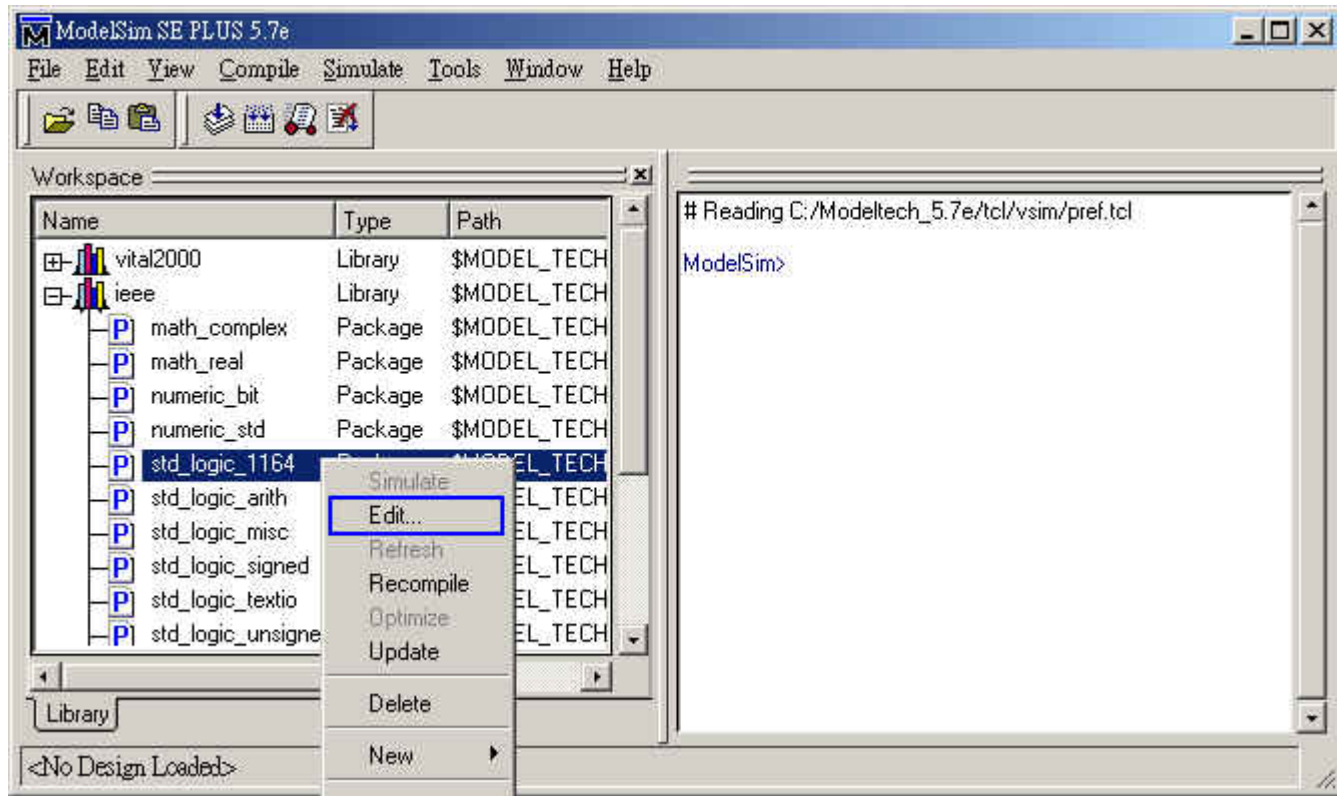


## 1. 建立一个新的 Project

1-1 第一次执行程序时，可以从 **[开始] \ [程序集] \ ModelSim SE \ ModelSim;** 或是执行 ModelSim 在桌面的快捷方式



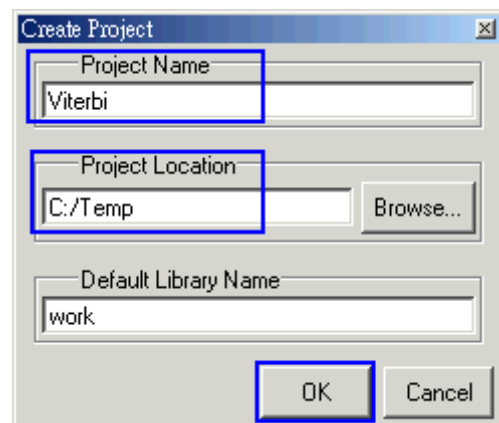
在 Library 标签页中，展开各 Library 就可以看到其下含的所有 Package (for VHDL)，进一步以 Edit 打开，可检视该 Package 与 Package Body 内容

### 1-2 **File \ New \ Project ...** 输入

project name and Location

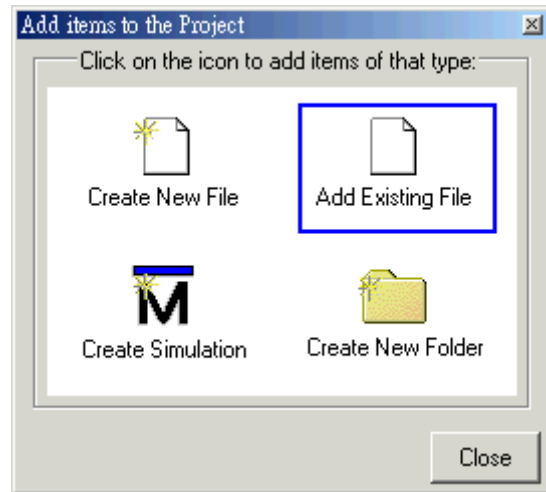
按 OK 键后

- 指定的路径下会产生一个叫“work”的预设子资料夹，还有 Viterbi.cr.mti、Viterbi.mpf 两个档案
- 主操作画面左边的 Workspace 内，在原本的 Library 标签



外，会出现另一个 Project 标签（但此时里面内容是空的）

- 还会蹦出另一个“Add items to the Project”窗口



.mpf 文件储存的是此

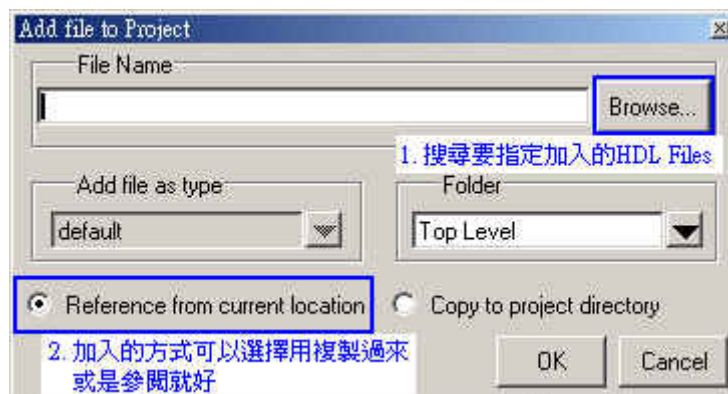
project 的相关数据，下次要开启此 project 就是利用 **File \ Open \ Project...** 开启此 .mpf

若要移除之前建立的

project，请从 **File \ Delete \ Project...** 移除

## 2. 载入 Project 的 HDL source codes

按“Add items to the Project”窗口中的“Add Existing File”（或是从 **File \ Add to Project \ Existing Files ...**）





HDL files 擺放的位置，路徑名稱不能有中文，否則軟件會抓不到 files

关掉“Add items to the Project”窗口，此时的 Project 下出现了 HDL File，一堆问号表示这些档案都还没 compile。


如果要对 project 新增或移除 HDL File:

在 Workspace 内按鼠标右键，选择 **Add to Project \ Existing File...** (新增)

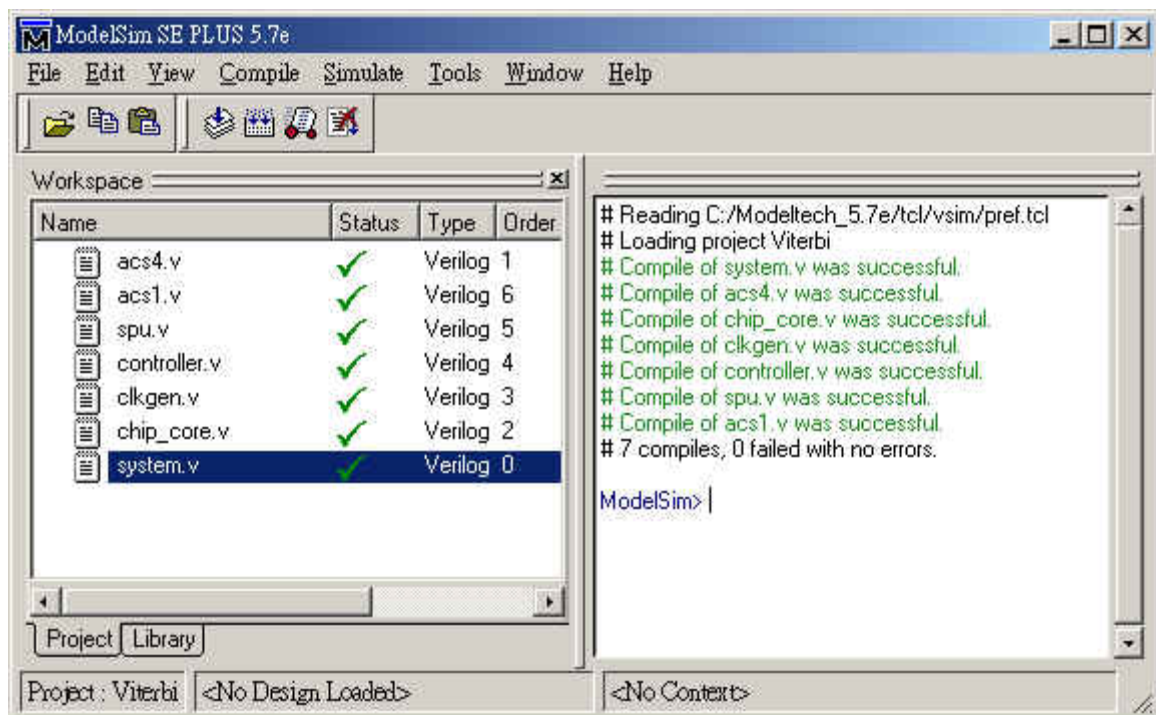
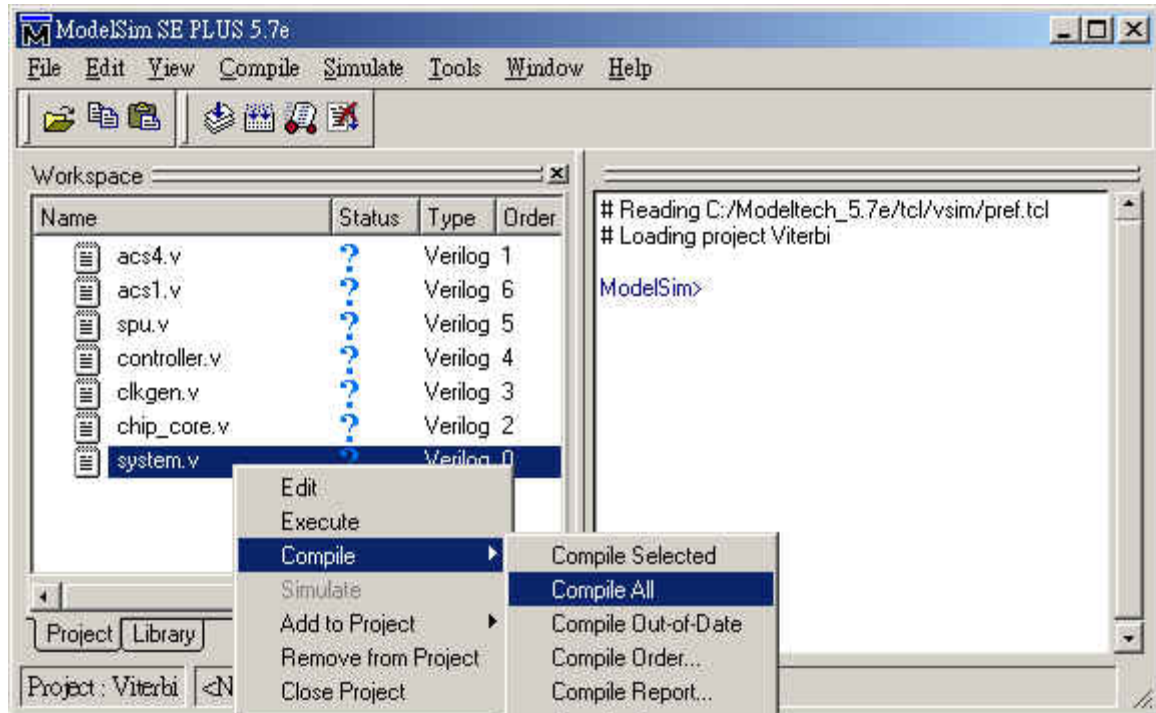
**Remove from**

**Project** (移除)

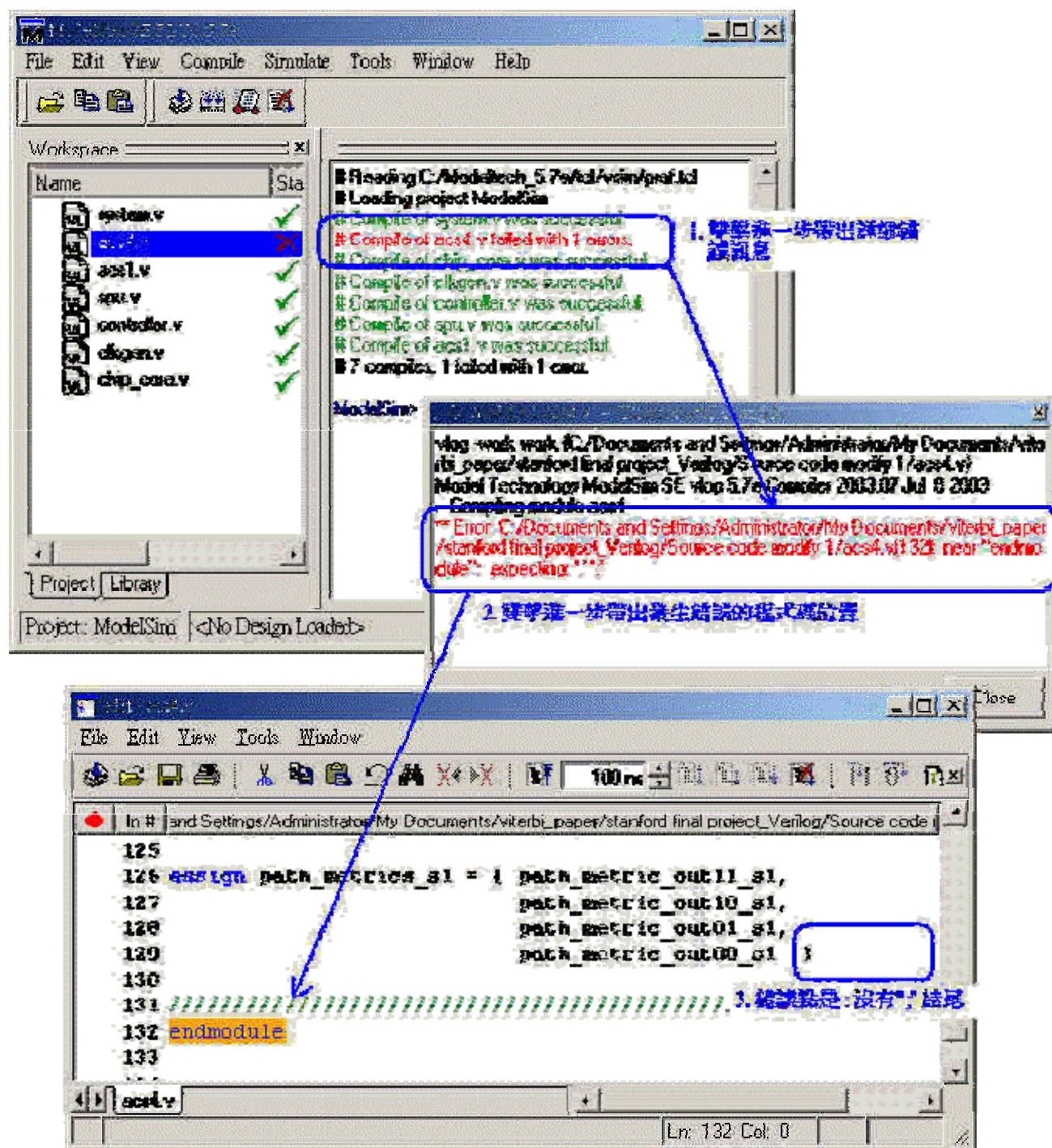
### 3. Compile

在 Project 标签页内，选定任一档案，按鼠标右键选择 **Compile \ Compile All** 或是直接按 icon 

**Compile Out-of-Data** 只重新 compile 有修改过的档案 (比较节省时间，故也较常用)

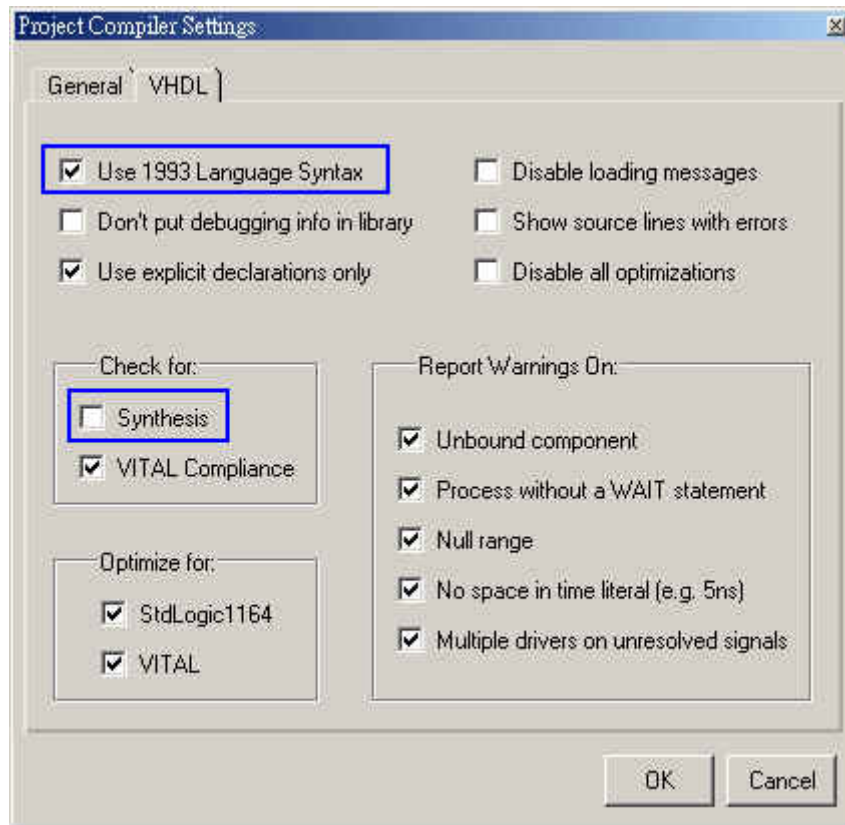


此刻只做 Compile 还没做 Simulate,而 ModelSim 必须要作完 Simulate (Loading) 后才会把所有档案 link 起来。如果 compiler 的结果出现有 errors 或 warnings 的讯息,直接在该讯息上双击,即可进一步带出详细的讯息说明。



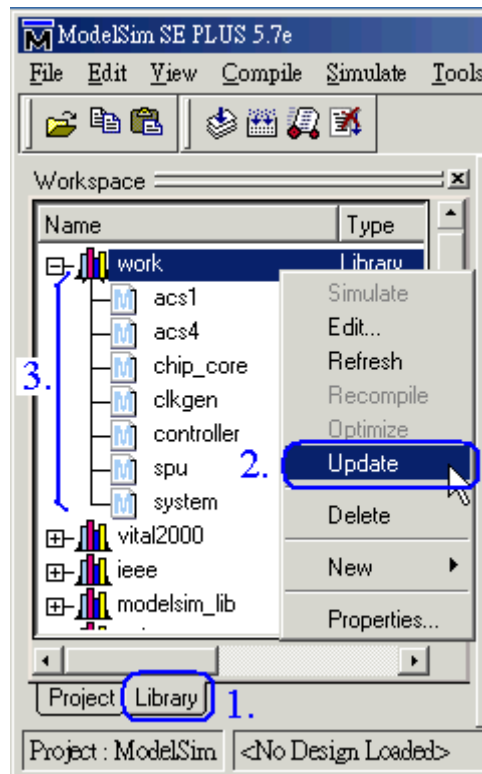
如果您处理的是 VHDL project，那在做 Compile 前，先在 Project 标签页内，选定所有档案，按鼠标右键选择“Properties”做如下设定：





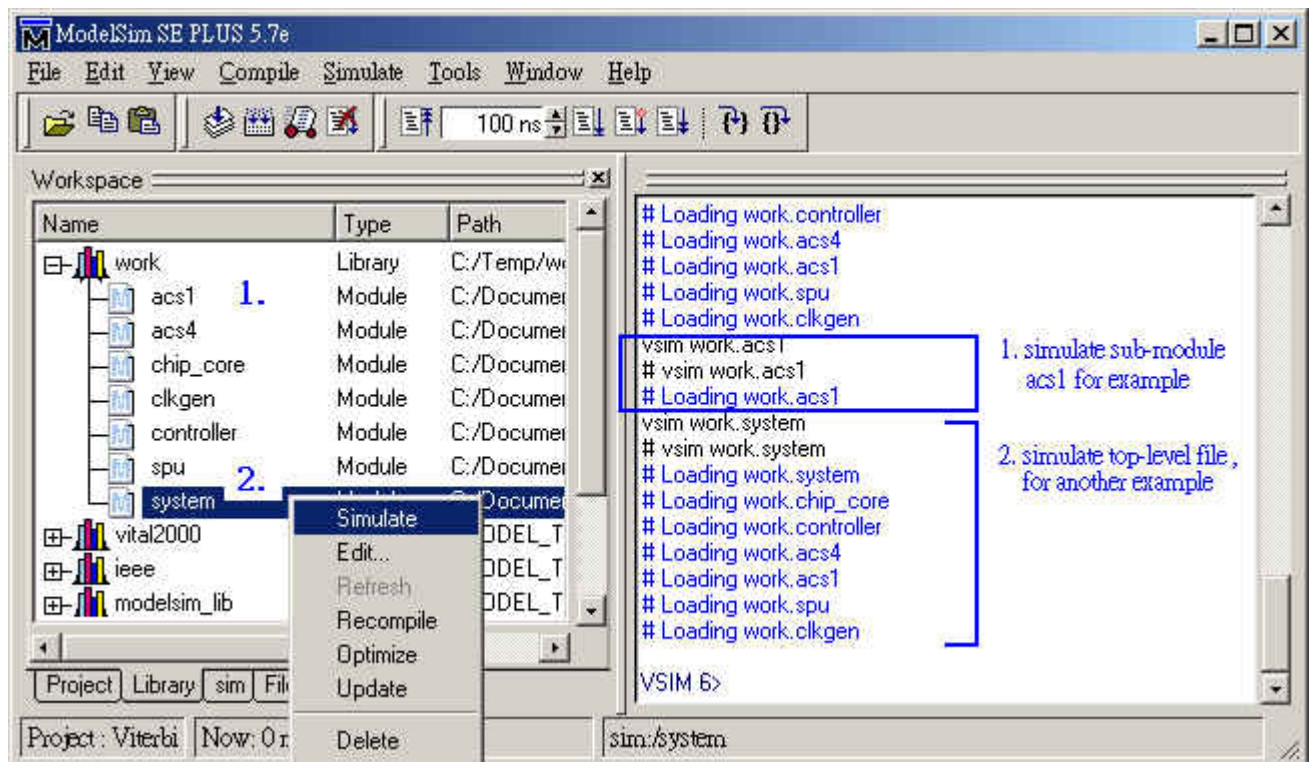
#### 4. Simulate (Loading)

切换到 Library 卷标页，展开“work”目录，其下可以看到此 project 包含的所有档案，档案前的符号“**M**”，表示这些档案的性质是“Module”；如果你写的是 VHDL 程序那档案前所看到的符号会变成“**E**”(Entity) and “**A**”(Architecture)。




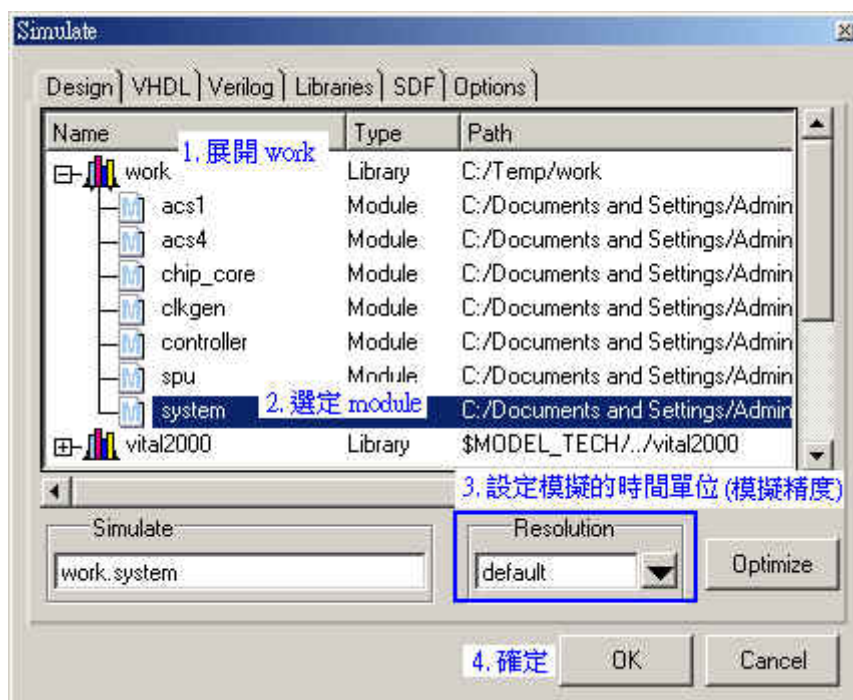
如果 compiler 完, work 目录下仍然看不到东西, 无法展开, 请稍等 5~10 秒 (等程序 update 最新状态), 或是按鼠标右键选择“**Update**”重新整理

4-1 直接以鼠标左键双击要 Simulate 的档案, 或是按鼠标右键选择 Simulate。  
 此处选择要 Simulate (Loading)的档案不一定要是 top-level file, 但如果你选的不是 top-level file, project 中的 sub-circuit 必须一个一个 load, 比较麻烦。



请注意，这里的“Simulate”并没有真的做仿真，只是把程序“Loading”进来，并将 sub-module 彼此间 link 起来

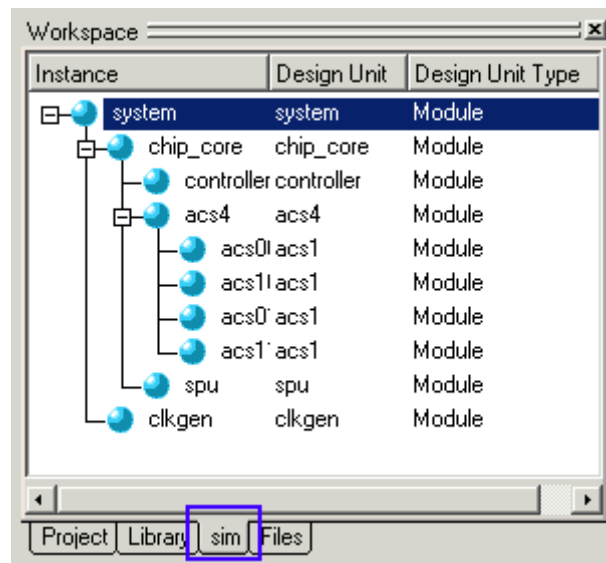
Loading 要 simulate 的档案的动作，也可以按 Simulate icon  来完成





执行以上动作时，如果再附加一个设定步骤：到 Options 标签页下，核选“**Enable source file coverage**”，即可在仿真的程序中，附带帮你检查 testbench 对于 if / case conditional statement 的模拟涵盖率。

执行完 Simulate (Load) 的动作后，程序会自动再跳出两个标签页 Sim 与 Files，此时在 Sim 下可以看到整个 Project 的 Hierarchy 关系



要编辑 HDL file，可以从 Project (显示档案的状态) 或 File 标签页内双击 HDL 档案以开启编辑窗口 (edit window)。sim 标签页内虽然也有列出所有 HDL 档案，但双击 HDL 档案开启的是来源窗口 (source window)，不能 edit。

档案编辑 (修改) 完成 (存盘) 后，回到 Project 标签页内看该档案的状态时，会发现它变成未 compile 的问号，要重新对它做 compile，把目前的 Simulate 关掉 (**Simulate \ End Simulation...**)，再重新 load 一次，才能对新的程序做仿真。

若要查看此 project 架构，必须切换到 Sim 标签页；不像 VeriLogger Pro 只要 Compiler 完，就可以在同一个 Project 画面下看到整个 project structure / hierarchy。Sim 卷标页另外一个有趣的功能是：在 Sim 标签页内选定哪一个 module，其相对的内容会自动 update 于 signals、source、edit... 等 window (如果你有开启这些窗口的话)。

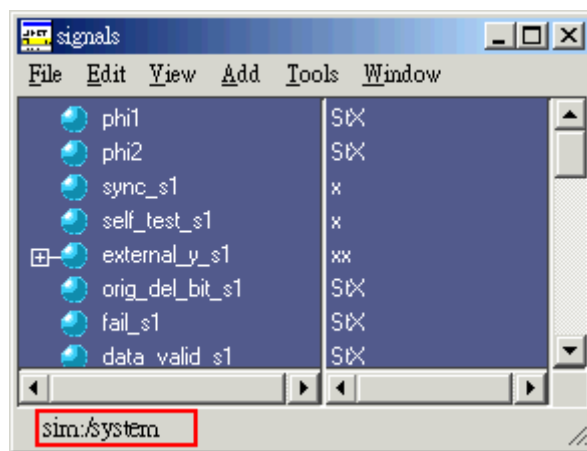
另外，source 与 edit window 两者都可用来看模拟时的某一个变量值或设定断点，但前者不可编辑。

- Project 卷标页用以显示 HDL 档案的状态(也可以开启其编辑窗口)
- Library 标签页用以选定要 Simulate 的档案
- Sim 标签页用以查看 project structure 与开启某 module 的来源窗口
- File 卷标页用以开启某 HDL 档案的编辑窗口

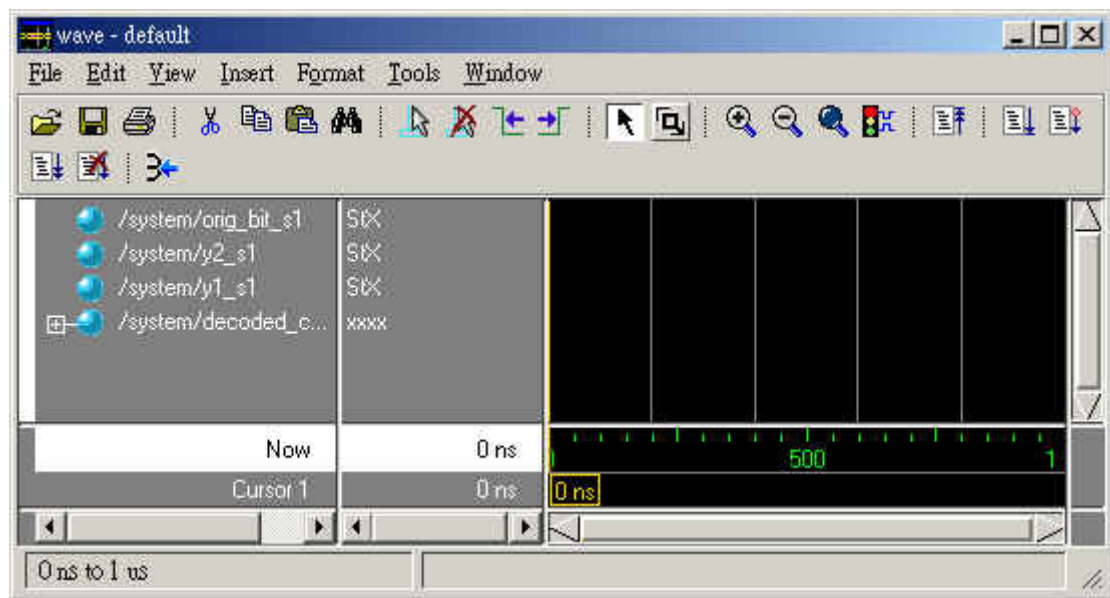
=> 四个功能拆成四个设定页，这是 ModelSim 比较耍宝的地方。

以下，我们接着要把想观察的讯号引出来，然后“Run” simulation ...

4-2 在 Sim 标签页内选定 top-level file (system.v)，打开 signals window: **View \ Signals** 就可以看到 top-level file 的所有 I/O signal。



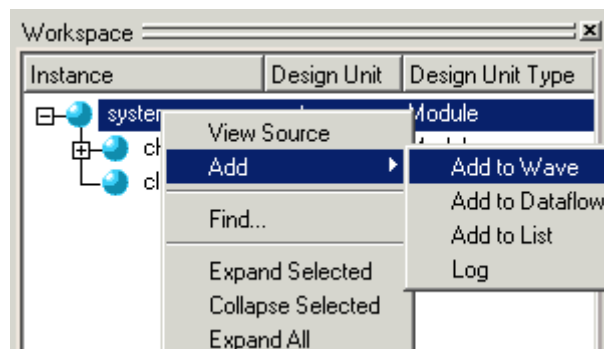
4-3 在 signals window 的左边字段内选定想看的讯号(如 orig\_bit\_s1、y1\_s1、y2\_s1、decoded\_coulmn\_s1)，按鼠标右键 **Add to Wave \ Selected Signals**，会跳出另一个“wave window”并把您想观察的讯号都列出来



直接把 signals window 的讯号拖曳到 wave window 也可以（讯号的排列次序也可以用左键拖曳搬移）。

讯号的排列次序，可以直接用鼠标拖曳调整。

如果要看的是 module / sub-module 所有讯号，4-2~4-3 步骤可以一次完成：




4-4 切回主窗口，**Simulate \ Run \ Run 100ns**  --> 预设 Run


length=100ns（一次跑 100ns；可以更改）

**Simulate \ Run \ Run-All**



--> 跑

到按 “Break”  才暂停

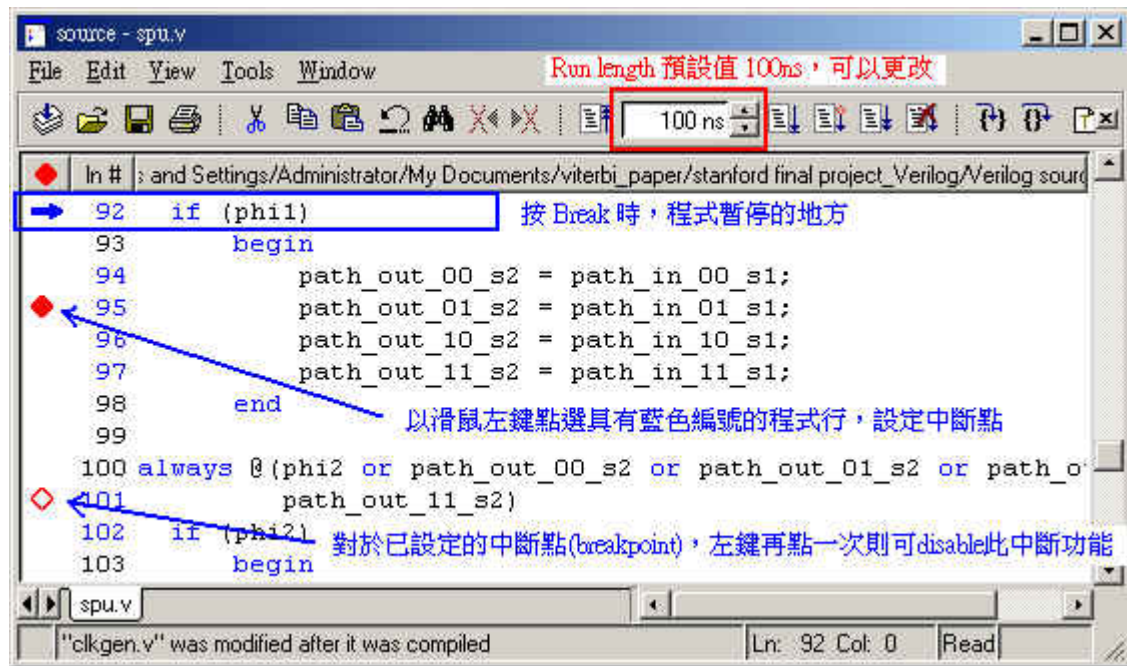
--> 按“Continue Run”  则继续

### **Simulate \ Run \**

**Step**  --> 单步执行程序 (for debugging)

**Simulate \ Run \ Restart**  --> 重

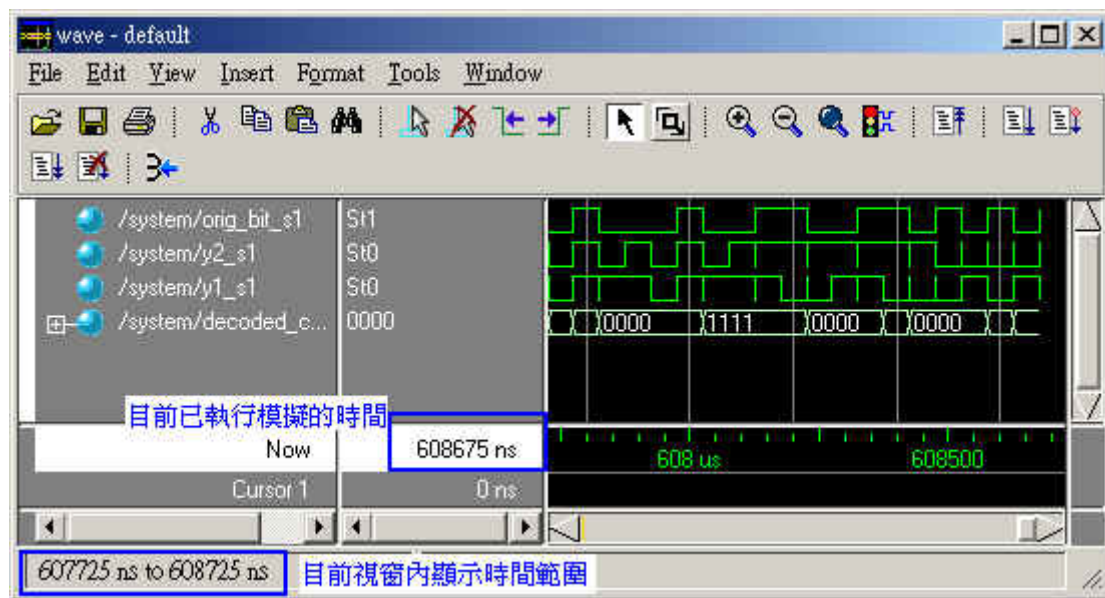
新执行模拟



程序执行暂停或是跑到断点时，要查看任一讯号或变量的数值，方法有三种

- 从 wave window 查看
- 光标指到 source or edit 窗口内的该变量，即会显示该变量当时的数值
- **View \ Variables**

模拟结果如下



如果想要看所有波形的范围：**View \ Zoom \ Zoom Full**

以鼠标左键在波形显示区域点一下，就会出现黄色的垂直坐标线(cursor，参阅 step 5-4)。

直接在讯号波形上双击，会带出 Dataflow window (step 5-3)；如果你是在红色的(unknown)在线双击，还可以进一步的在 Dataflow window 内选定该 unknown signal 的 wire，按鼠标右键选择 **TraceX** 协助你 debug

如果讯号不想以预设的二进制表示，可以在 wave window 内选定该讯号名称，按鼠标右键选择 **Signal Properties...**，然后选择表示方法(如十进制表示 Decimal)。

## 5. Dataflow window (for debugging and tracing)

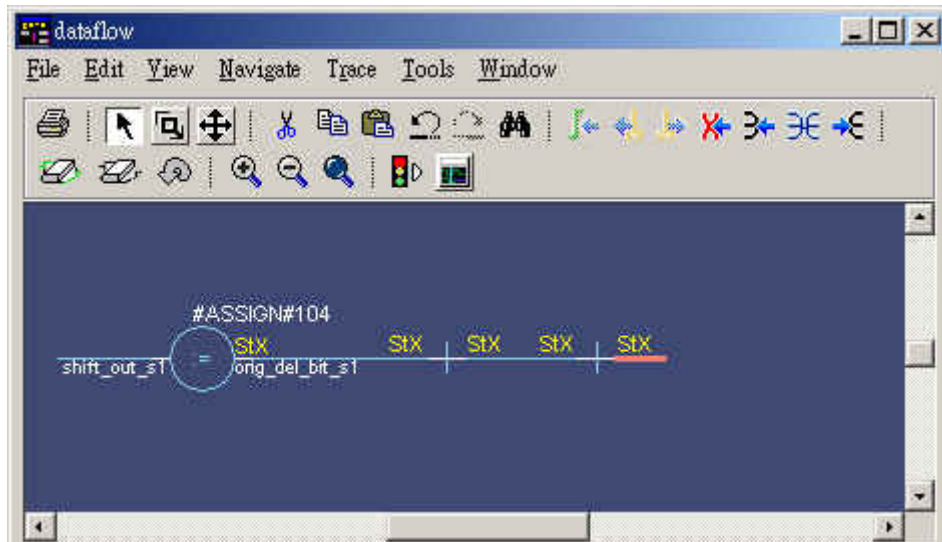
“Dataflow window”可用来检视所设计的电路，其讯号的实际连接情况(Drivers / Receivers)，也就是显示讯号在执行仿真的过程中，经过哪些程序(process)，开启的方法有两个：


- 直接双击 wave window 的讯号波形，会自动跳出 dataflow window，其中显示该讯号的连接情况；以这种方法带出 dataflow window 会在下方同时显示 wave viewer，不需要的话可以从 **View\Show Wave** 取消。

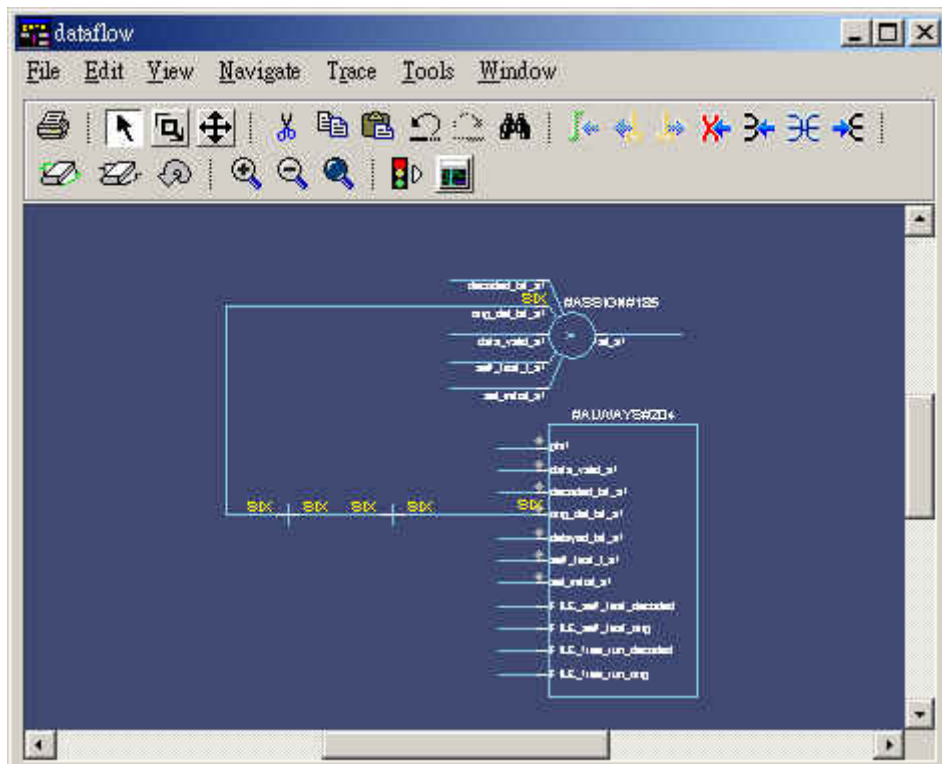




- 从 ModelSim 主窗口中的 **View\Dataflow** 开启窗口，从 signals window 将要看的讯号“拖曳”到 Dataflow window

5-1 假设我们要看的是 orig\_del\_bit\_sl 讯号，此时 Dataflow window 显示如下：



5-2 如果想进一步察看此讯号的 Receiver，选定其输出联机 (red highlight)，按  (expand net to all readers, ie. down-stream circuit) 就会显示如下结果：  
(直接以鼠标左键双击 red highlighted line 也有同样效果)



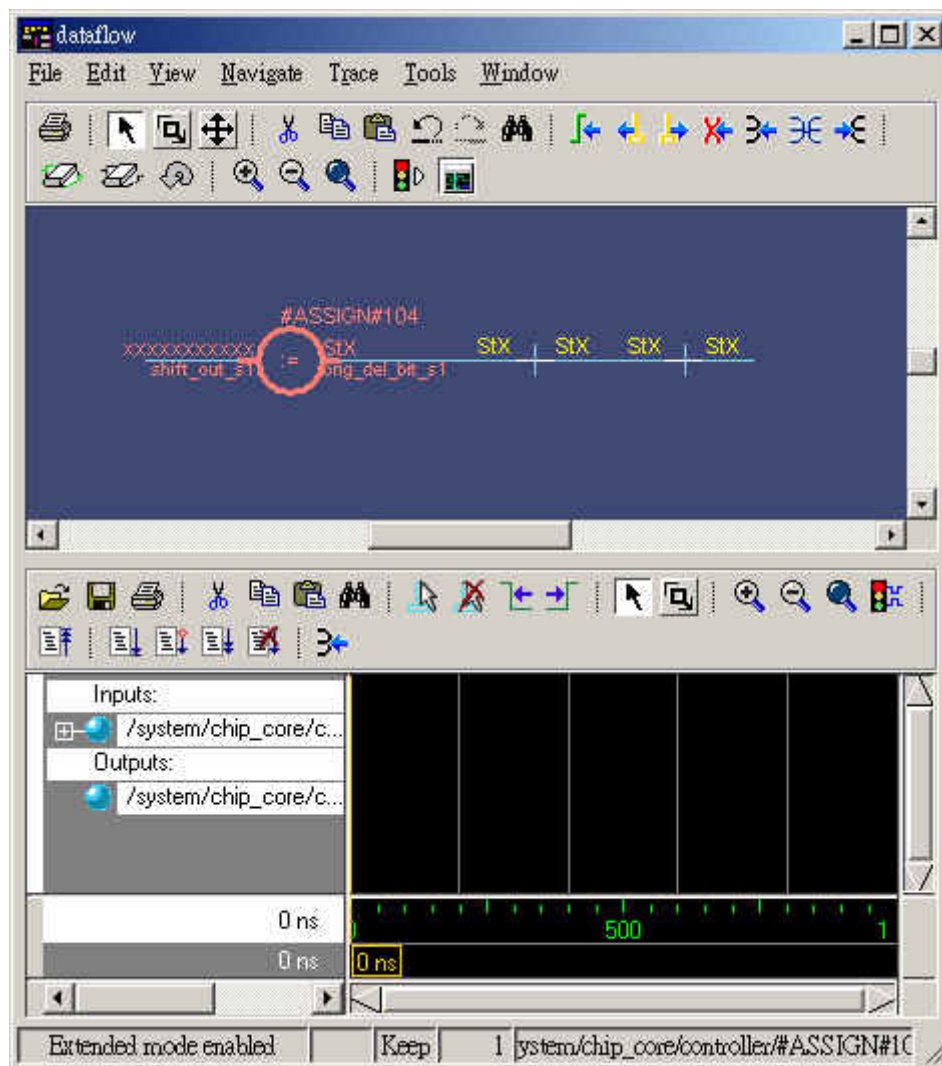
同理，可以使用  查看讯号的 Drivers (ie. up-stream circuit)、使用  查看讯号的 Drivers and Receivers。

 (Erase All): 清除 dataflow window

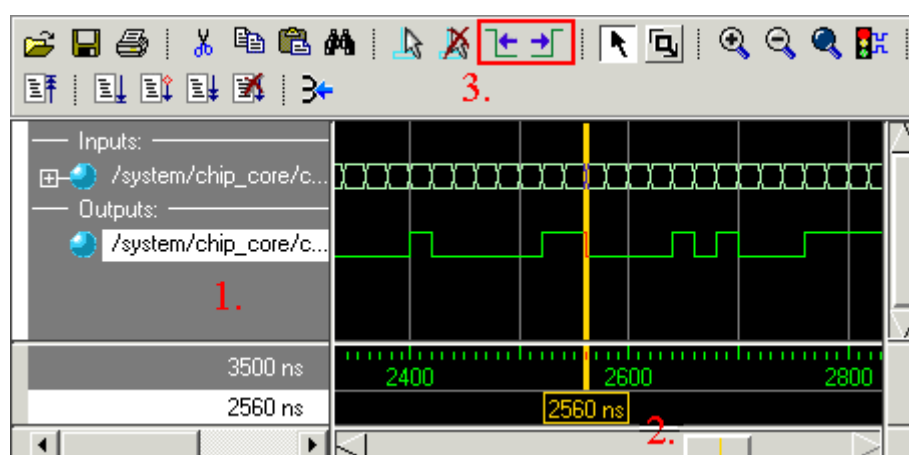
5-3 从 dataflow 窗口的 **View\Show Wave**，打开 embedded wave viewer

选定 dataflow 窗口内的组件#ASSIGN#104，此时会看到 wave 窗口内列出该组件的所有 I/O:

(此时如果有打开 edit or source window，会自动显示出组件#ASSIGN#104在原始码的相对位置)



5-4 执行模拟 3500 ns，结果如下：可以看到此 cell 的输入触发输出的情况



1. 选定想要用 cursor 测量触发时间点的讯号
2. 以鼠标左键在 wave 显示窗口上点一下，cursor(黄色垂直线含时间坐标)会自动出现

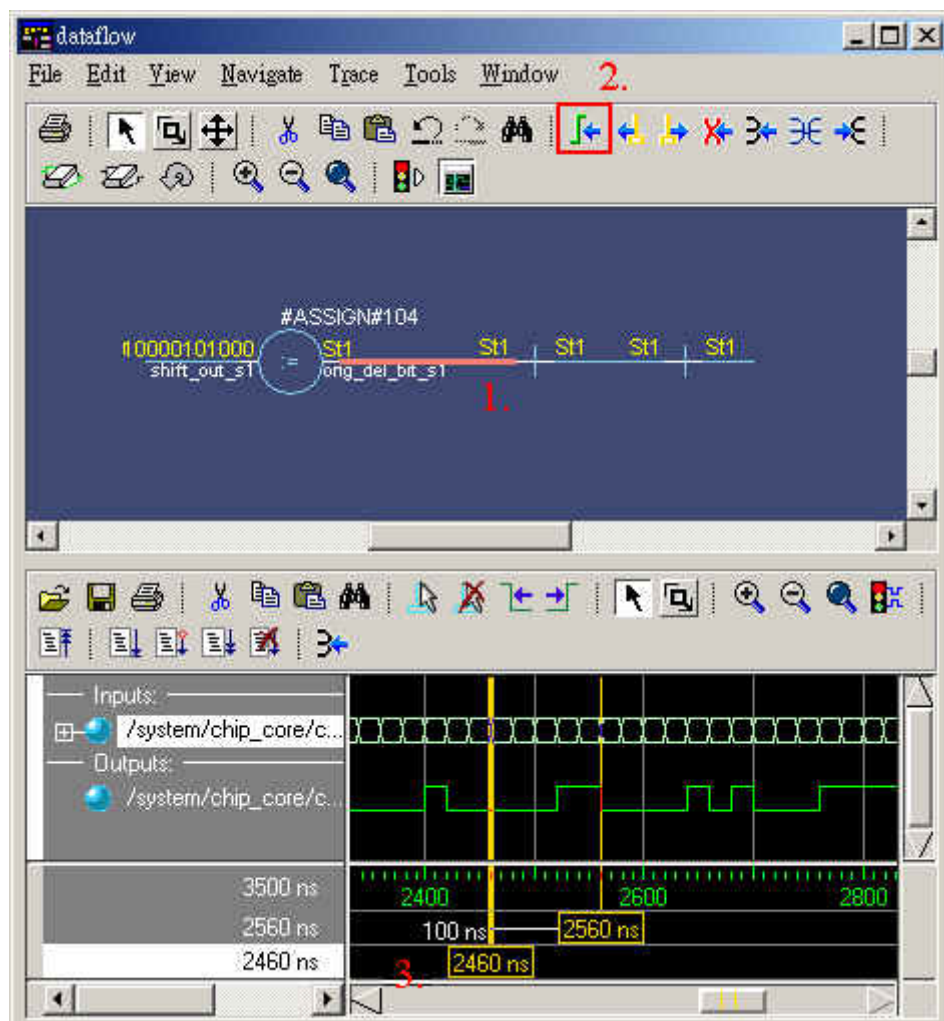
3. 利用“Finder Previous Transition”、“Finder Next Transition”两个 icons，可以让 cursor 自动贴到 trigger edge，以便正确的量测触发时间点。此功能非常便于寻找讯号的触发/转态点(active point)

如果想将某一个 cursor 快速显示在 wave window 内 (将显示波形范围移到该处):

**View \ Cursors \ 选择想跳至的 cursor**

5-5 如何 step by step 追踪 output 被 input 触发的情况呢?

1. 选定想要追踪被触发情况的讯号
2. 按 “Trace input net to event” icons
3. 自动产生另一个 cursor 指到前一个触发 output 的 input 时间点，多按几次 “Trace input net to event”，会看到如下图所示的情况，两个 cursor 之间的时间间距，为白色文字所显示的 100 ns。

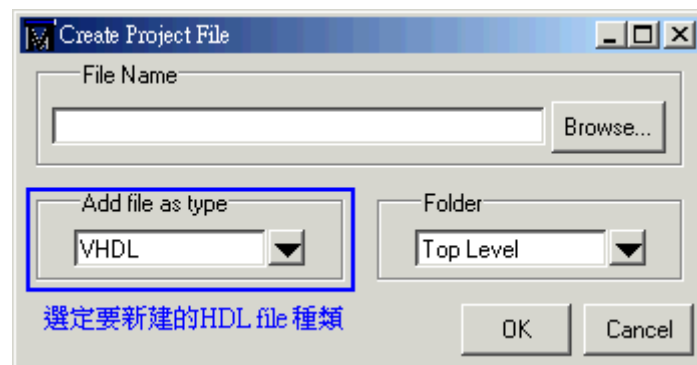


6. 如果只要编辑与 Compile HDL file

**File \ New \ Source \ VHDL or Verilog or Other**

或是

**File \ Add to Project \ New File** (或是在 Project 卷标内，按鼠标右键 Add to Project \ New File)



7. 储存档案并关闭程序

**File \ Save \ ...**

**Simulate \ End Simulation...**

**File \ Close \ Project** (要先关掉 simulation, project 才能关掉)

**File \ Quit** (直接 Quit 最快，可以省下上面关闭 Simulation、Project 的动作)

8. 重新开启 Project

**File \ Open \ Project...** (选择.mpf 檔)

9. 进一步讯息，请参考

**Help \ SE PDF Documentation \ Tutorials** 在线使用手册

一些值得进一步参阅的功能：creating and viewing datasets、performance analyzer、code coverage ...

其它用途： [\(以下这些做法，必须在工作站下执行，PC 环境下产生的档案不能用\)](#)



- 如何用 ModelSim 产生.vcd 檔 (Value Change Dump)

在 testbench 内加入以下这段程序，然后执行 ModelSim 从 compile -> Simulate -> Run -All，关闭 ModelSim 后，就会在工作目录下看见“file\_name.vcd”。

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

- 如何用 ModelSim 产生.fsdb 檔

在 testbench 内加入以下这段程序，然后执行 ModelSim 从 compile -> Simulate -> Run -All，关闭 ModelSim 后，就会在工作目录下看见“file\_name.fsdb”。

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

要用 ModelSim 产生.fsdb 的条件是：跑 ModelSim 的环境下必须有安装 Debussy，否则 ModelSim 会说它看不懂“fsdbDumpfile”这个指令

- 如何用 ModelSim 产生.vec 檔 (vector file)

在 testbench 内加入以下这段程序，然后执行 ModelSim 从 compile -> Simulate -> Run -All，关闭 ModelSim 后，就会在工作目录下看见“add4.vec”。

```
integer openfile;

initial
begin
    // output to "add4.vec" and standard display
    openfile = $fopen("add4.vec") | 1;
    // generator the header of the vector file
    $fdisplay(openfile,"type vec");
    $fdisplay(openfile,"signal sum c_out x y c_in");
    $fdisplay(openfile,"redix 1111 1 1111 1111 1");
    $fdisplay(openfile,"10      0  0 1    1  1");
    // output all the information of node transition in vector file
    $fmonitor(openfile,"%d %b %b %b %b %b", $time,sum,c_out,x,y,c_in);
end
```

您必须依自己的需要，在 header information 那儿稍做修改，改成你 design 的 I/Os(此处所列是一个 4-bit full-adder 的 vector 范例)。这是目前我所知道产生 vector file 最快的方法，否则你就必须先从 testbench.v 转成 file.vcd 再转成 file.vec。[\(参阅 NanoSim 教学 Step 1 ~ 2\)](#)

系统任务(system task)命令\$monitor, \$fmonitor, \$display, \$fdisplay 的用法，请自行参阅 Verilog 书籍[2]sec. 3-3, sec. 9-5