



# NC-Verilog

Advisor : Lih-Yih Chiou

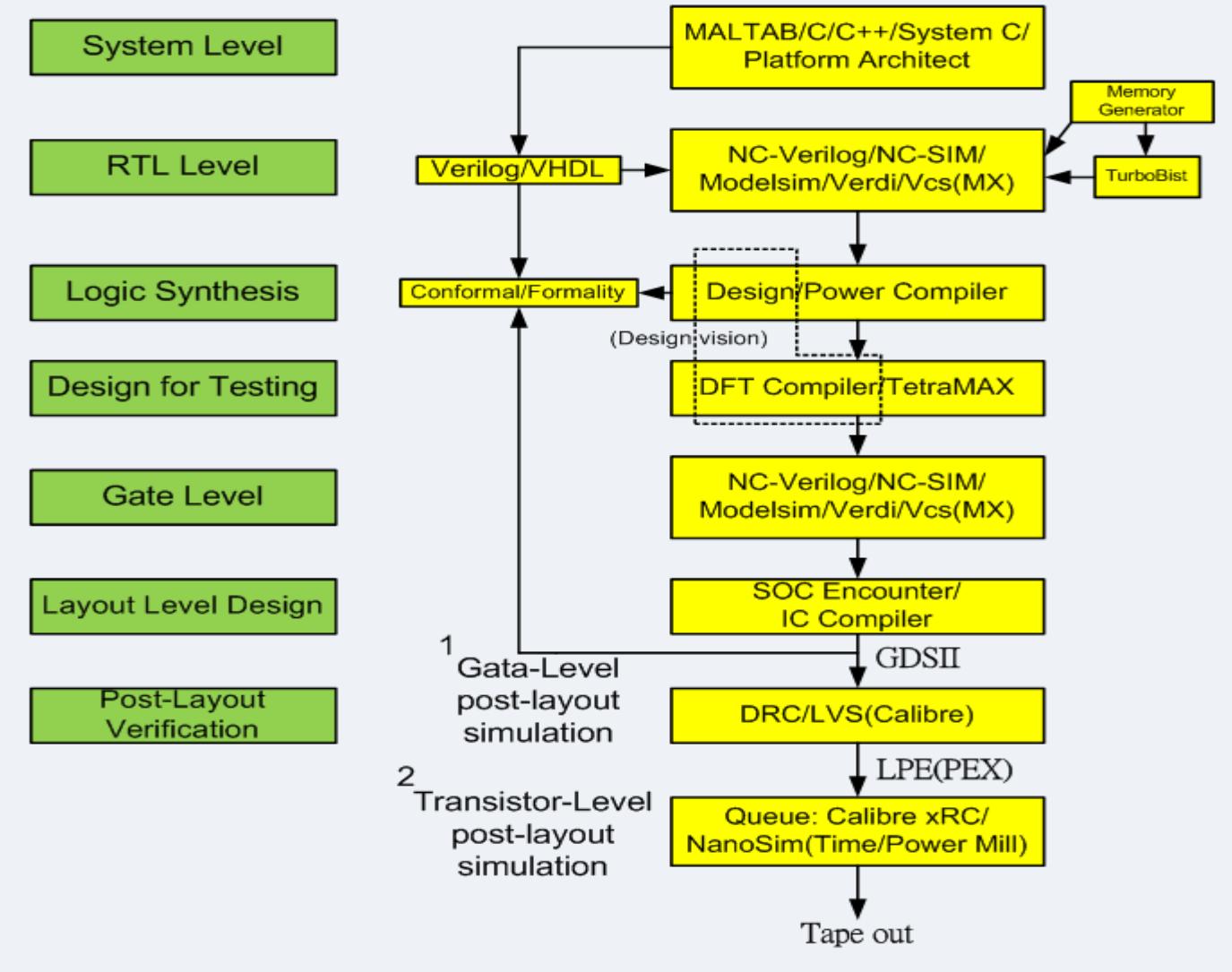
Speaker :

Date : 2010/09/16



# Cell-Based Design Flow

## Cell-Based Design Flow



# Compiling & Simulating Using NC-Verilog (1/2)

## □ To dump vcd file (attach to testbench)

→ Initial

```
begin
    $dumpfile("test.vcd");
    $dumpvars;
end
```

→ Initial

```
begin
    $fsdbDumpfile("test.fsdb");
    $fsdbDumpvars;
end
```

```
'timescale 1ns / 10ps

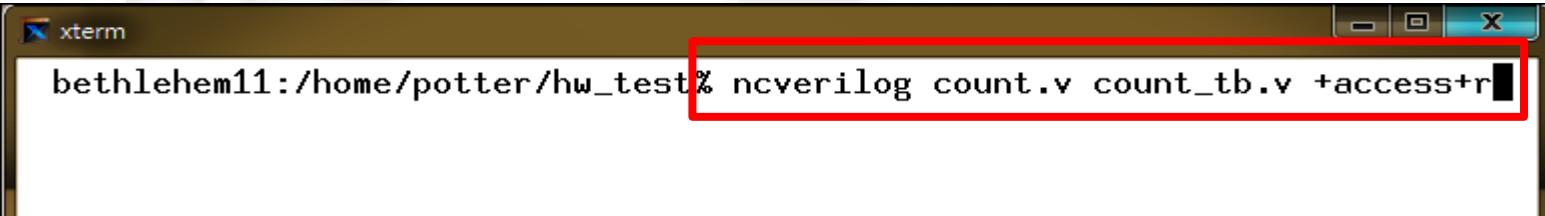
`define period 4

module count_tb;
    wire [5:0]count;
    reg clk,reset;

    count      a1(
        .count(count),
        .clk(clk),
        .reset(reset)
    );
    always #(`period/2) clk=~clk;
    initial begin
        clk=0;
        reset=0;
        #( `period/4) reset=1;
        #( `period)   reset=0;
        #( `period*30) $finish;
    end
    initial begin
        $dumpfile("test.vcd");
        $dumpvars();
    end
endmodule
```

## □ To compile & dump vcd file

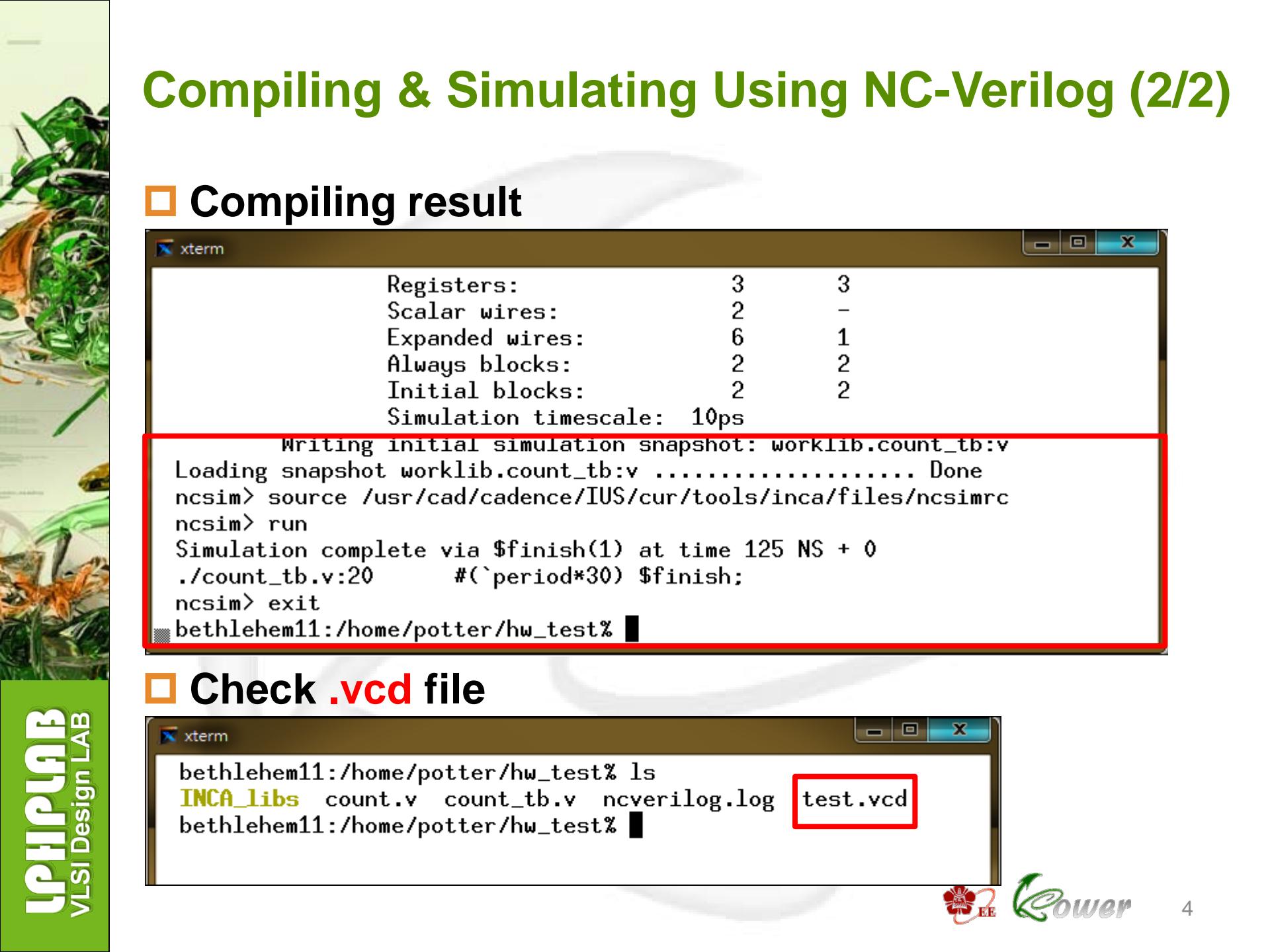
→ ncverilog test.v test\_tb.v +access+r



```
bethlehem11:/home/potter/hw_test% ncverilog count.v count_tb.v +access+r
```

# Compiling & Simulating Using NC-Verilog (2/2)

## □ Compiling result

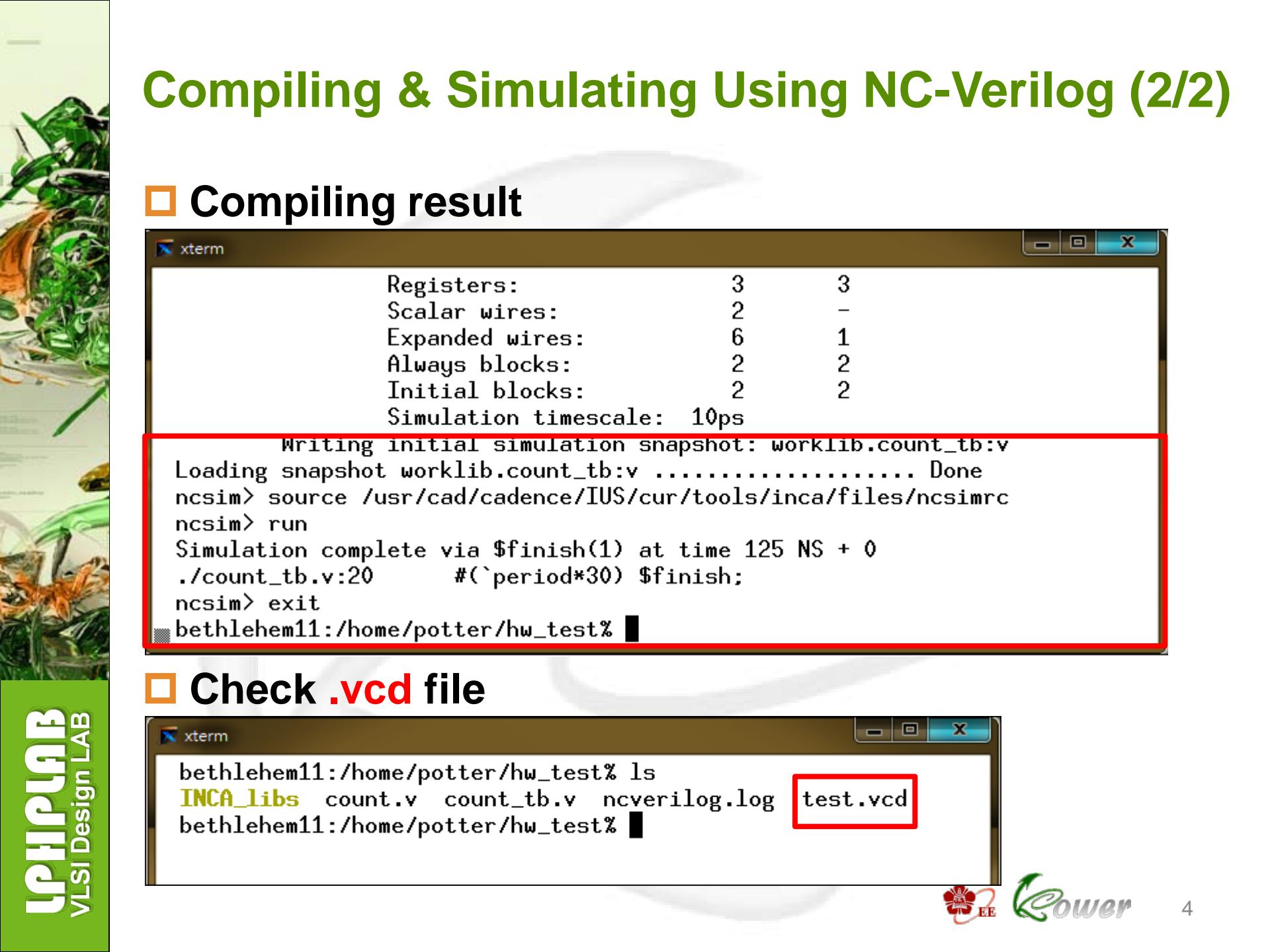


xterm

Registers:	3	3
Scalar wires:	2	-
Expanded wires:	6	1
Always blocks:	2	2
Initial blocks:	2	2
Simulation timescale:	10ps	

```
Writing initial simulation snapshot: worklib.count_tb:v
Loading snapshot worklib.count_tb:v ..... Done
ncsim> source /usr/cad/cadence/IUS/cur/tools/inca/files/ncsimrc
ncsim> run
Simulation complete via $finish(1) at time 125 NS + 0
./count_tb.v:20      #(`period*30) $finish;
ncsim> exit
bethlehem11:/home/potter/hw_test%
```

## □ Check .vcd file



```
xterm
```

```
bethlehem11:/home/potter/hw_test% ls
INCA_libs  count.v  count_tb.v  ncverilog.log  test.vcd
```

# Debussy教學

Advisor: Lih-Yih Chiou

Speaker:

Date: 2010/9/16



# Outline

## 1. Introduction

## 2. nTrace

- Overview
- Import Files
- hierarchy browser and source code
- nWave

# Outline

## 1. Introduction

## 2. nTrace

- Overview
- Import Files
- hierarchy browser and source code
- nWave

# Introduction

- Debussy是NOVAS Software, Inc(思源科技)發展的HDL Debug & Analysis tool . 現在改名為Verdi.
- 它強大的功能在：能在**HDL source code**、**schematic diagram**、**waveform**、**state bubble diagram**做即時的trace.
- Debussy本身沒有模擬器，所以要透過外部來模擬(如NcVerilog, Verilog-XL, ModelSim).

# Outline

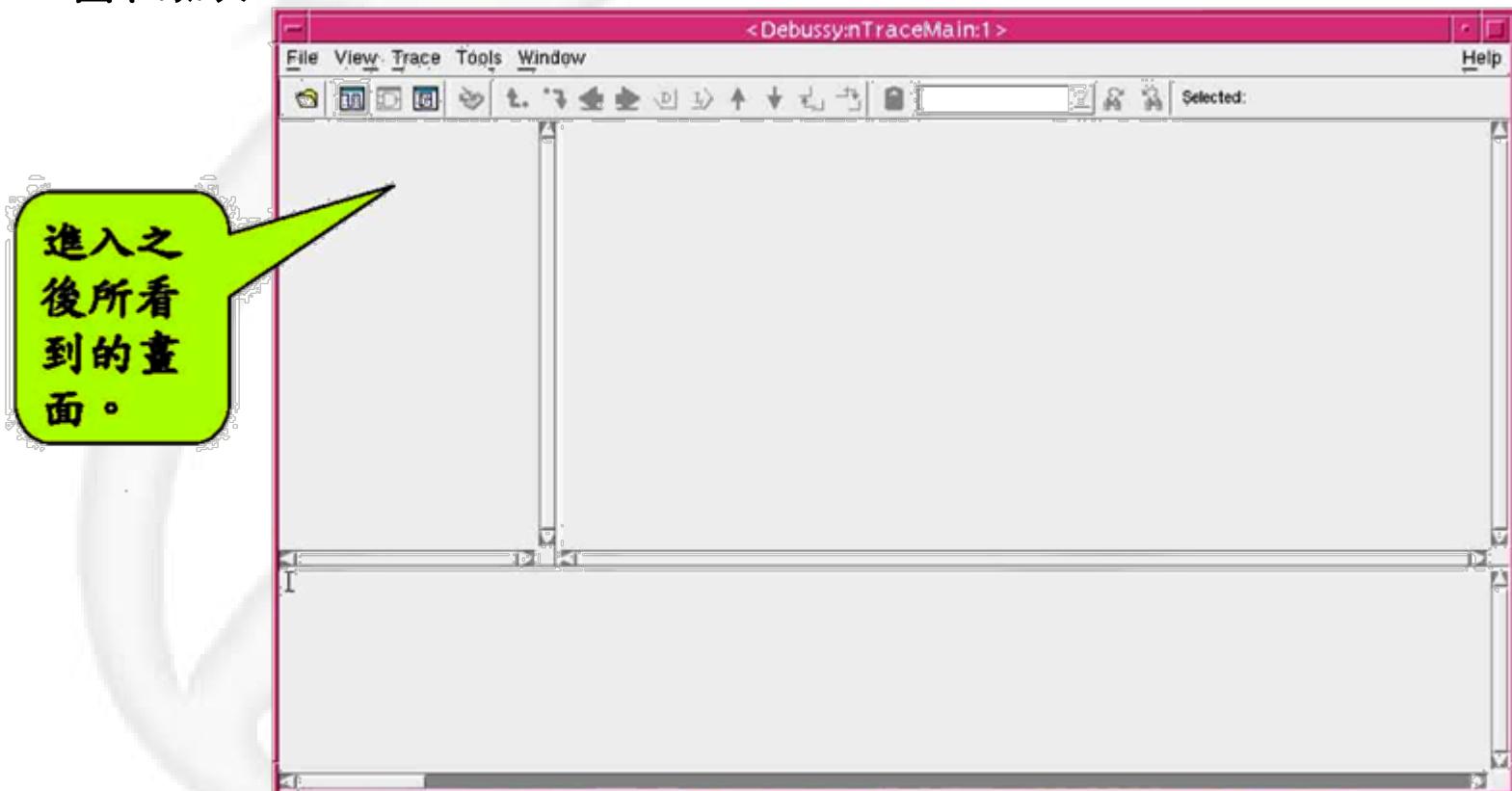
## 1. Introduction

## 2. nTrace

- Overview
- Import Files
- hierarchy browser and source code
- nWave

# nTrace Overview

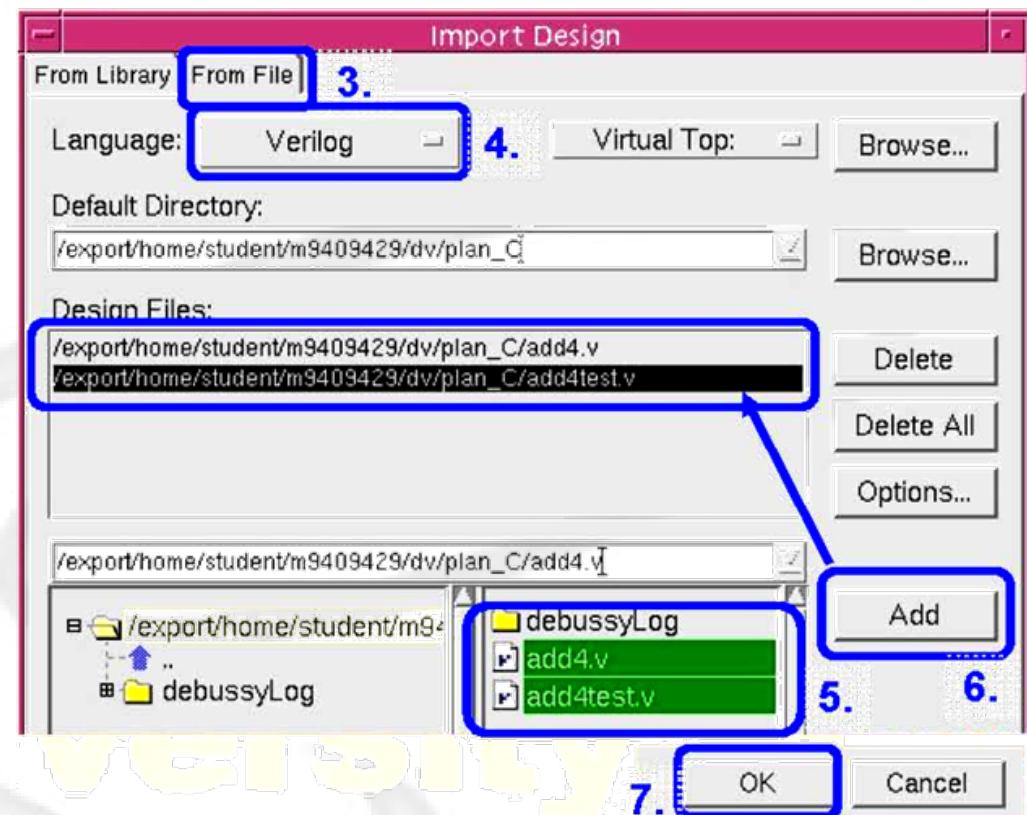
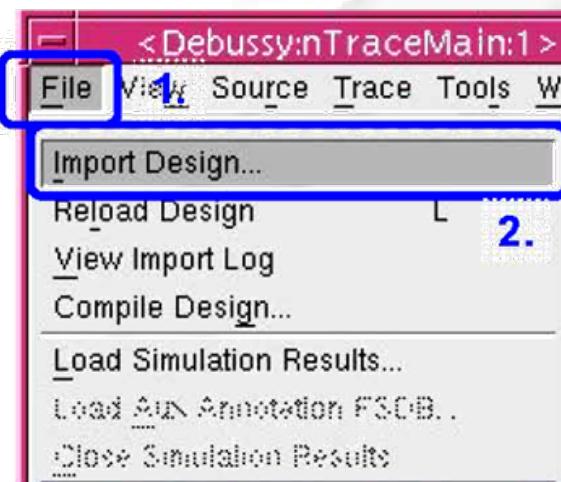
- Invoke Debussy : Debussy & 會開啟nTrace



# nTrace :: Import Files

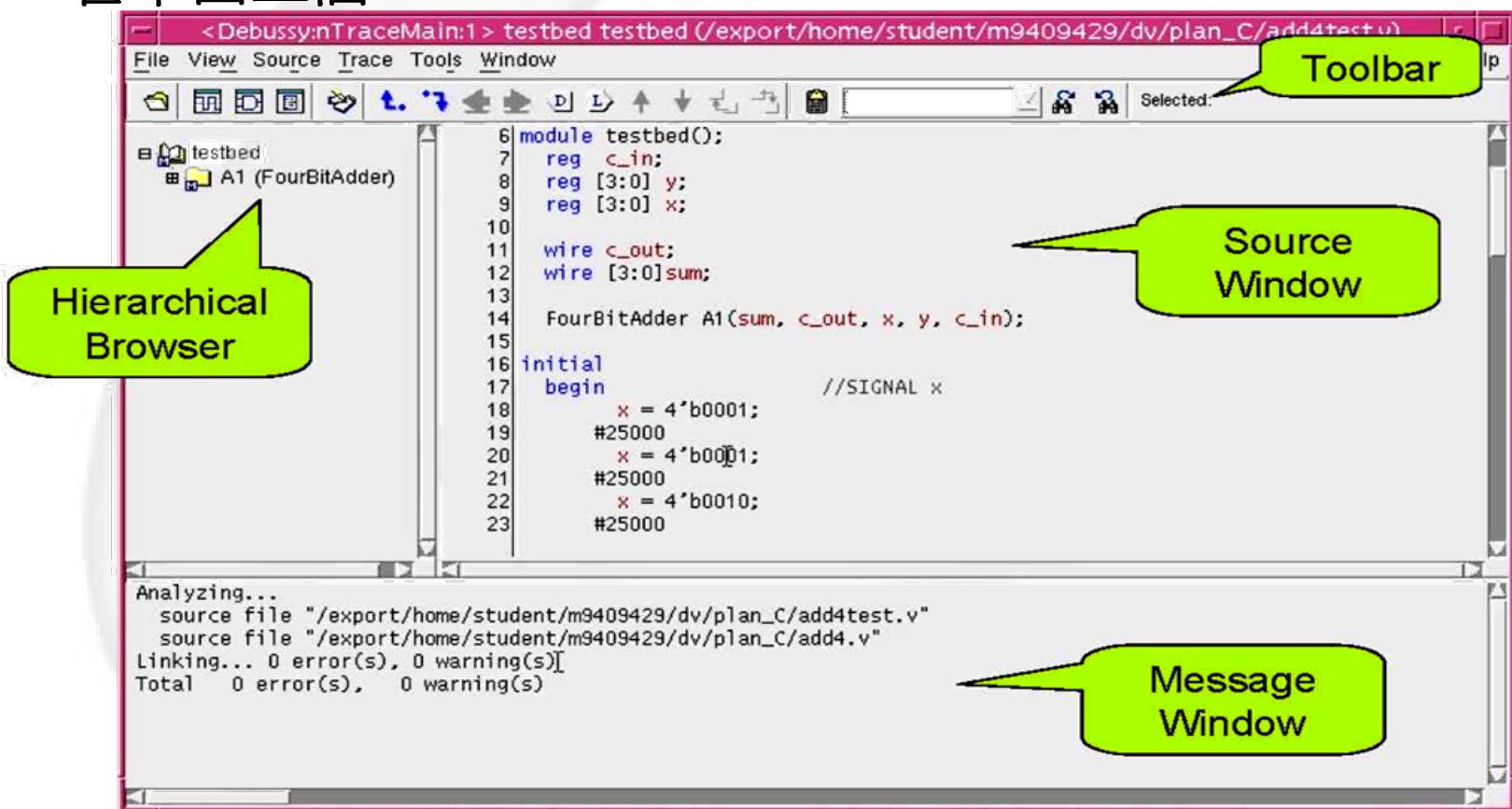
## □ Import verilog design

→ By GUI: **File** → **Import Design** → **From File**



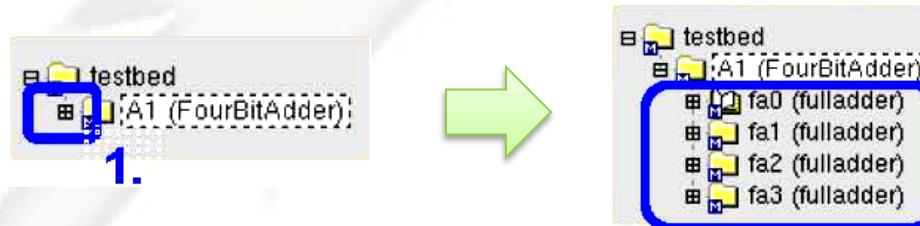
# nTrace :: Import Files

- 當打開檔案之後，每個**window**出現的畫面如下，會下面三個**window**:



# nTrace :: hierarchy browser and source code

- 在Hierarchy browser點擊A1(FourBitAdder)左側的符號“+”，可以展開四個Full A的(fa0, fa1, fa2, fa3)。



- 雙擊Hierarchy brower內的fa0...，右邊的source code window會立即切換到相對應的module。

The image shows the Hierarchy browser on the left with the 'fa0 (fulladder)' folder selected and highlighted with a blue box. To the right is a Source code window displaying Verilog code for the 'fulladder' module. The code defines inputs 'sum', 'c\_out', 'x', 'y', and 'c\_in', and outputs 'sum' and 'c\_out'. It uses primitives 'wire', 'xor', and 'and' to implement the adder logic. The line numbers from 8 to 25 are visible on the left side of the code.

```
8 module fulladder(sum, c_out, x, y, c_in);
9   output sum, c_out;
10  input x, y, c_in;
11
12  wire a, b, c;
13
14  xor (a, x, y);
15  xor (sum, a, c_in);
16
17  and (b, x, y);
18  and (c, a, c_in);
19  or (c_out, c, b);
20 endmodule
21
22 /**
23 * 4-Bit Adder *****
24 */
25 module FourBitAdder(sum, c_out, x, y, c_in);
26   output [3:0] sum;
```

# nTrace :: hierarchy browser and source code

- 如果你想追蹤fa3的話，雙擊source code window內的“fulladder”，軟體會自動幫你追蹤出引用fa3的地方，結果顯示於圖b – A1(FourBitAdder)的fa3；若再雙擊fa3，則又回到“fulladder”。
- 你可以藉由這個方法，輕易的追蹤出project內的所有design彼此之間的關連性。

The diagram illustrates the flow of control between two source code windows, labeled '圖a' and '圖b'. In '圖a', the 'fulladder' module is defined with the following code:

```
8 module Fulladder(sum, c_out, x, y, c_in);
9   output sum, c_out;
10  input x, y, c_in;
11
12  wire a, b, c;
13
14  xor (a, x, y);
15  xor (sum, a, c_in);
16
17  and (b, x, y);
18  and (c, a, c_in);
19  or (c_out, c, b);
20 endmodule
```

A red box highlights the 'fulladder' keyword at line 8, and a red arrow points from this box to the same keyword in '圖b' at line 35. In '圖b', the 'FourBitAdder' module is defined with the following code:

```
30 wire c1, c2, c3;
31
32 fulladder fa0(sum[0], c1, x[0], y[0], c_in);
33 fulladder fa1(sum[1], c2, x[1], y[1], c1);
34 fulladder fa2(sum[2], c3, x[2], y[2], c2);
35 fulladder fa3(sum[3], c_out, x[3], y[3], c3);
36 endmodule
```

A red box highlights the 'fa3' label at line 35, and another red arrow points from this box back to the 'fulladder' keyword in '圖a' at line 8.

圖a

圖b

# nTrace :: hierarchy browser and source code

- 除了追蹤各個**module**之間的關連性，也可以用同樣的方法來追蹤出**signal's drivers**和**loads**。
- 若雙擊**FourBitAdder**的訊號**c\_in**，將顯示該訊號的所有**drive loads**，結果如**message window**顯示，有4處**drive c\_in**，分別在**add4test line 69, 71, 73, 74**。

The screenshot shows the nTrace interface with a Verilog module and its signal drivers.

**Verilog Module:**

```
24 module [FourBitAdder(sum, c_out, x, y, c_in);  
25   output [3:0] sum;  
26   output c_out;  
27   input [2:0] x, y;  
28   input t c_in;  
29  
30   wire c1, c2, c3;  
31  
32   fulladder fa0(sum[1], c1, x[0], y[0], c_in);  
33   fulladder fa1(sum[1], c2, x[1], y[1], c1);  
34   fulladder fa2(sum[2], c3, x[2], y[2], c2);  
35   fulladder fa3(sum[3], c_out, x[3], y[3], c3);  
36 endmodule  
37  
38
```

**Message Window:**

```
[3] testbed.A1.c_in /* results of traced driver */  
*<D> /export/home/student/m9409429/dv/plan_C/add4test v(68): c_in = 1'b0;  
*<D> /export/home/student/m9409429/dv/plan_C/add4test v(70): c_in = 1'b1;  
*<D> /export/home/student/m9409429/dv/plan_C/add4test v(72): c_in = 1'b0;  
*<D> /export/home/student/m9409429/dv/plan_C/add4test v(74): c_in = 1'b1;  
*testbed : 4 driver(s)  
*Total : 4 driver(s)
```

**Signal Browser:**

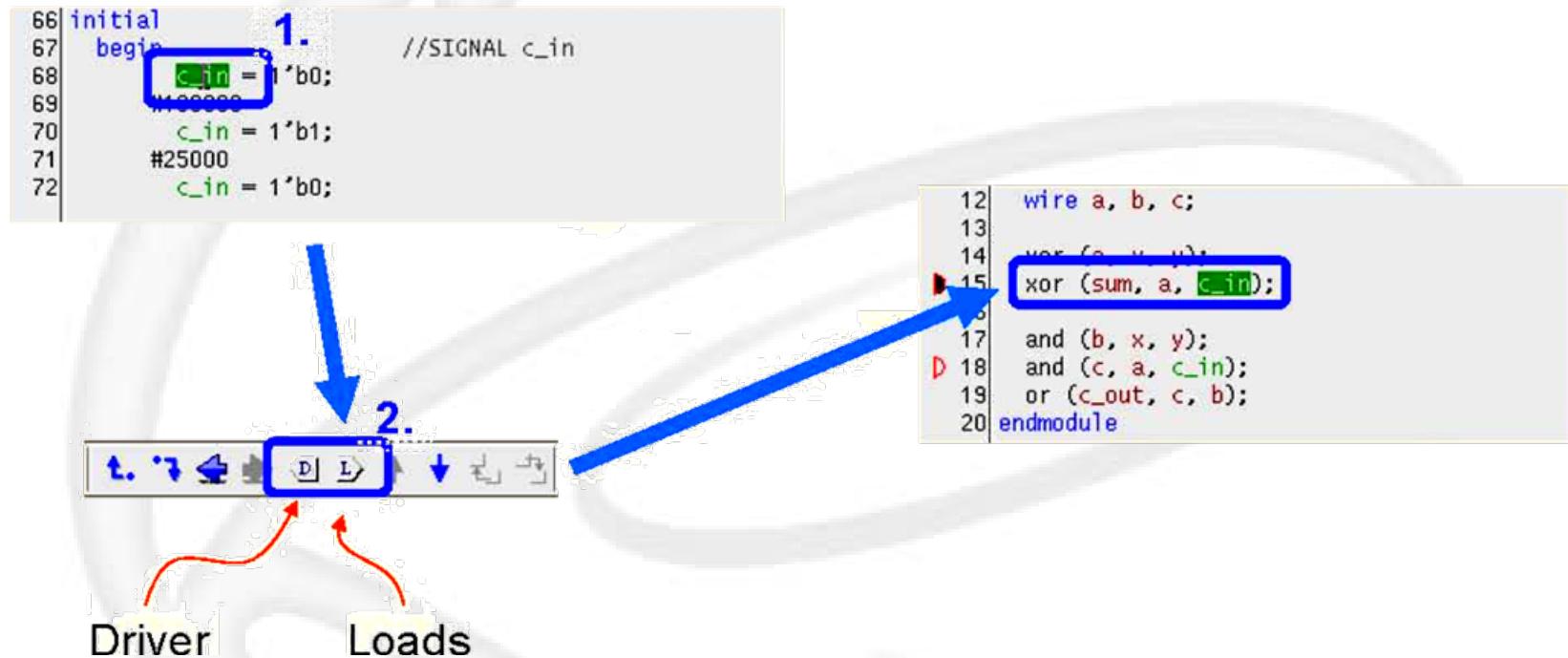
```
65  
66 initial  
67 begin  
68   c_in = 1'b0;  
69   #100000  
70   c_in = 1'b1;  
71   #25000  
72   c_in = 1'b0;  
73   #25000  
74   c_in = 1'b1;  
75   #100000  
76   ;
```

**Annotations:**

- A blue arrow points from the highlighted **c\_in** in the Verilog code to the corresponding entry in the message window.
- A blue box highlights the entries in the message window for lines 68, 70, 72, and 74.
- A blue box highlights the signal browser entries for lines 68, 70, 72, and 74.
- A green box contains the text: **!!!可用來查下一個或上一個的load.** with up and down arrows indicating navigation.

# nTrace :: hierarchy browser and source code

- 選定line 69，*Trace*→*load*則可用來追蹤該訊號的所有“load”。



# nTrace :: nWave

□ 接下來的操作，需要利用模擬器所產生的資料(.vcd)，用“nWave”，來顯示訊號波形。有兩種作法：

→ 直接利用nWave來開啟波形，有三種方法打開nWave：

◆ 按nTrace工具列中的New Waveform Icon.

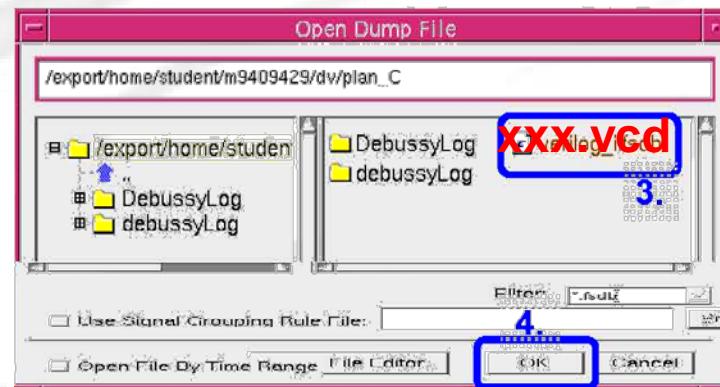
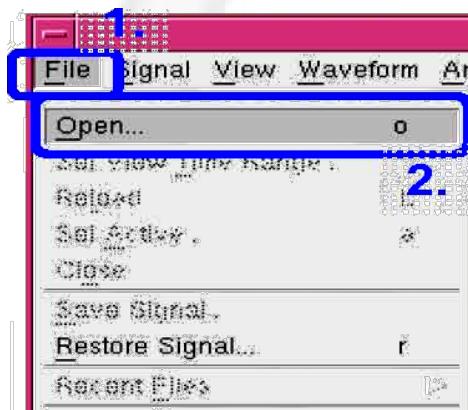
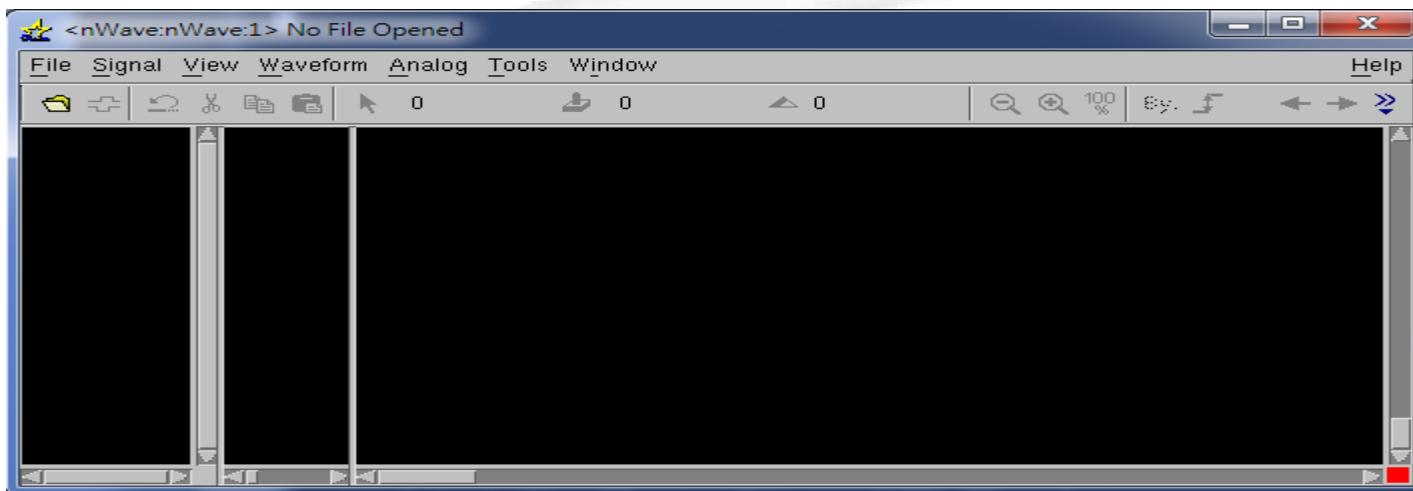


◆ (nTrace) Tools → New Waveform.



◆ %nWave & (注意大小寫)

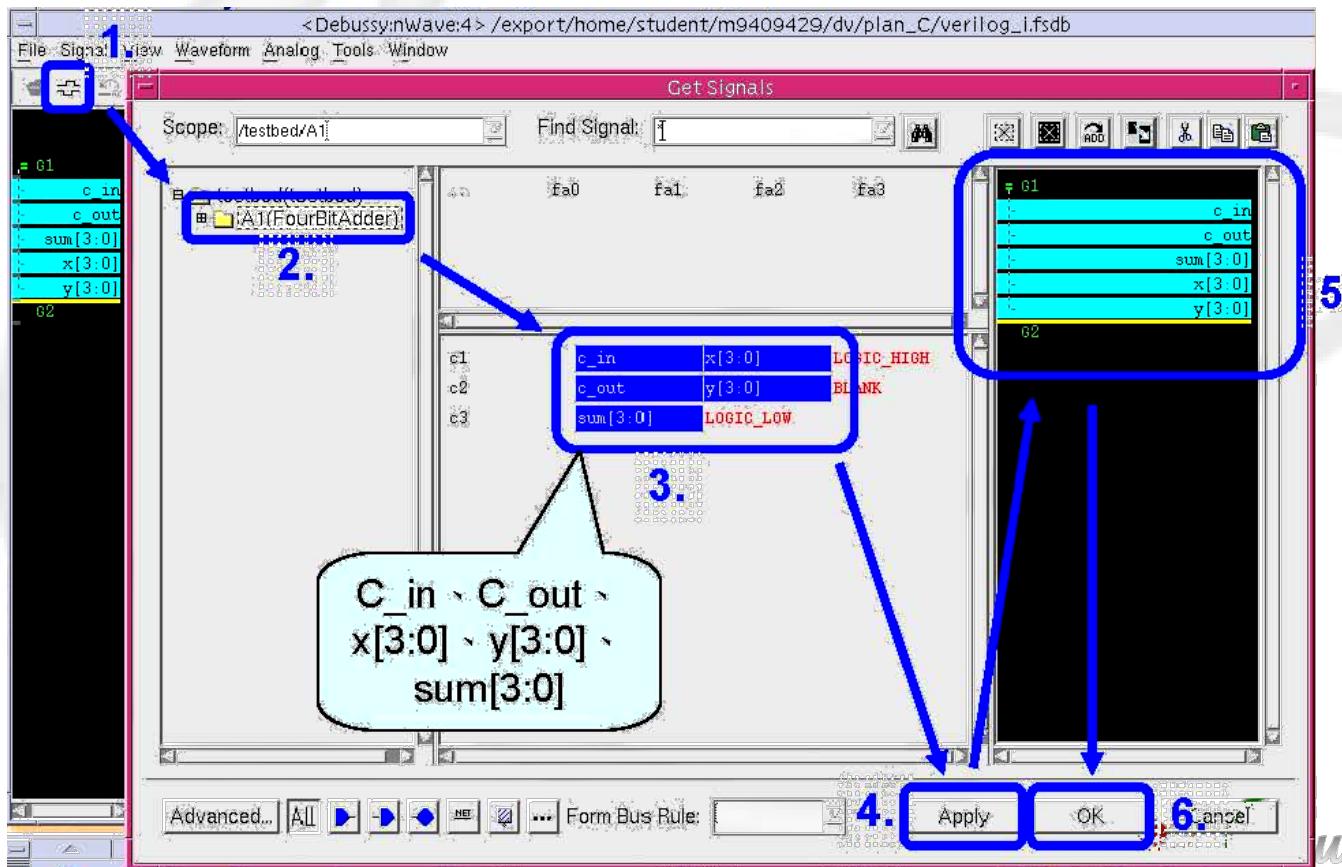
# nTrace :: nWave



# nTrace :: nWave

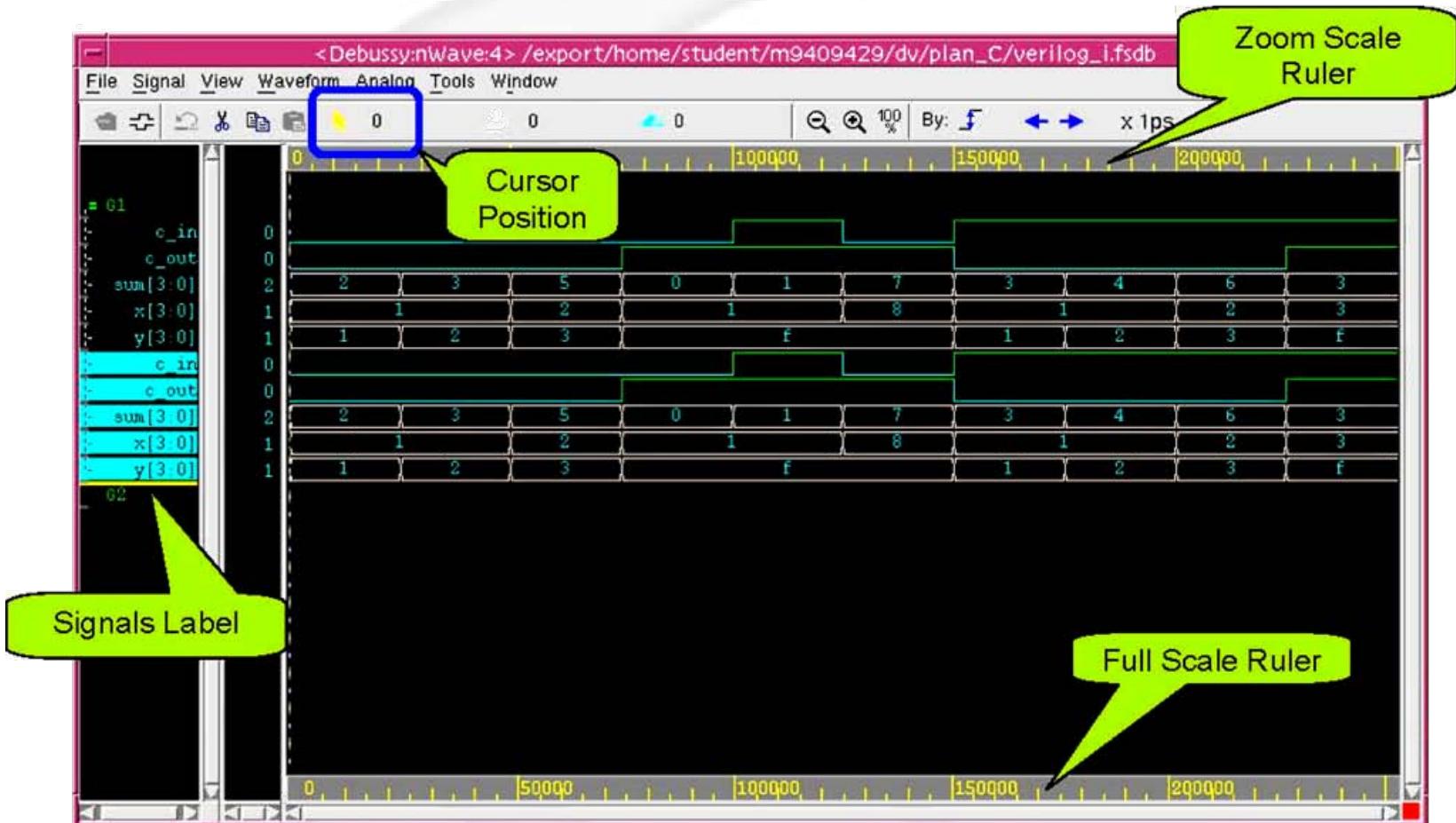
□ 接著按Get Signals icon，此時就會看到有訊號可以讓你選擇了。

→ 如果你看不到訊號，把想觀察的**design(A1)**，直接從**(nTrace) Hierarchy browser**拖進nWave就可以了。



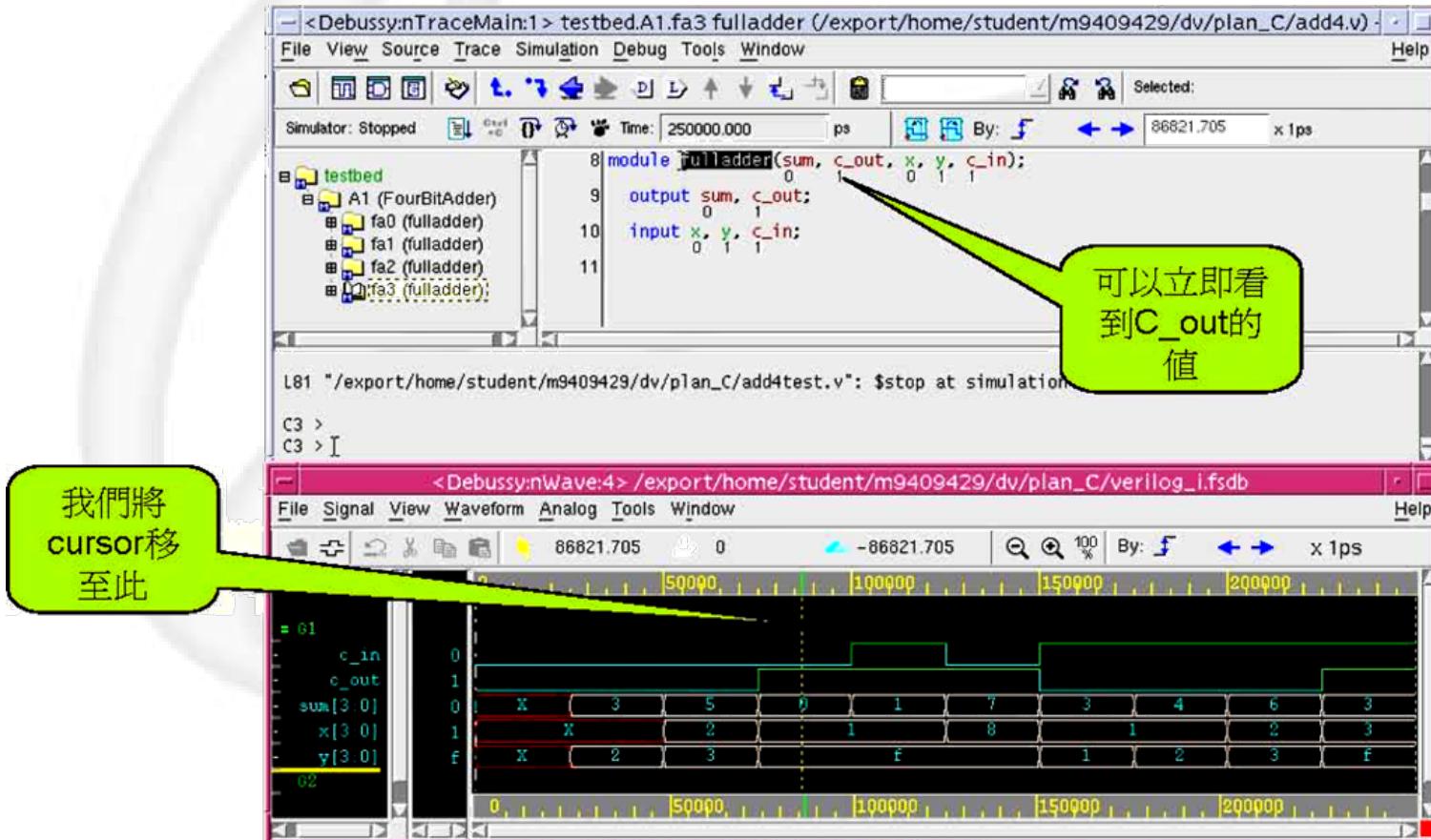
# nTrace :: nWave

□ (nWave) View → Zoom → Zoom All



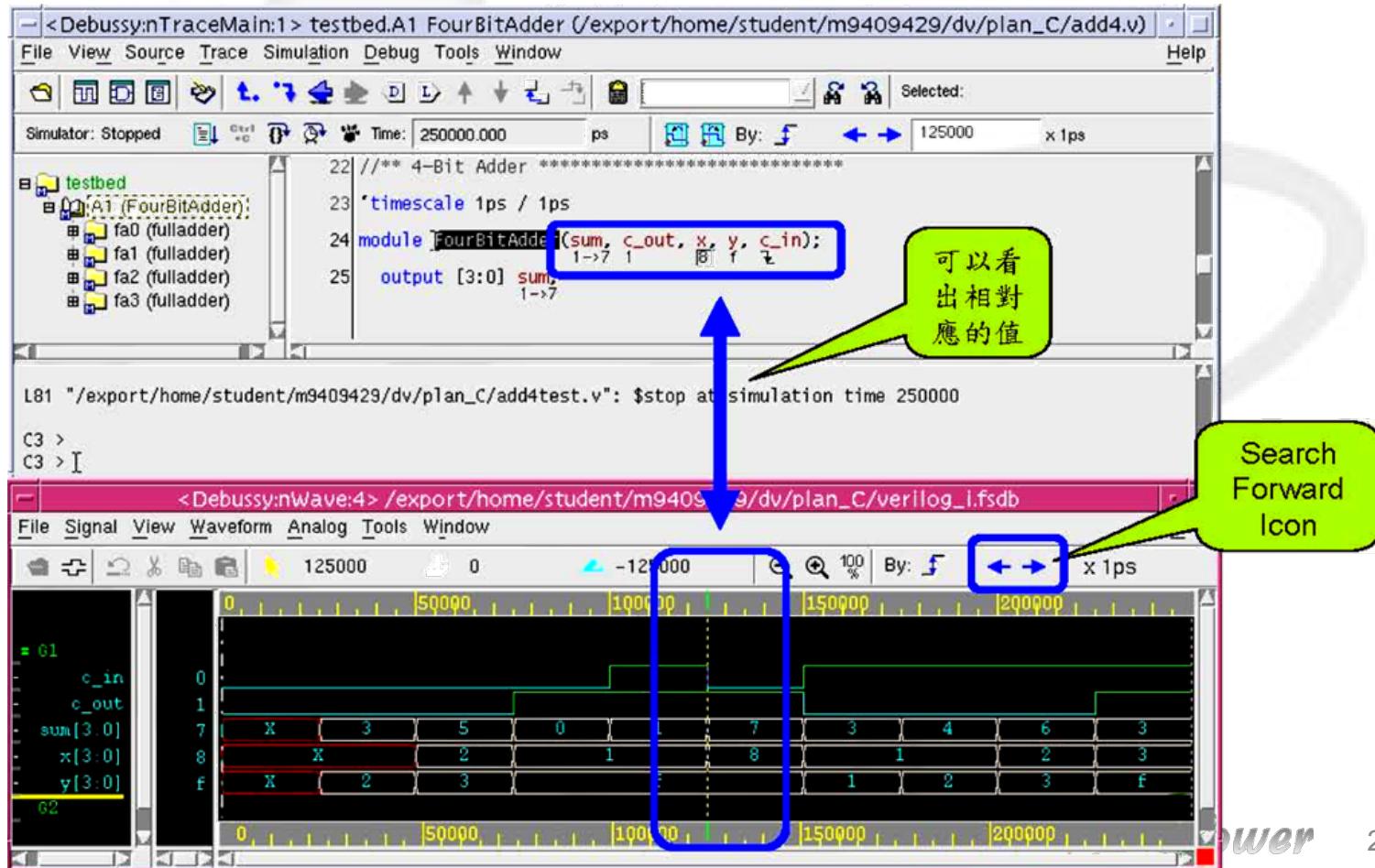
# nTrace :: nWave

- 啟動Active Annotation功能: (nTrace) Source → Active Annotation能夠在nWave選擇訊號觸發緣，同時在nTrace的source code的所有訊號符號下方，直接看到數值得變換。
- 在nWave雙擊想觀察其觸發狀況的訊號(如c\_out).



# nTrace :: nWave

- 從(nTrace) Hierarchy browser 雙擊 A1(FourBitAdder)，按Search Forward icon，看看在nTrace與nWave的變化。





**Thanks for your participation  
and attendance ! !**