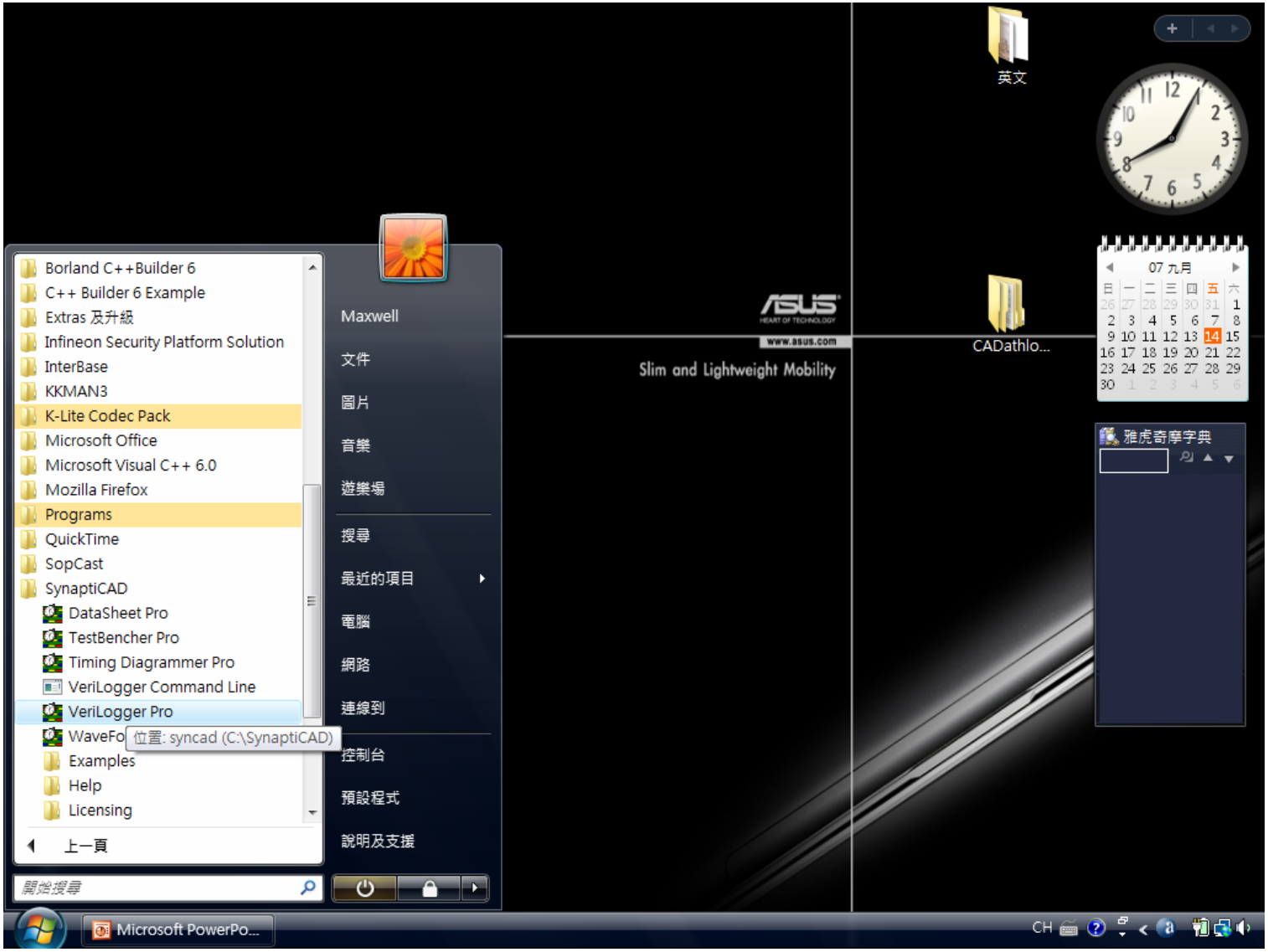

# Verilog簡介

傅仁弘
電機系碩士班一年級
E-mail:965201011@cc.ncu.edu.tw

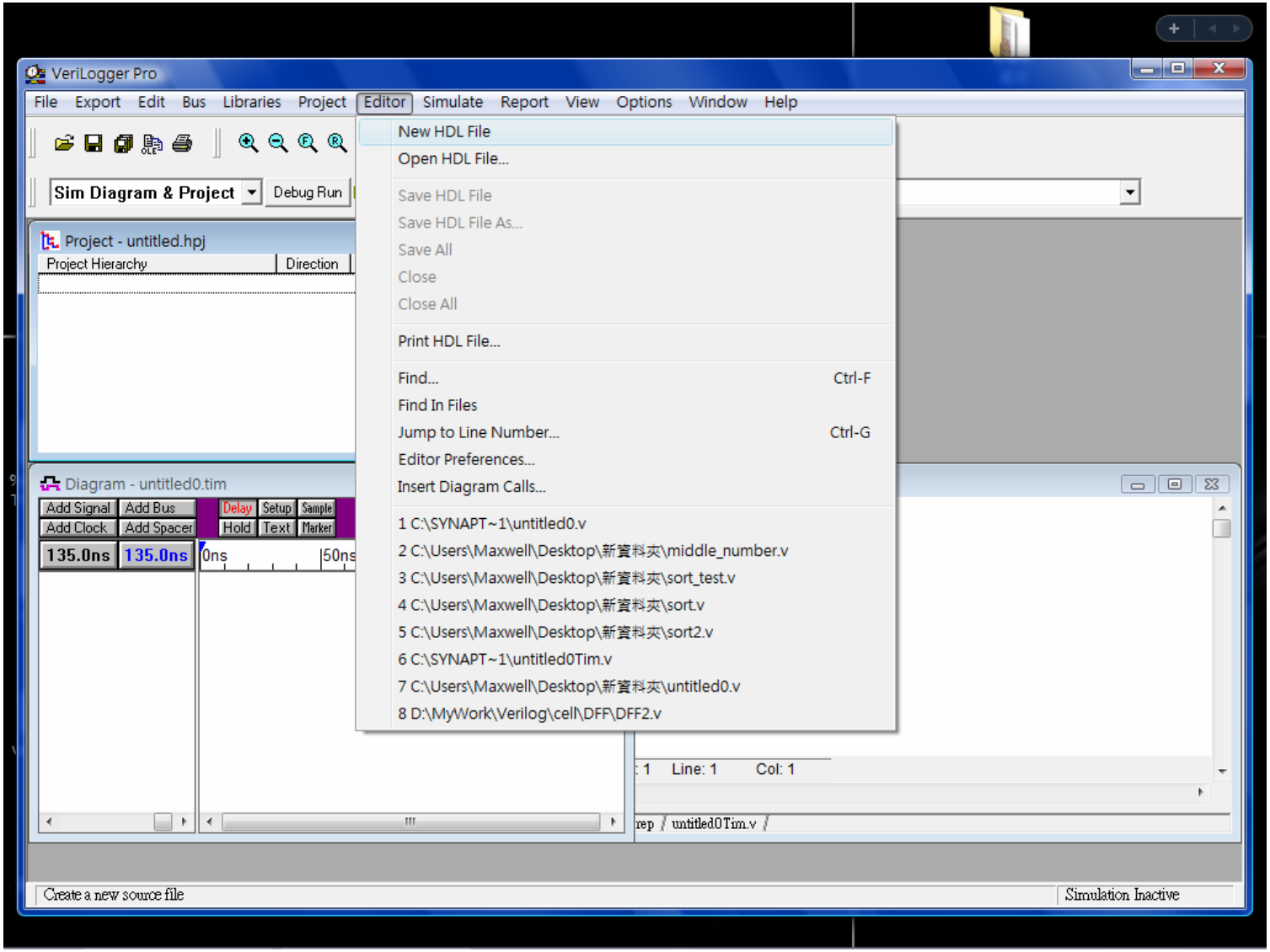
# outline

- Tool簡單操作
- Case1: full adder
- Case2: Sequence Recognizer

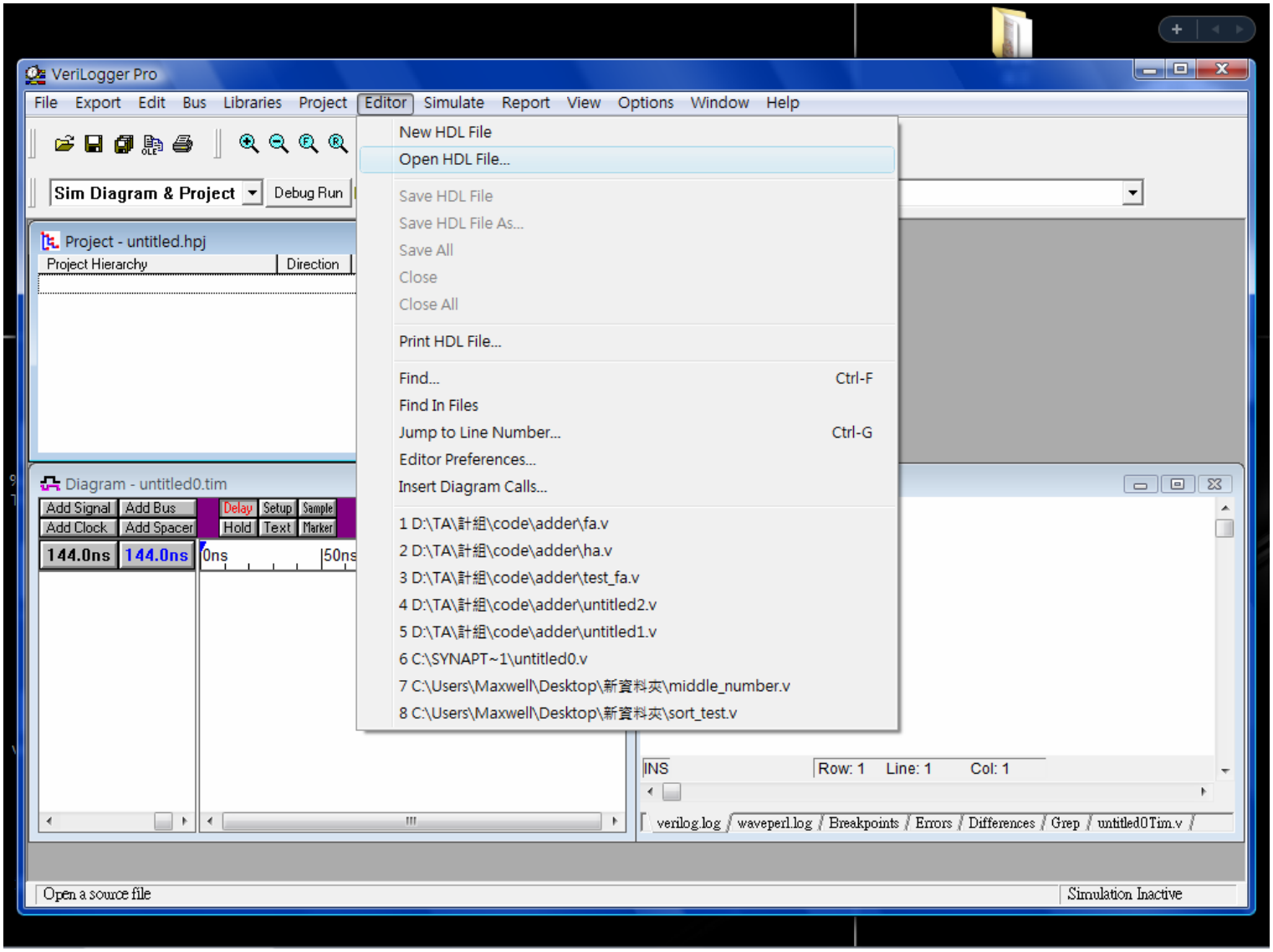
# 執行VeriLogger Pro



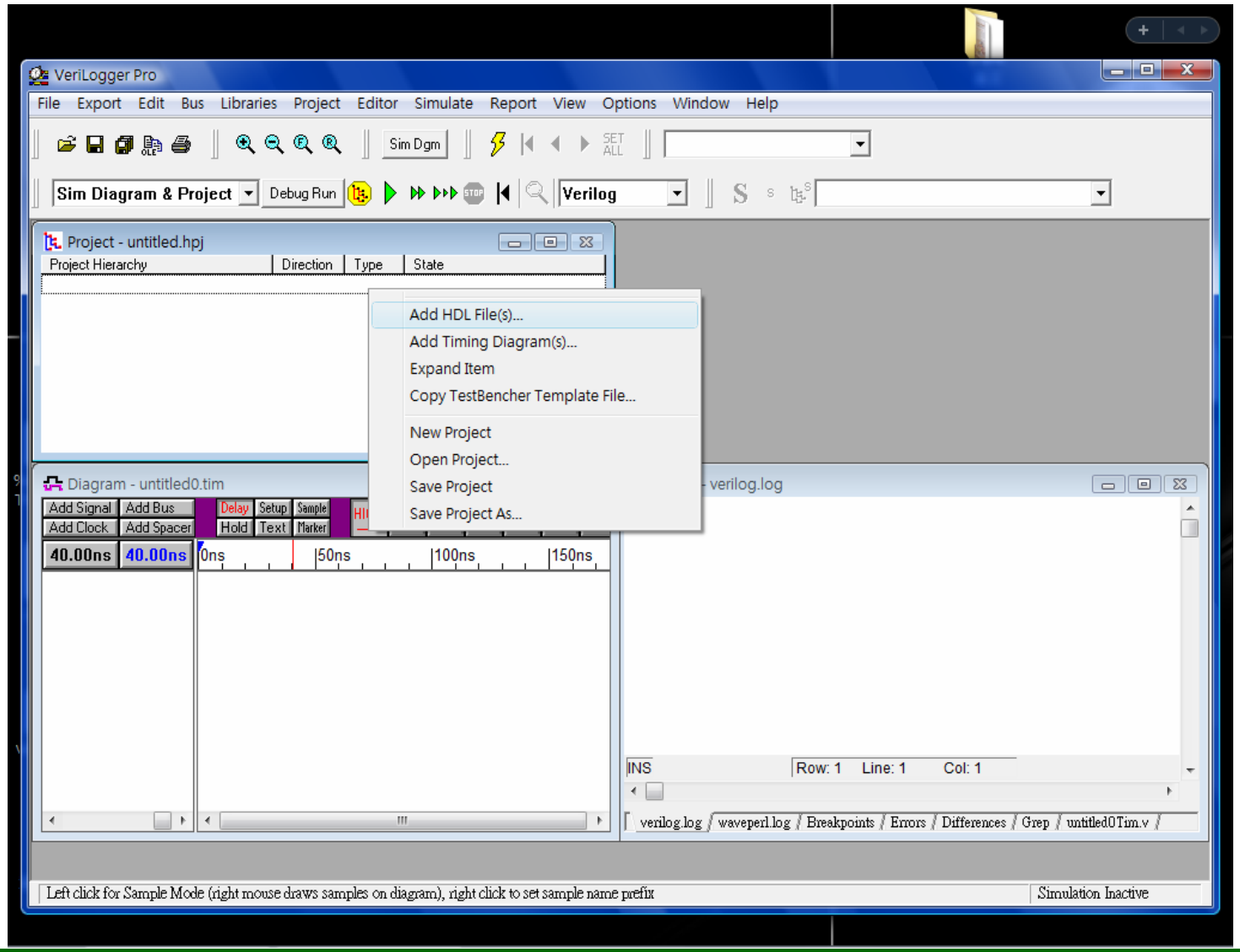
# 開新檔案


VeriLogger Pro
File Export Edit Bus Libraries Project Editor Simulate Report View Options Window Help
Sim Diagram & Project
Debug Run
Project - untitled.hpj
Project Hierarchy
Direction
Diagram - untitled0.tim
Add Signal
Add Bus
Add Clock
Add Spacer
Delay
Setup
Sample
Hold
Text
Marker
135.0ns
135.0ns
0ns
50ns
New HDL File
Open HDL File...
Save HDL File
Save HDL File As...
Save All
Close
Close All
Print HDL File...
Find... Ctrl-F
Find In Files
Jump to Line Number... Ctrl-G
Editor Preferences...
Insert Diagram Calls...
1 C:\SYNAPT~1\untitled0.v
2 C:\Users\Maxwell\Desktop\新資料夾\middle_number.v
3 C:\Users\Maxwell\Desktop\新資料夾\sort_test.v
4 C:\Users\Maxwell\Desktop\新資料夾\sort.v
5 C:\Users\Maxwell\Desktop\新資料夾\sort2.v
6 C:\SYNAPT~1\untitled0Tim.v
7 C:\Users\Maxwell\Desktop\新資料夾\untitled0.v
8 D:\MyWork\Verilog\cell\DFF\DFF2.v
Line: 1
Col: 1
untitled0Tim.v
Create a new source file
Simulation Inactive

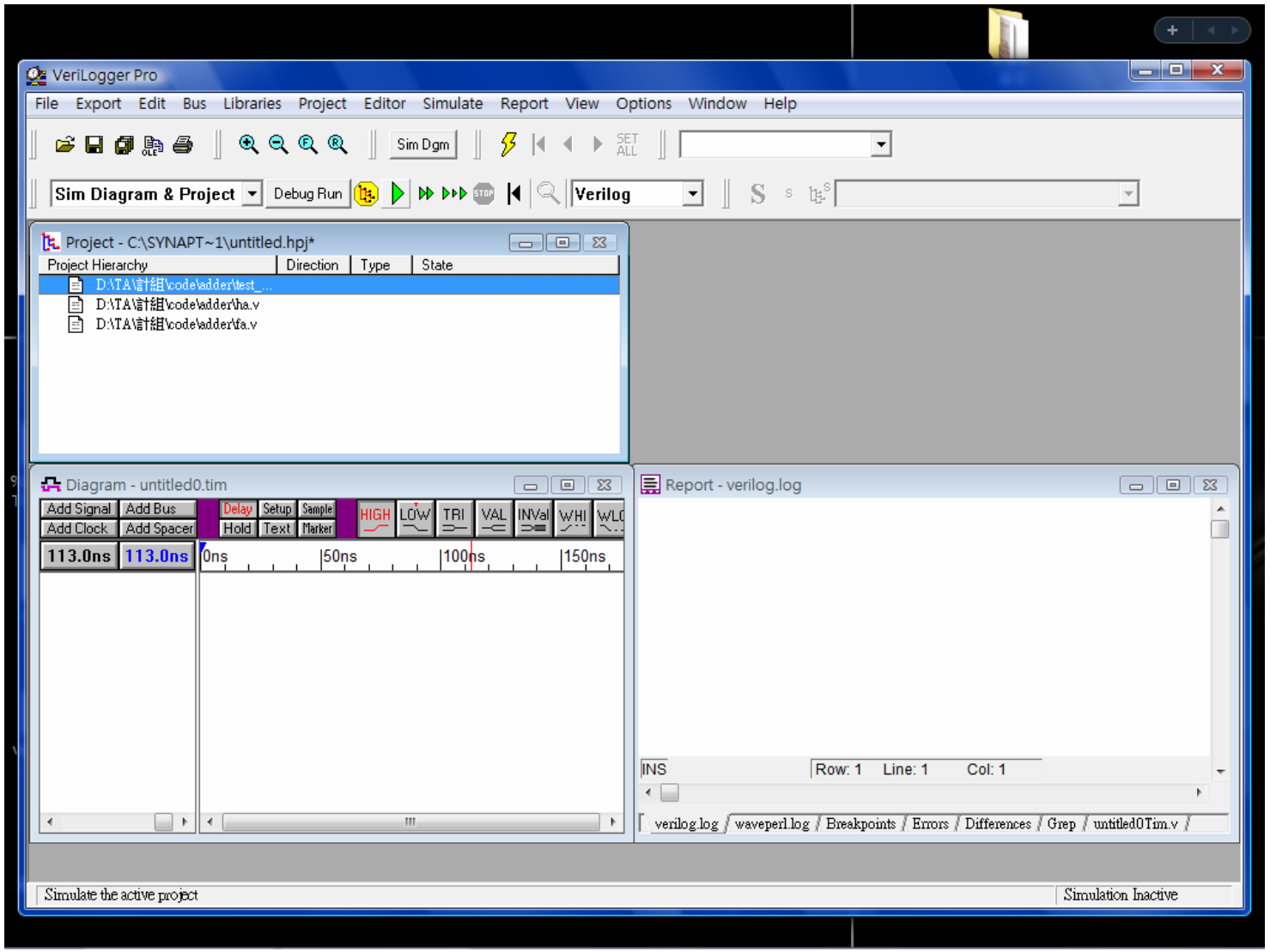
# 讀入已經寫好的code



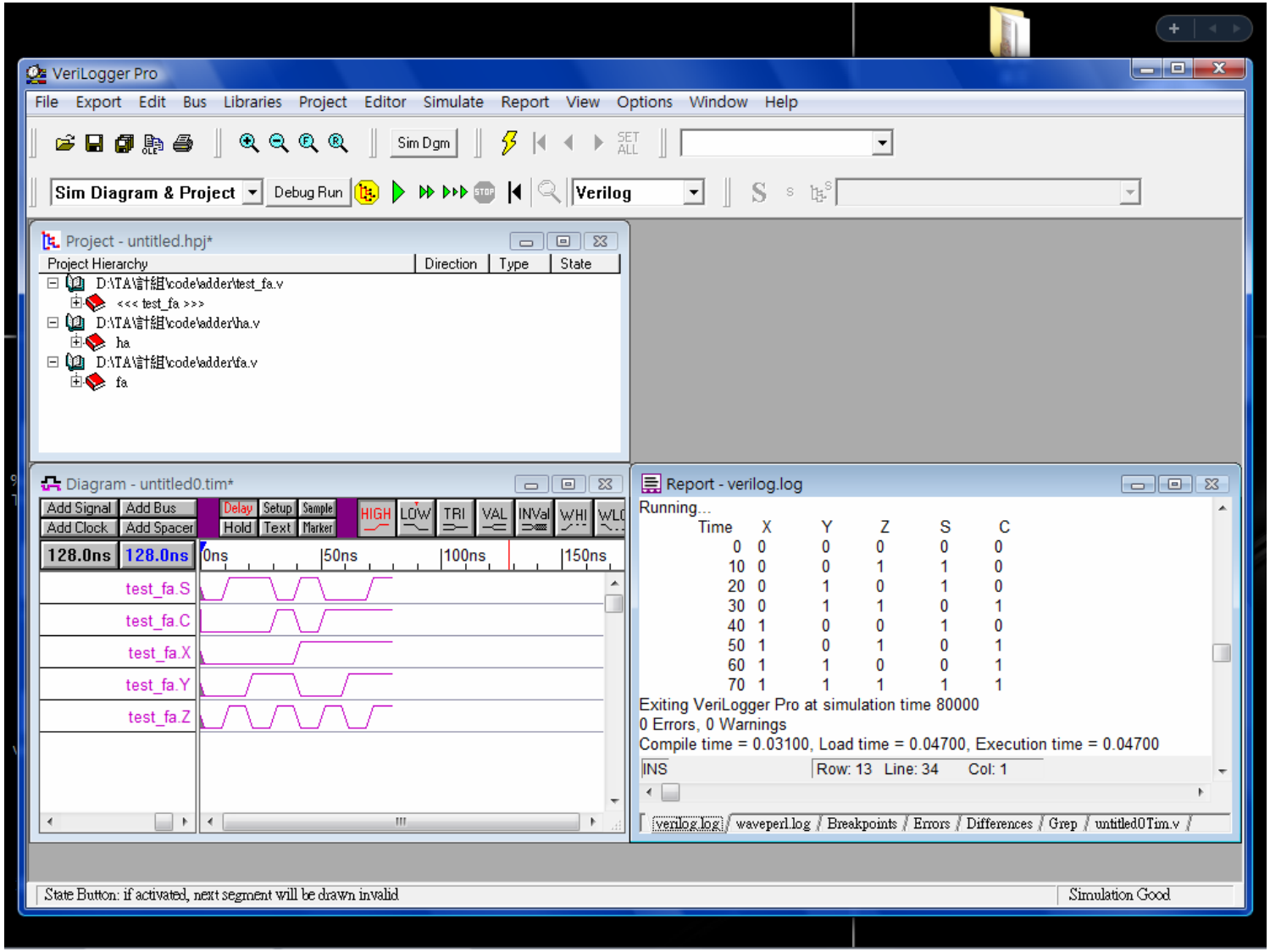
# 將寫好的code加入project



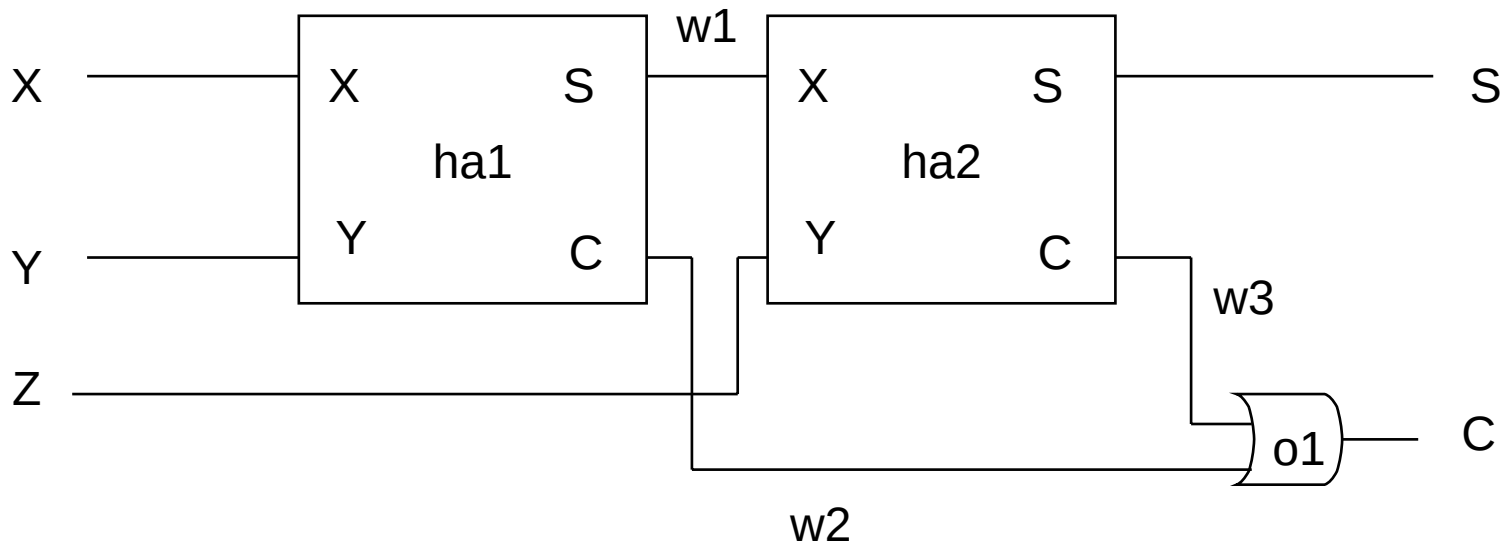
# 按下綠色的simulation



# 模擬結果


VeriLogger Pro
File Export Edit Bus Libraries Project Editor Simulate Report View Options Window Help
Sim Dgm
Sim Diagram & Project
Debug Run
Verilog
Project - untitled.hpj*
Project Hierarchy
Direction
Type
State
D:\TA\計組\code\adder\test_fa.v
<<< test_fa >>>
D:\TA\計組\code\adder\ha.v
ha
D:\TA\計組\code\adder\fa.v
fa
Diagram - untitled0.tim*
Add Signal
Add Bus
Add Clock
Add Spacer
Delay
Setup
Sample
Hold
Text
Marker
HIGH
LOW
TRI
VAL
INVal
WHI
128.0ns
0ns
50ns
100ns
150ns
test_fa.S
test_fa.C
test_fa.X
test_fa.Y
test_fa.Z
Report - verilog.log
Running...
Time X Y Z S C
0 0 0 0 0 0
10 0 0 1 1 0
20 0 1 0 1 0
30 0 1 1 0 1
40 1 0 0 1 0
50 1 0 1 0 1
60 1 1 0 0 1
70 1 1 1 1 1
Exiting VeriLogger Pro at simulation time 80000
0 Errors, 0 Warnings
Compile time = 0.03100, Load time = 0.04700, Execution time = 0.04700
INS
Row: 13 Line: 34 Col: 1
verilog.log
waveperl.log
Breakpoints
Errors
Differences
Grep
untitled0Tim.v
State Button: if activated, next segment will be drawn invalid
Simulation Good

# Full adder (1/4)



# Full adder (2/4)


```
module ha(S,C,X,Y);

    output S;
    output C;
    input X;
    input Y;

    assign S=X^Y;
    assign C=X&Y;

endmodule
```

```
module fa(S,C,X,Y,Z);

    output S;
    output C;
    input X;
    input Y;
    input Z;

    wire w1;
    wire w2;
    wire w3;

    ha h1(.S(w1),.C(w2),.X(X),.Y(Y));
    ha h2(.S(S),.C(w3),.X(w1),.Y(Z));
    or o1(C,w3,w2);

endmodule
```

# Full adder (3/4)


```
module test_fa;
```
→ Test bench不用宣告port

```
wire S;
wire C;
```
→ output

```
reg X;
reg Y;
reg Z;
```
→ input

```
fa set0(.S(S),.C(C),.X(X),.Y(Y),.Z(Z));
```
→ 選擇要測試的 module

```
initial
begin
$display("\t Time \t X \t Y \t Z \t S \t C \t");
$monitor($time," \t %b \t %b \t %b \t %b \t %b ",X,Y,Z,S,C);
end
```

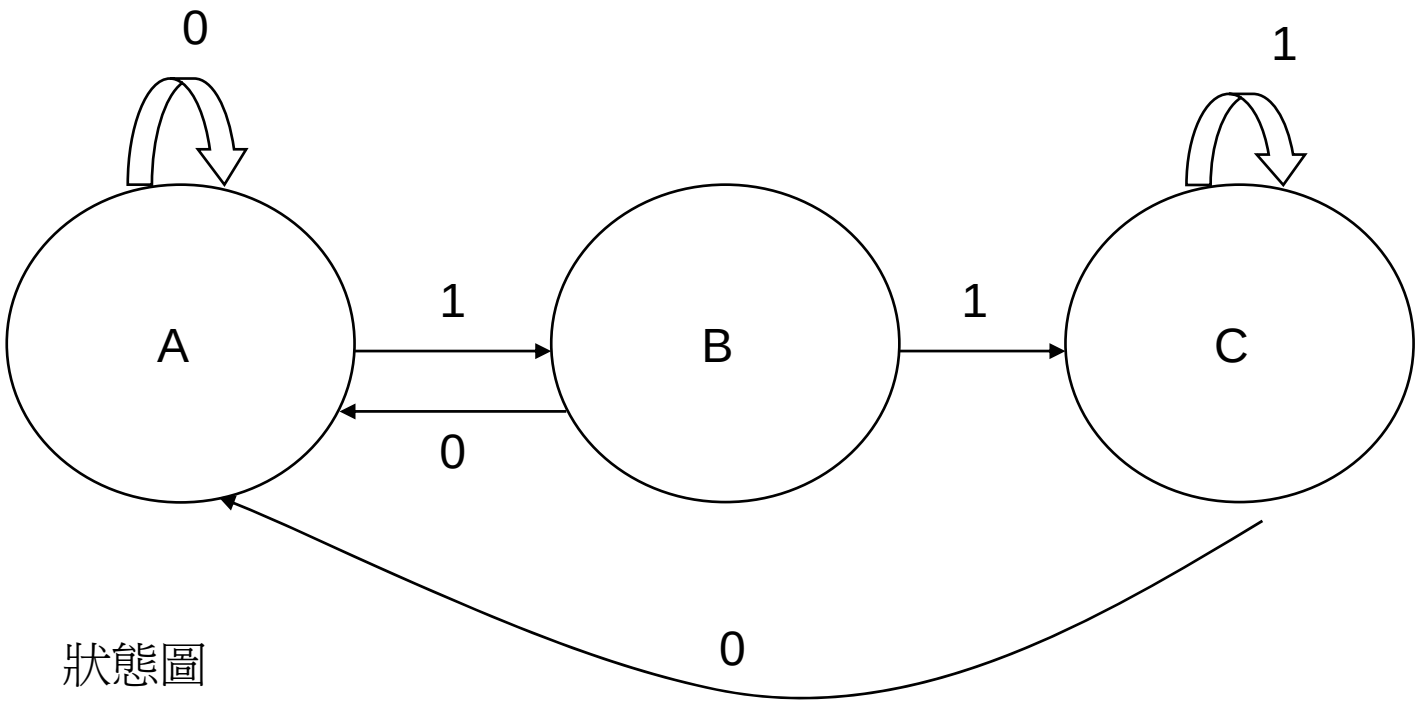
# Full adder (4/4)


```
initial
   begin
   X=0; Y=0; Z=0;
   #10
   X=0; Y=0; Z=1;
   #10
   X=0; Y=1; Z=0;
   #10
   X=0; Y=1; Z=1;
   #10
   X=1; Y=0; Z=0;
   #10
   X=1; Y=0; Z=1;
   #10
   X=1; Y=1; Z=0;
   #10
   X=1; Y=1; Z=1;
   #10
   $finish;
   end

endmodule
```

# Sequence Recognizer(1/2)

- 連續收到兩個1時，output=1; else output=0




狀態圖

# Sequence Recognizer(2/2)


```
module seq_r(Y,CLK,RESET,X);

   output Y;
   input CLK;
   input RESET;
   input X;

   reg [1:0]state;
   reg [1:0]next_state;
   reg Y;

   parameter A=3'b000, B=3'b001, C=3'b010;

   always@(posedge CLK or posedge RESET)
   begin
      if(RESET==1)
         state<=A;
      else
         state<=next_state;
   end

   always@(X or state)
     begin
        case(state)
           A: if(X)
                 next_state<=B;
              else
                 next_state<=A;
           B: if(X)
                 next_state<=C;
              else
                 next_state<=A;
           C: if(X==1)
                 next_state<=C;
              else
                 next_state<=A;
        endcase
     end

   always@(X or state)
   begin
      if(state==C)
         Y=1;
      else
         Y=0;
   end

endmodule
```