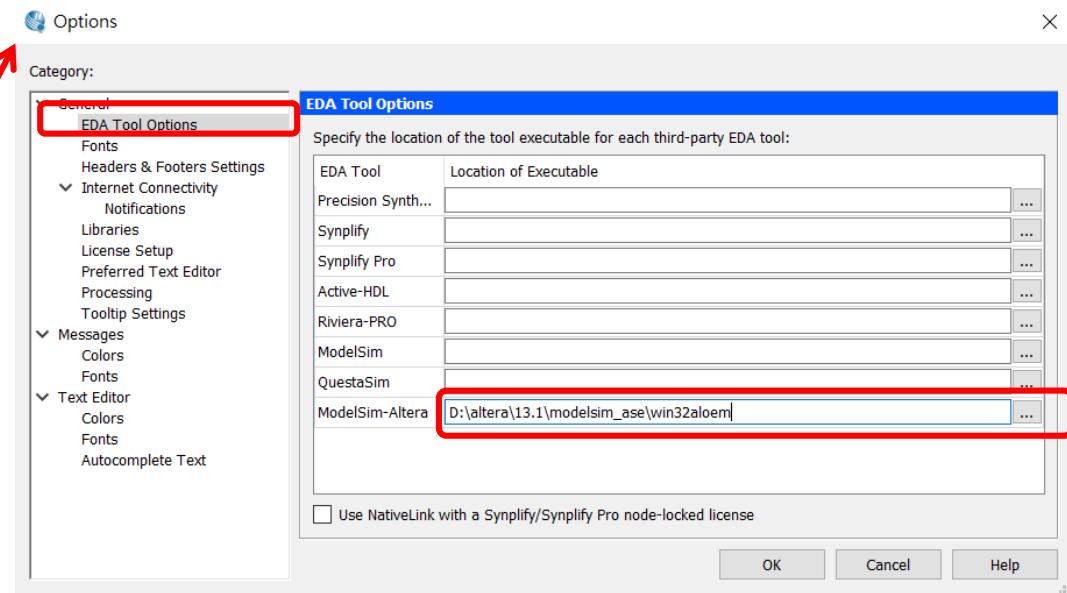
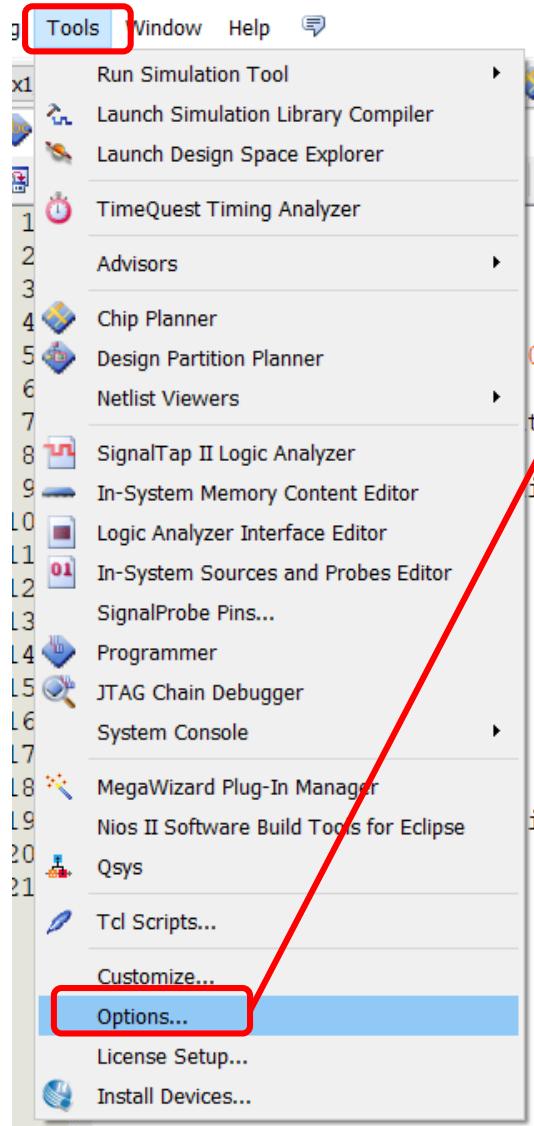


存檔，跟 Verilog module 名稱相同





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 \



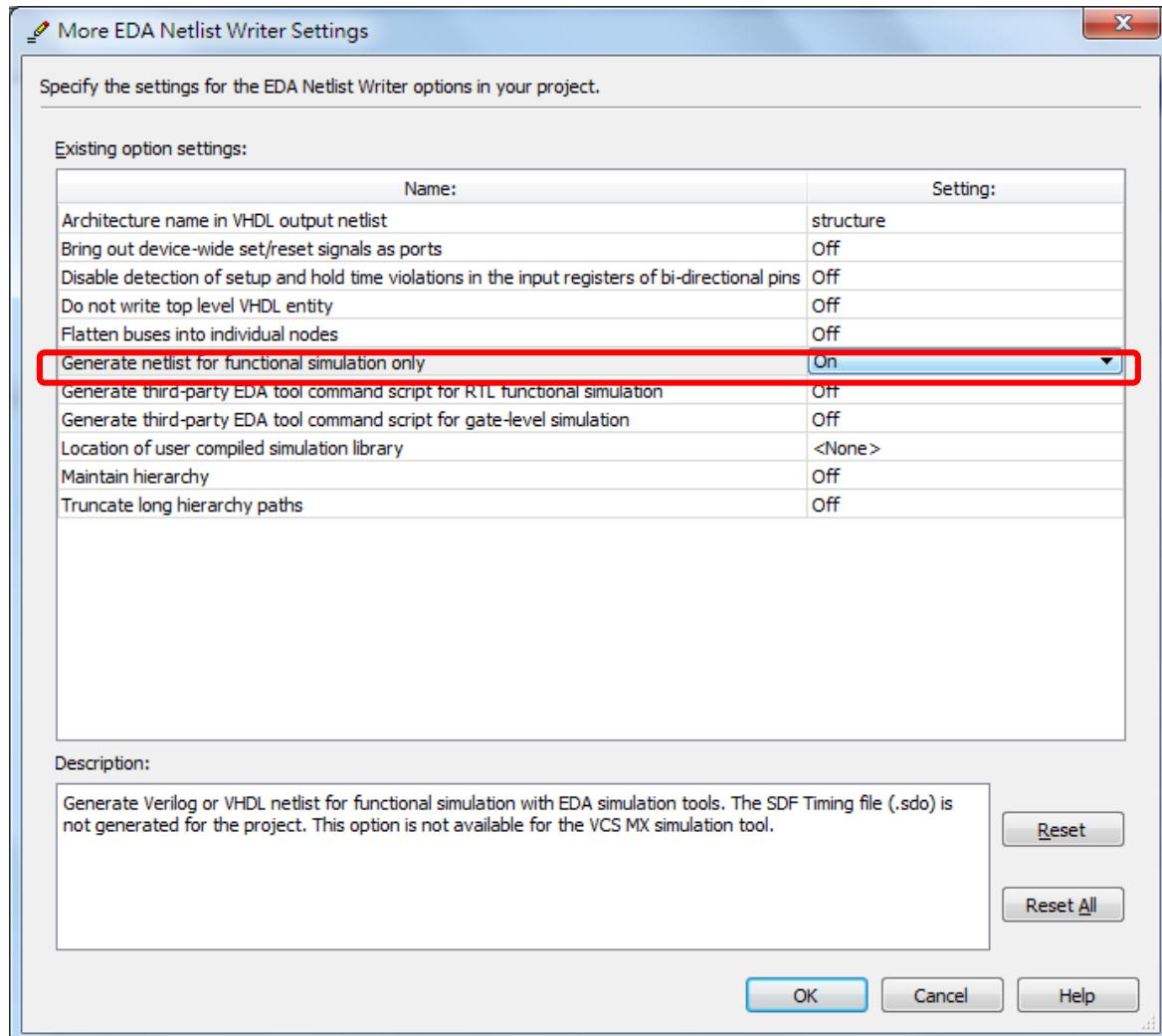


Setting for simulation (2)

The screenshot shows the Quartus Prime software interface. On the left, the 'Assignments' menu is highlighted with a red box, and the 'Settings...' option is also highlighted with a red box. A red arrow points from the 'Settings...' option to the 'Settings' dialog box on the right. The 'Settings' dialog box is titled 'Settings - mux_2x1_beh'. The 'Category' tree on the left includes General, Files, Libraries, Operating Settings and Conditions, Compilation Process Settings, EDA Tool Settings, Analysis & Synthesis Settings, and Fitter Settings. The 'Simulation' category is selected and expanded, showing options like 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, and More EDA Netlist Writer Settings... . At the bottom of the dialog box are buttons for Buy Software, OK, Cancel, Apply, and Help.



EDA netlist writer setting





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

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

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

預設時間單位

選這項，compile test bench



Test Benches

Specify settings for each test bench.

Existing test bench settings:

Name	Top Level Module	Design Instance	Run For	Test Bench File(s)

New... **Edit...** **Delete**

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

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

Test bench and simulation files

File name: []

File Name	Library	HDL Version
t_mux_2x1_beh.v		Default

... Add Remove Up Down Properties...

OK Cancel Help

名稱要正確

選擇 test bench 檔案後按 add 加入



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

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

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

上述步驟選完後的結果

< >

Buy Software OK Cancel Apply Help



Run RTL simulation

Quartus II 32-bit - D:/altera/13.1/workspace2019/mux2x1beh/mux_2x1_beh - mux_2x1_beh

File Edit View Project Assignments Processing Tools Window Help

Run Simulation Tool

RTL Simulation

Launch Simulation Library Compiler
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
SignalProbe Pins...
Programmer
JTAG Chain Debugger
System Console
MegaWizard Plug-In Manager
Nios II Software Build Tools for Eclipse
Qsys
Tcl Scripts...
Customize...
Options...
License Setup...
Install Devices...

0;
t, t_A, t_B, t_select);
ish;
t_B=1;
ime, "select=%b, A=%b, B=%b,

Project Navigator

Entity

Cyclone III: EP3C10E144C8

mux_2x1_beh

Hierarchy Files Design

Tasks

Flow: Compilation Customize...

Task

Compile Design

Analysis & Synthesis

Fitter (Place & Route)

Assembler (Generate programming)

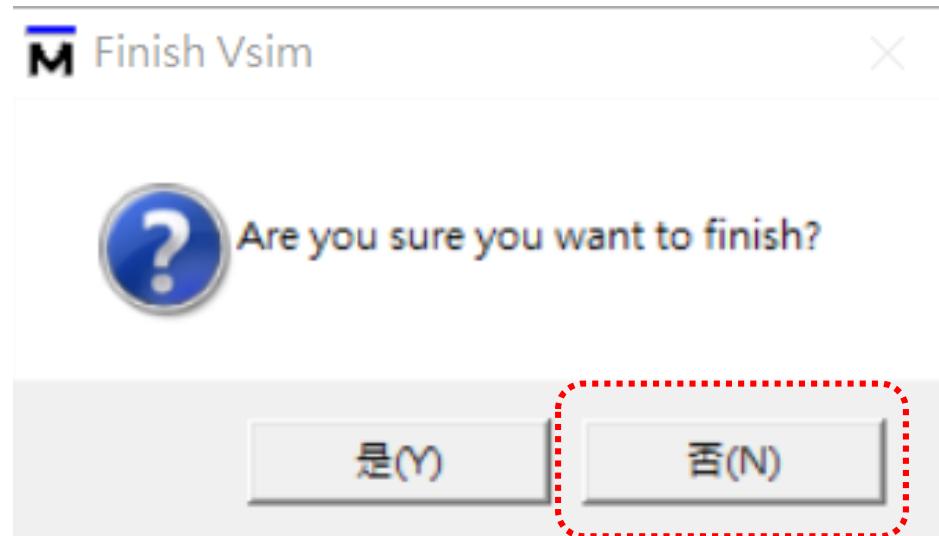
TimeQuest Timing Analysis

EDA Netlist Writer

Program Device (Open Programmer)



不要終止 Simulator





ModelSim Altera 執行結果

再按 Zoom full, 放大波形

點選此一視窗

波形被壓縮

The screenshot shows the ModelSim Altera interface with several windows open:

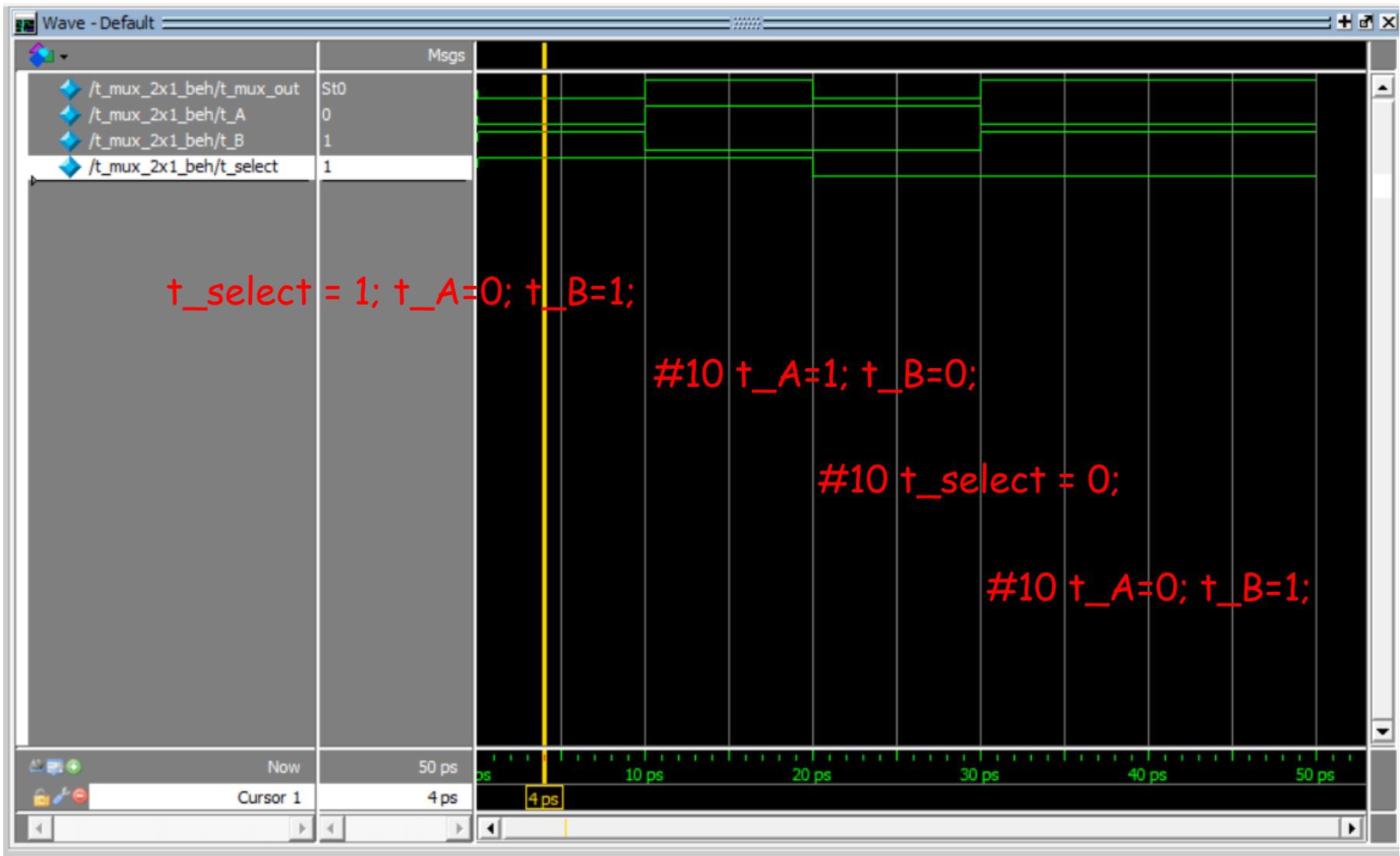
- Objects**: A table showing internal parameters and their values.
- Processes (Active)**: A table showing active processes.
- Wave - Default**: A waveform window displaying signals over time. A red dashed box highlights a specific section of the waveform, and a red arrow points to the top of the window with the text "點選此一視窗". Another red arrow points to the highlighted area with the text "波形被壓縮".
- Transcript**: A window showing simulation logs and commands.

The waveform window has a zoomed-in view of a signal. The x-axis is labeled from 0 ps to 2000 ps. The y-axis shows multiple signals, one of which is highlighted with a green border. The waveform itself is very compressed horizontally, appearing as a series of vertical steps or spikes.

```
# 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
VSIM(paused)>
```



Wave window





Transcript window

The screenshot shows the ModelSim ALTERA Starter Edition interface. The menu bar at the top includes File, Edit, View (which is highlighted with a red box), Compile, and Simulate. Below the menu is a toolbar with icons for Call Stack, Capacity, Class Browser, Coverage, Dataflow, Files (n), Library (y), List, Locals, Message Viewer, Memory List (w), Objects, Process, Profiling, Project (x), Schematic (1), Structure (z), Transcript (highlighted with a red box and connected by a red arrow to the transcript window), Verification Management, Watch, and Wave. The main workspace shows a project tree with an instance named 't_mux' and a simulation setup for '#vsim'. The right side of the screen displays the Transcript window, which contains the following text:

```
# -- Compiling module t_mux_2x1_beh
#
# Top level modules:
#     t_mux_2x1_beh
#
# vsim -t lps -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
```

A red box highlights the output from the \$monitor command, and the text '\$monitor 輸出結果' is overlaid in red on the right side of the transcript window.



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
```

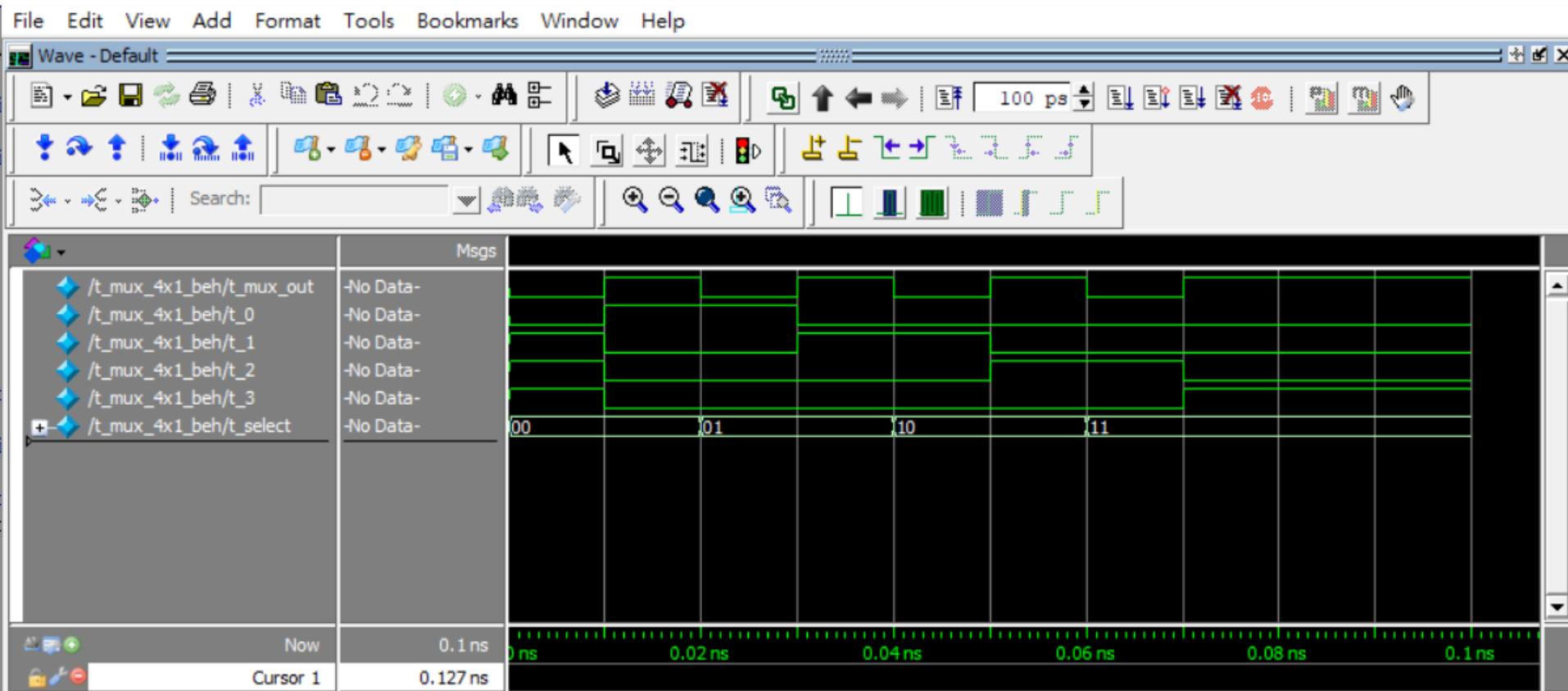
Generate test signal as the
following slides

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 lps -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 cyclone
# 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

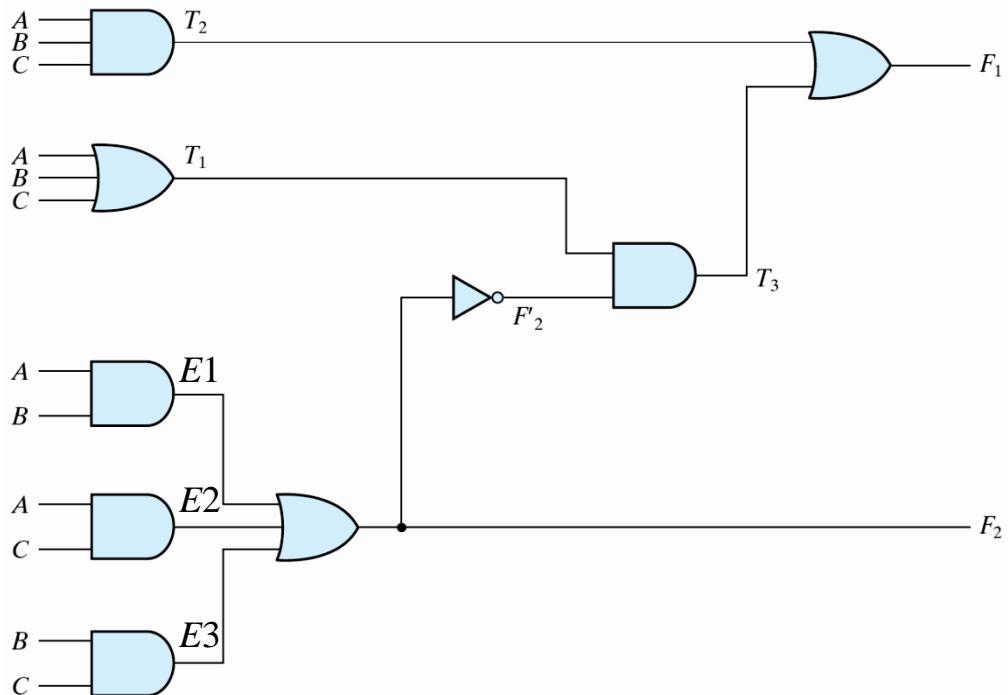


HDL Example 4-13

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)

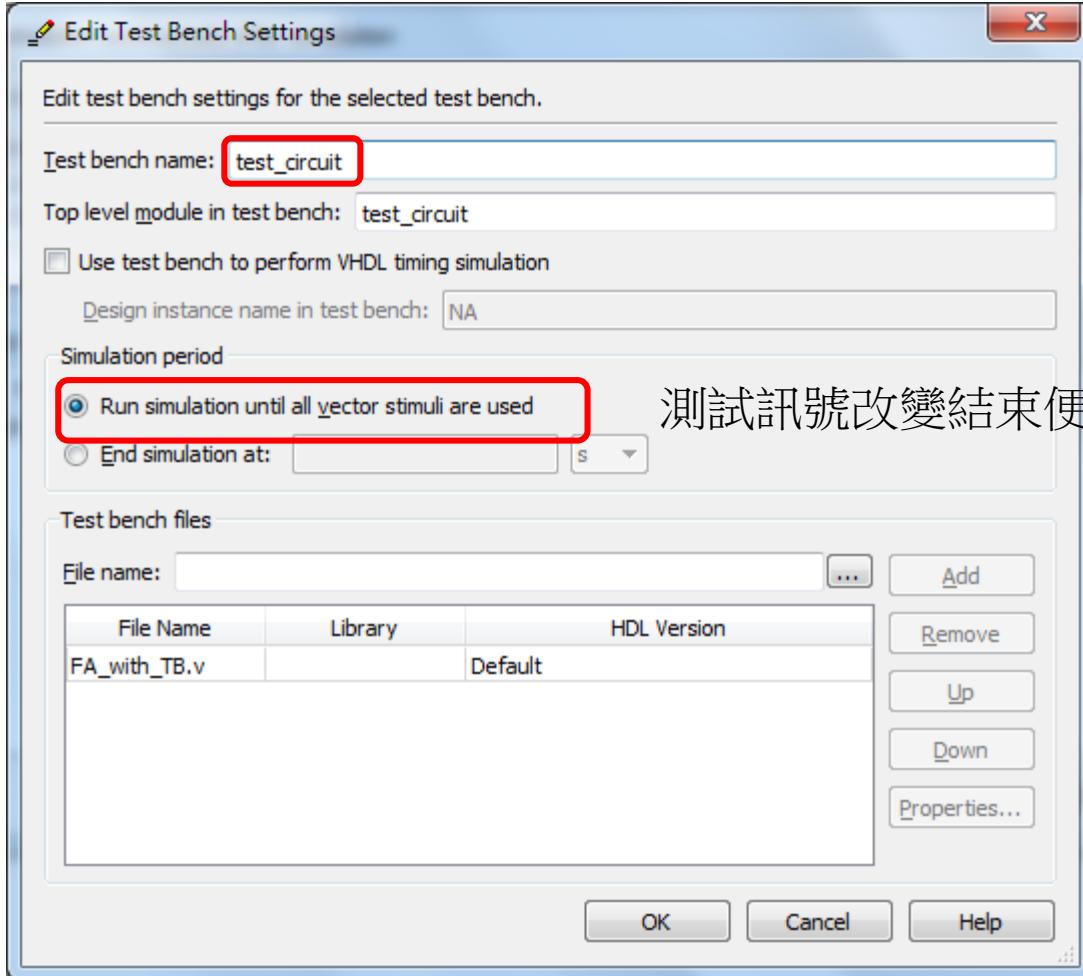
```
// Stimulus to analyze the circuit
module test_circuit;
    reg [2: 0] D;
    wire F1, F2;
    Circuit_of_Fig_4_2 M_F4_32 (D[2], D[1], D[0], F1, F2);
    initial
        begin
            D = 3'b000;
            repeat (7) #10 D = D + 1'b1;
        end
    initial
        $monitor ("ABC = %b F1 =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

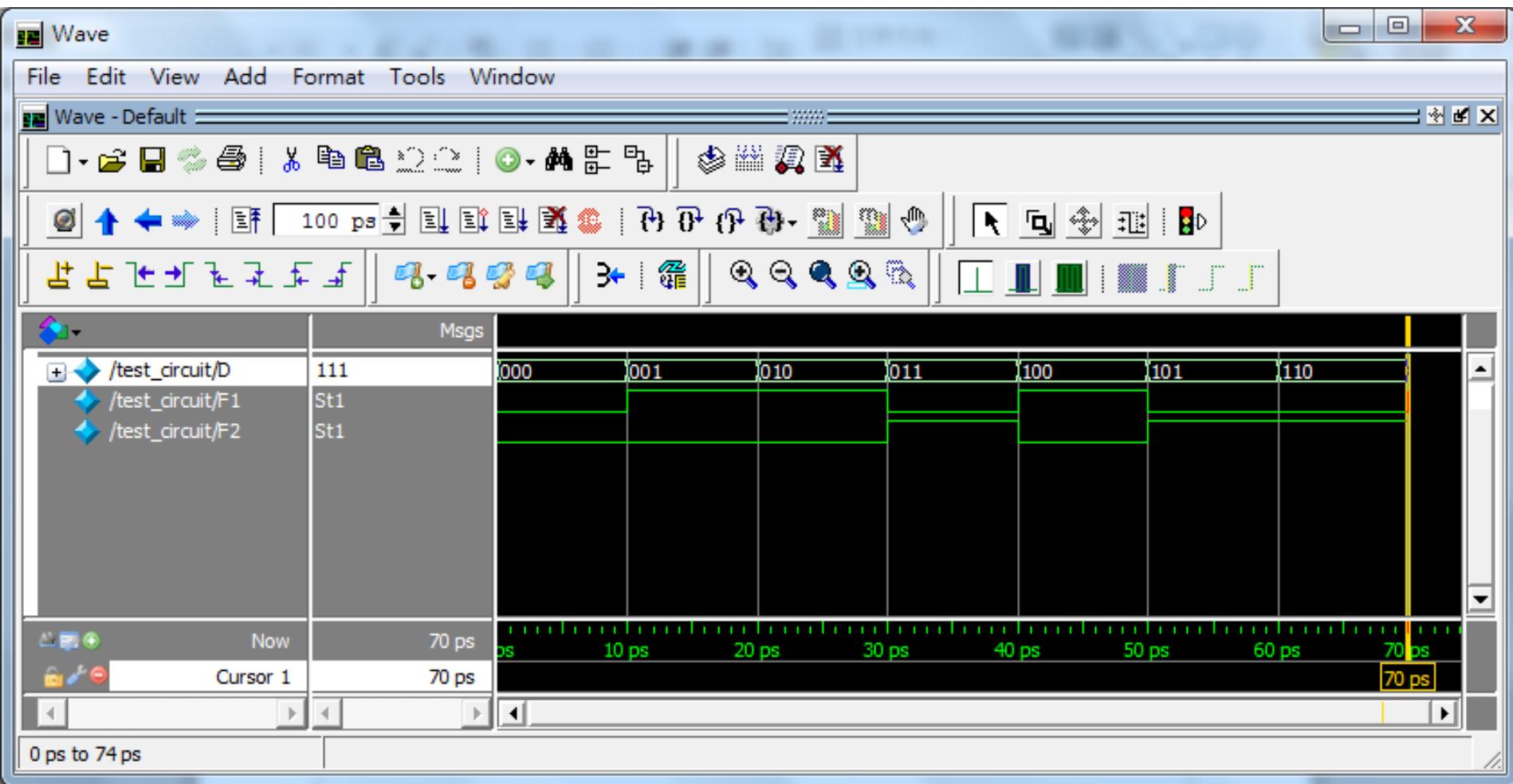


Assignment -> setting -> simulation -> test bench





RTL simulation => waveform





RTL simulation => transcript

Transcript

```
#  
# vlog -vlog01compat -work work +incdir+D:/altera/11.1sp2/workspace/FA_tb  
# Model Technology ModelSim ALTERA vlog 10.0c Compiler 2011.09 Sep 21 2011  
# -- Compiling module FA_with_TB  
# -- Compiling module test_circuit  
#  
# Top level modules:  
#     test_circuit  
#  
# vsim -t 1ps -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L al  
# vsim -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_ln  
# Loading work.test_circuit  
# Loading work.FA_with_TB  
#  
# add wave *  
# view structure  
# .main_pane.structure.interior.cs.body.struct  
# view signals  
# .main_pane.objects.interior.cs.body.tree  
# run all  
# 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  
#  
VSIM 2>
```

