

Tutorial on Verilog HDL

HDL

- Hardware Description Languages
 - Widely used in logic design
 - Verilog and VHDL
- Describe hardware using code
 - Document logic functions
 - Simulate logic before building
 - Synthesize code into gates and layout
 - Requires a library of standard cells

Verilog

- Verilog is one of the two major Hardware Description Languages(HDL) used by hardware designers in industry and academia.
- VHDL is another one
- Verilog is easier to learn and use than VHDL
- Verilog HDL allows a hardware designer to describe designs at a high level of abstraction such as at the architectural or behavioral level as well as the lower implementation levels (i.e., gate and switch levels).

Why use Verilog HDL

- Digital systems are highly complex.
- Verilog language provides the digital designer a software platform.
- Verilog allows users to express their design with behavioral constructs.
- A program tool can convert the Verilog program to a description that was used to make a chip, like VLSI.

Taste of Verilog

```
module Add_half ( sum, c_out, a, b );  
  input  a, b;  
  output sum, c_out;  
  wire  c_out_bar;  
  
  xor (sum, a, b);  
  // xor G1(sum, a, b);  
  nand (c_out_bar, a, b);  
  not (c_out, c_out_bar);  
endmodule
```

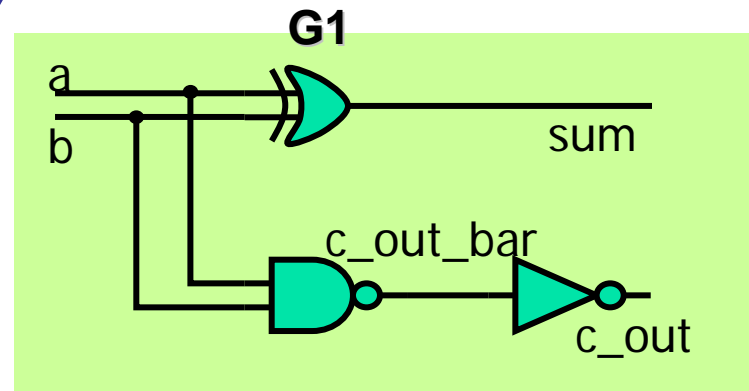
Module name

Module ports

Declaration of port modes

Declaration of internal signal

Instantiation of primitive gates



Verilog keywords

Lexical Convention

- Lexical convention are close to C++.
- Comment

// to the end of the line.

/ to */ across several lines*

- . Keywords are lower case letter.

the language is case sensitive

Lexical Convention

- Numbers are specified in the traditional form or below .
 <size><base format><number>
- Size: contains *decimal* digitals that specify the size of the constant in the number of bits.
- Base format: is the single character ‘ followed by one of the following characters *b(binary), d(decimal), o(octal), h(hex)*.
- Number: legal digital.

Lexical Convention

- Example :

347 // decimal number

4'b101 // 4-bit binary number 0101

2'o12 // 2-bit octal number

5'h87f7 // 5-bit hex number h87f7

2'd83 // 2-bit decimal number

- String in double quotes

“ this is a introduction”

Lexical Convention

- Operator are one, two, or three characters and are used in the expressions.
just like C++.
- Identifier: specified by a letter or underscore followed by more letter or digits, or signs.

Program structure

- Structure

```
module <module name> (< port list>);  
    < declares >  
    < module items >  
endmodule
```

- . Module name

an identifier that uniquely names the module.

- . Port list

a list of input, inout and output ports which are referenced in other modules.

Program structure

- . Declares

 - section specifies data objects as registers, memories and wires as well as procedural constructs such as functions and tasks.*

- . Module items

 - initial constructs*
 - always constructs*
 - assignment*

 -

Test Module structure

- `module <test module name> ;`
- `// Data type declaration. Inputs declared as reg and outputs declared as wire`
- `// Instantiate module (call the module that is going to be tested)`
- `// Apply the stimulus`
- `// Display results`
- `endmodule`

Three Modeling Styles in Verilog

- Structural modeling (Gate-level)
 - **Use predefined or user-defined primitive gates.**
- Dataflow modeling
 - **Use assignment statements (*assign*)**
- Behavioral modeling
 - **Use procedural assignment statements (*always*)**

■ Structural model

```
//structural model of a NAND gate
// program nand2.v
module my_NAND(A, B, F);
    input A, B;
    output F;
    nand G(F, A, B); // first parameter must be output.
endmodule
```

Example of gate NAND

Test bench module test_nand for the nand1.v

```
module test_my_nand;
// Test bench to test nand
reg A, B; wire F;
my_NAND test_my_nand(A, B, F); // instantiate my_NAND.
initial
begin // apply the stimulus, test data
    A = 1'b0; B = 1'b0;
    #100 A = 1'b1; // delay one simulation cycle, then change A=>1.
    #100 B = 1'b1;
    #100 A = 1'b0;
end
initial #500 $finish;
begin // setup monitoring
    //$monitor("Time=%0d a=%b b=%b out1=%b", $time, A, B, F);
    // #500 $finish;
end
endmodule
```

Structural Modeling

//Gate-level description of a 2-to-4-line decoder

//Figure 4-19

module decoder_gl (input A,B,E, output [0:3] D);

wire Anot, Bnot, Enot;

not

n1 (Anot, A),

n2 (Bnot, B),

n3 (Enot, E);

nand

n4 (D[0], Anot, Bnot, Enot),

n5 (D[1], Anot,B, Enot),

n6 (D[2], A, Bnot, Enot),

n7 (D[3], A, B, Enot);

endmodule


```
//Gate-level hierarchical description of 4-bit adder
```

```
// Description of half adder (see Fig 4-5b)
```

```
//module halfadder (S,C,x,y);
```

```
    // input x,y;
```

```
    // output S,C;
```

```
module halfadder (output S,C, input x,y);
```

```
//Instantiate primitive gates
```

```
    xor (S,x,y);
```

```
    and (C,x,y);
```

```
endmodule
```

```
//Description of full adder (see Fig 4-8)
```

```
module fulladder (output S,C, input x,y,z);
```

```
    wire S1,C1,C2; //Outputs of first XOR and two AND gates
```

```
    halfadder HA1 (S1,C1,x,y), HA2 (S,C2,S1,z); //Instantiate the halfadder
```

```
    or g1(C,C2,C1);
```

```
endmodule
```

//Description of 4-bit adder (see Fig 4-9)

module ripple_carry_4bit_adder (output [3:0] S, output C4, input [3:0] A,B, input C0)

// input [3:0] A,B;

//input C0;

//output [3:0] S;

//output C4;

wire C1,C2,C3; //Intermediate carries

//Instantiate the fulladder

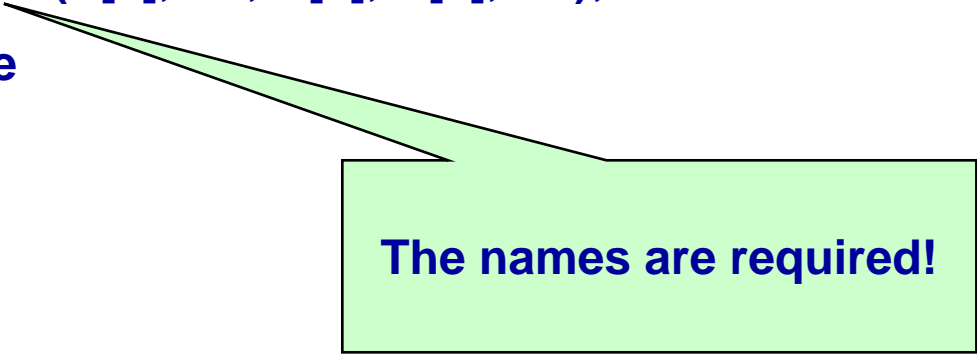
fulladder FA0 (S[0], C1, A[0], B[0], C0),

FA1 (S[1], C2, A[1], B[1], C1),

FA2 (S[2], C3, A[2], B[2], C2),

FA3 (S[3], C4, A[3], B[3], C3);

endmodule



The names are required!

Dataflow Modeling

//HDL Example 4-3

//-----

//Dataflow description of a 2-to-4-line decoder

//See Fig.4-19

**module decoder_df (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

Dataflow Modeling

//HDL Example 4-4

//-----

//Dataflow description of 4-bit adder

module binary_adder (A, B, Cin, SUM, Cout);

input [3:0] A,B;

input Cin;

output [3:0] SUM;

output Cout;

assign {Cout, SUM} = A + B + Cin;

endmodule



concatenation

Binary addition

Dataflow Modeling

//HDL Example 4-5

//-----

//Dataflow description of a 4-bit comparator.

module magcomp (A,B,ALTB,AGTB,AEQB);

input [3:0] A,B;

output ALTB,AGTB,AEQB;

assign ALTB = (A < B),

AGTB = (A > B),

AEQB = (A == B);

endmodule

Dataflow Modeling

//HDL Example 4-6

//-----

//Dataflow description of 2-to-1-line multiplexer

module mux2x1_df (A, B, select, OUT);

input A,B,select;

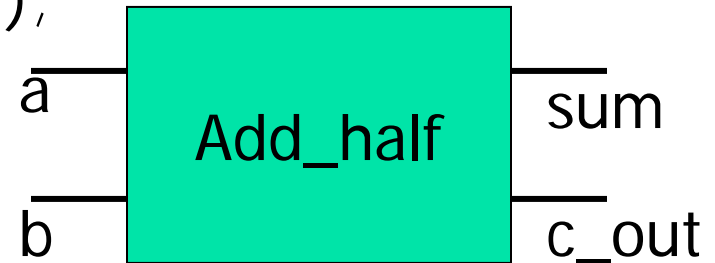
output OUT;

assign OUT = select ? A : B;

endmodule

Behavioral Description

```
module Add_half ( sum, c_out, a, b );  
  input    a, b;  
  output   sum, c_out;  
  reg sum, c_out;  
  always @ ( a or b )  
  begin  
    sum = a ^ b;           // Exclusive or  
    c_out = a & b;         // And  
  end  
endmodule
```



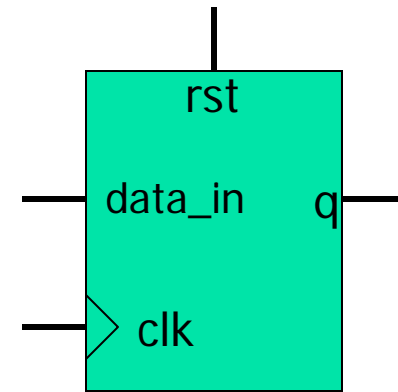
Must be of the
'reg' type

Procedure
assignment
statements

Event control
expression

Example of Flip-flop

```
module Flip_flop ( q, data_in, clk, rst );  
  input      data_in, clk, rst;  
  output    q;  
  reg q;
```



```
  always @ ( posedge clk ) ← Declaration of synchronous behavior
```

```
    begin
```

```
      if ( rst == 1) q = 0; ← Procedural statement
```

```
      else q = data_in; ← Procedural statement
```

```
    end
```

```
endmodule
```


Using Verilogger Pro

- Evaluation Version.
- enter the window of Verilogger
Start → Program → SynaptiCad → Verilogger Pro..

How to build a new project

- Click Menu [Project] → [New Project] → enter the conversation window.
 - Enter the Project Name.
default: untitled.hpj. *.hpj
 - Enter the Project Directory
C:\SynaptiCAD\project\
Or others.
- .Click the [Finish] to close the window.

Other menus of [Project]

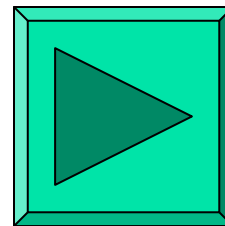
- [Open Project]
- [Close Project]
- [Save Project]
- [Save Project as]
- [Add User Source Files]
 - all the user source used by this project.
- Project setting
- Print Project Hierarchy

Verilogger Editor

- Use the Verilogger Editor to build a program.
- In the Verilogger Window:
 - click [Editor] → [New HDL file] → pop up a editor window for you.
- . Others Menu in the [Editor] same as Menu[Project]

Example of gate NAND

- Save the HDL files as nand1.v in menu [Editor] → [Save HDL File As] and save another HDL file as test-nand1.v
- Attach these two HDL files to a new project test.hpj in [project window]
- Run the simulation program
run/resume simulation button or in the [simulate].



How to build a new project?

Debug Run [Icons] SET ALL [Dropdown] \$settrace: [Dropdown]

Project - untitled0.hpi

Project Hierarchy

- Simulated Model
 - Stimulus & Results (StimulusAndResults.btim)
- User Source Files

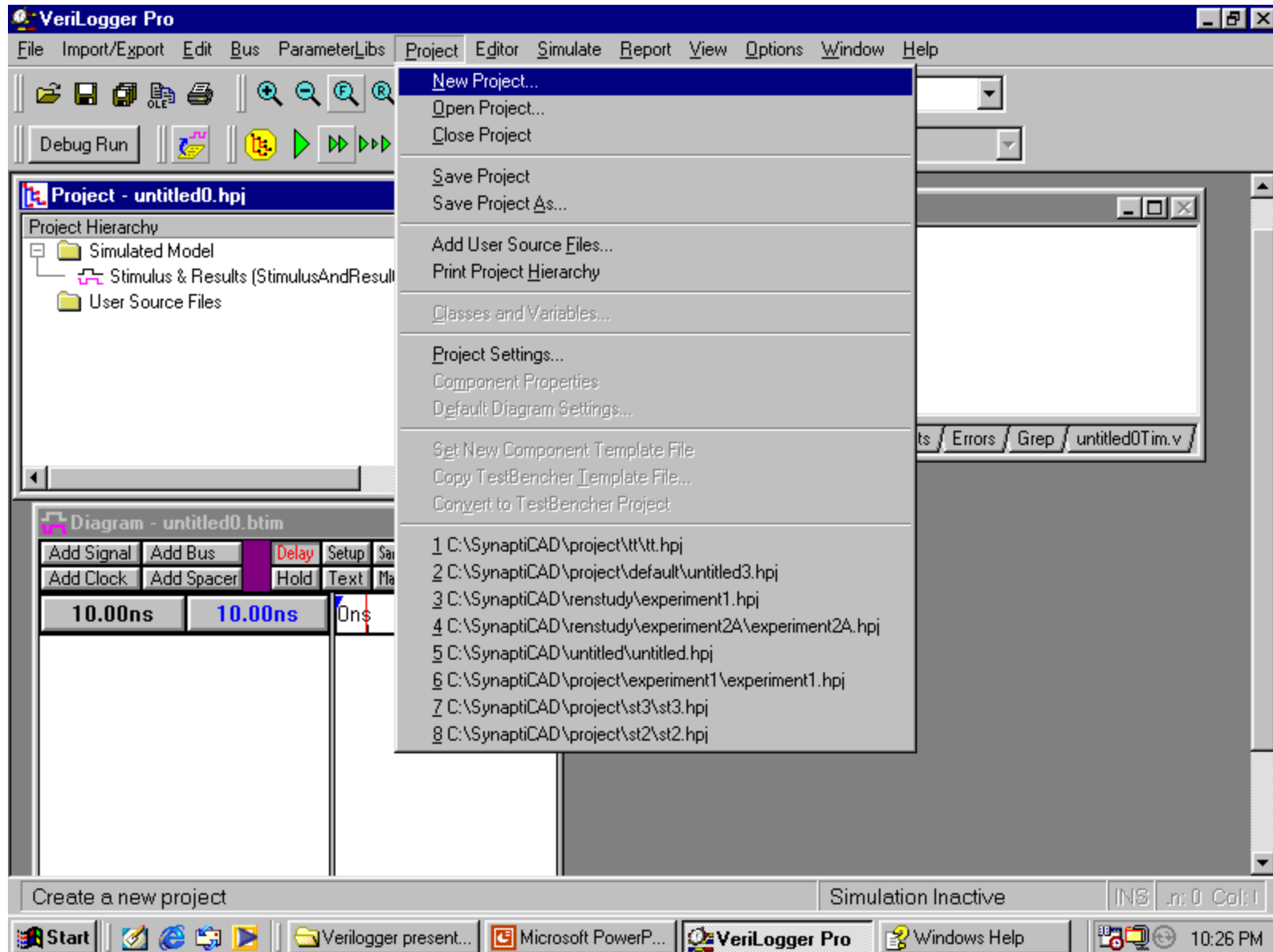
Report - verilog.log

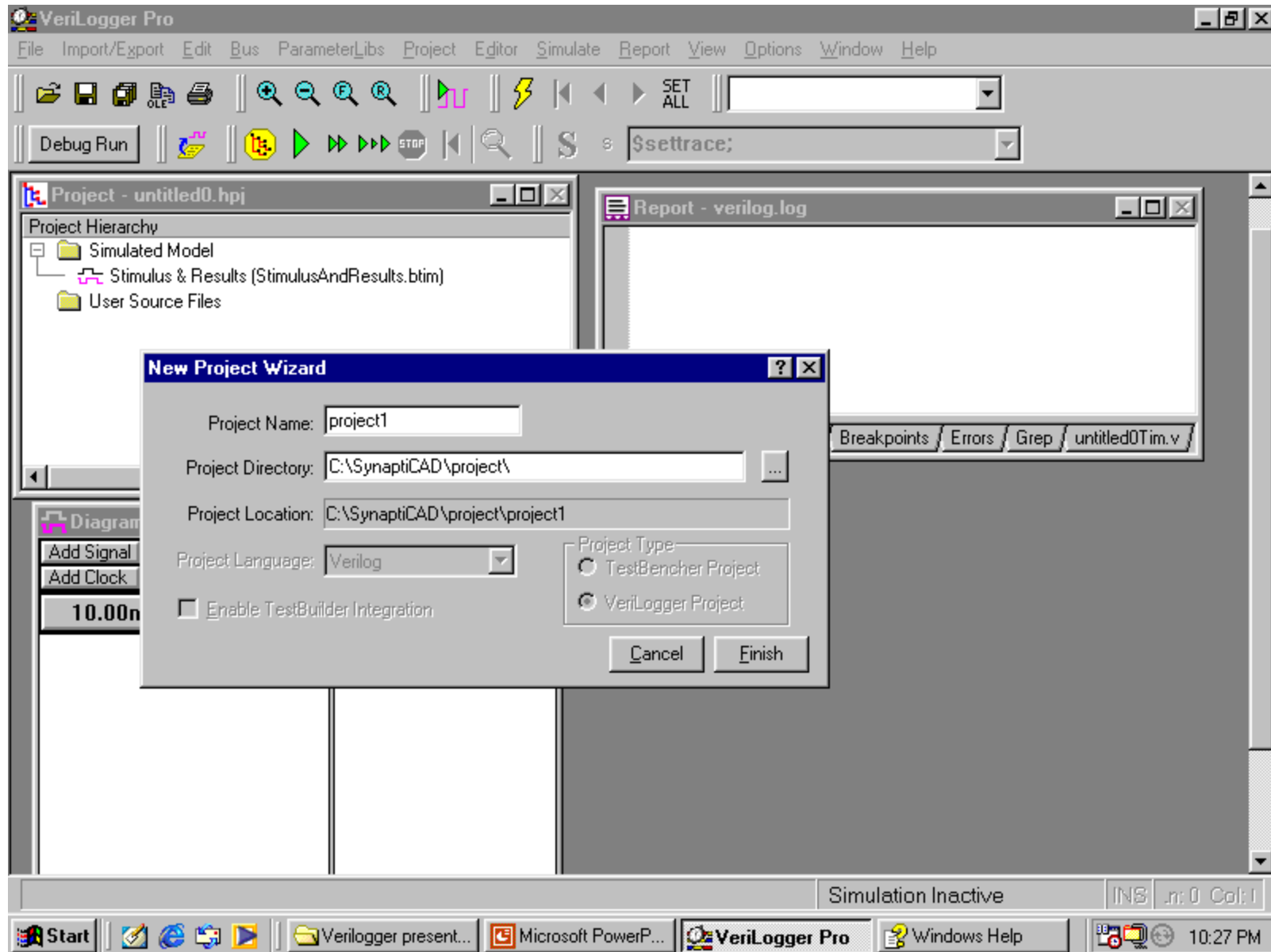
verilog.log waveperl.log Breakpoints Errors Grep untitled0Tim.v

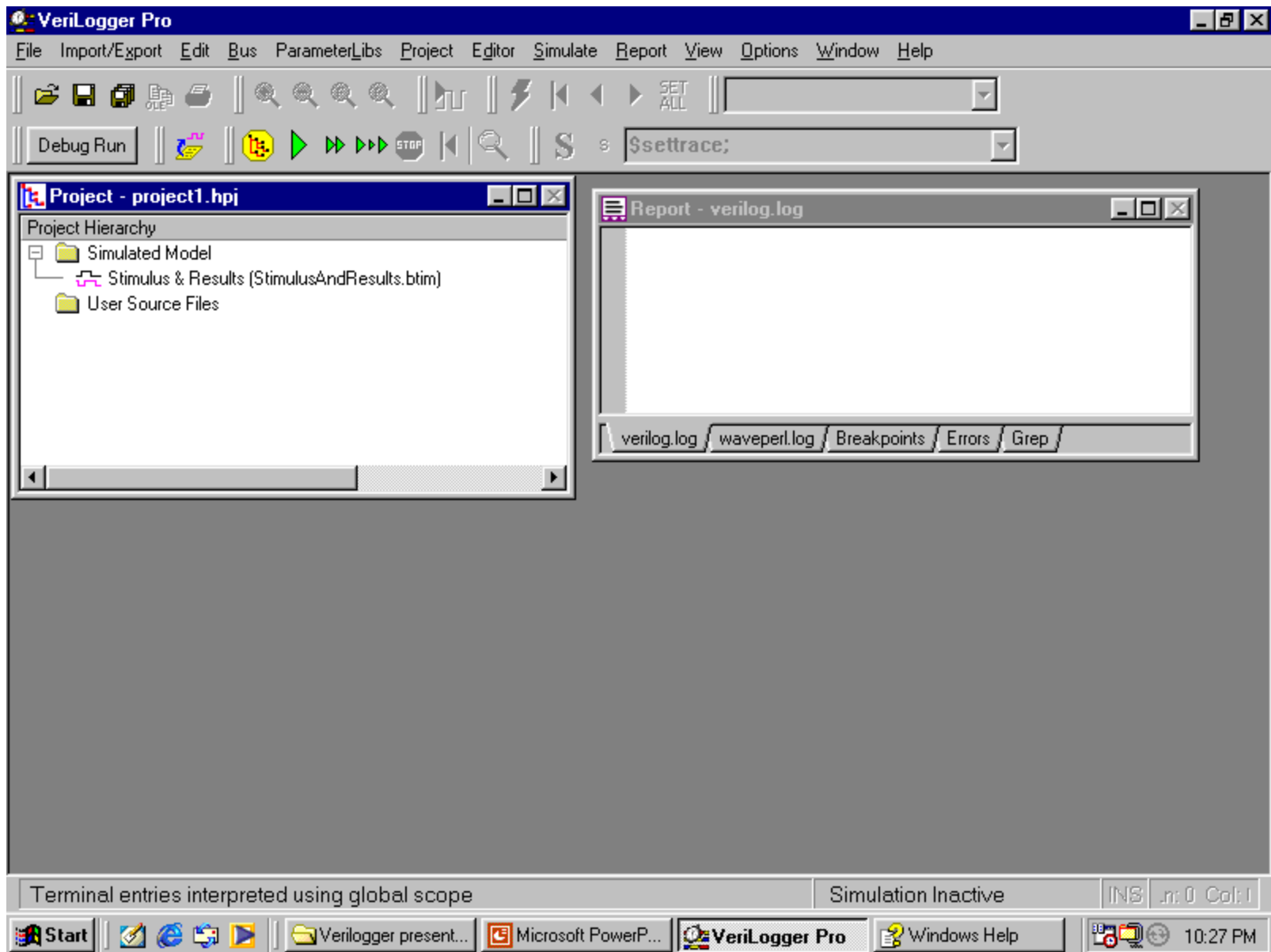
Diagram - untitled0.btim

Add Signal	Add Bus	Delay	Setup	Sample	HIGH	LOW
Add Clock	Add Spacer	Hold	Text	Marker		

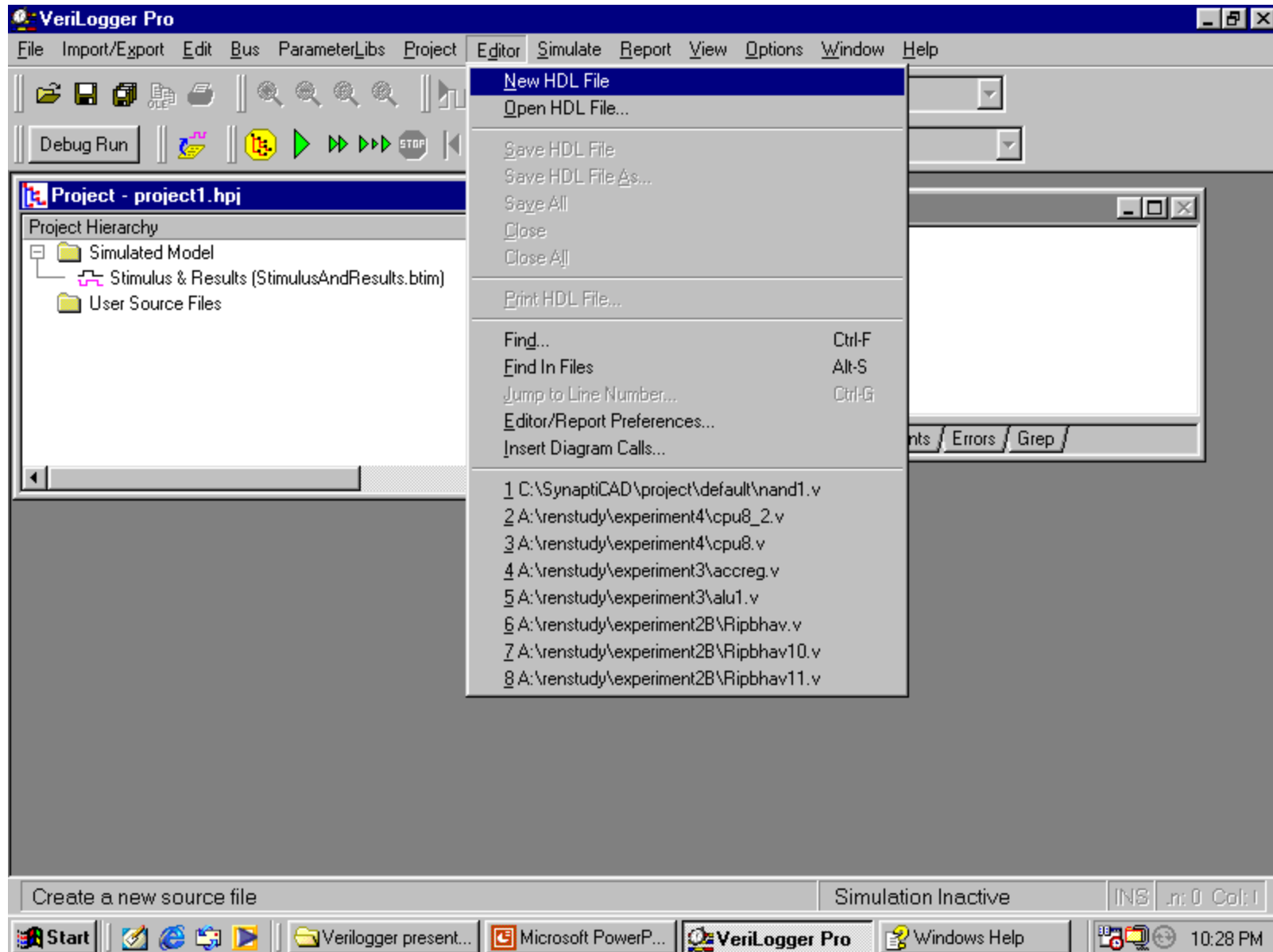
62.00ns 62.00ns 0ns 50ns

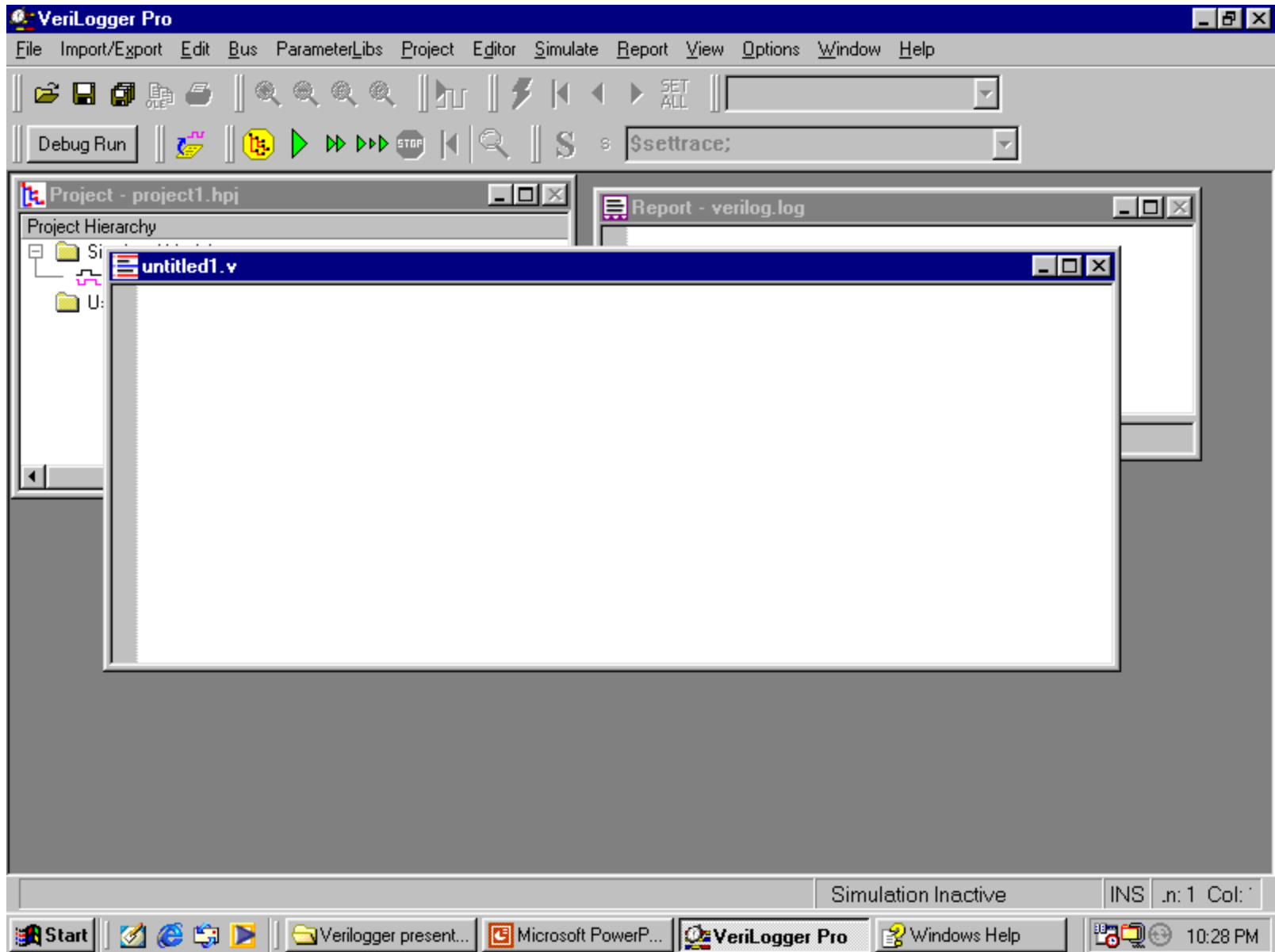


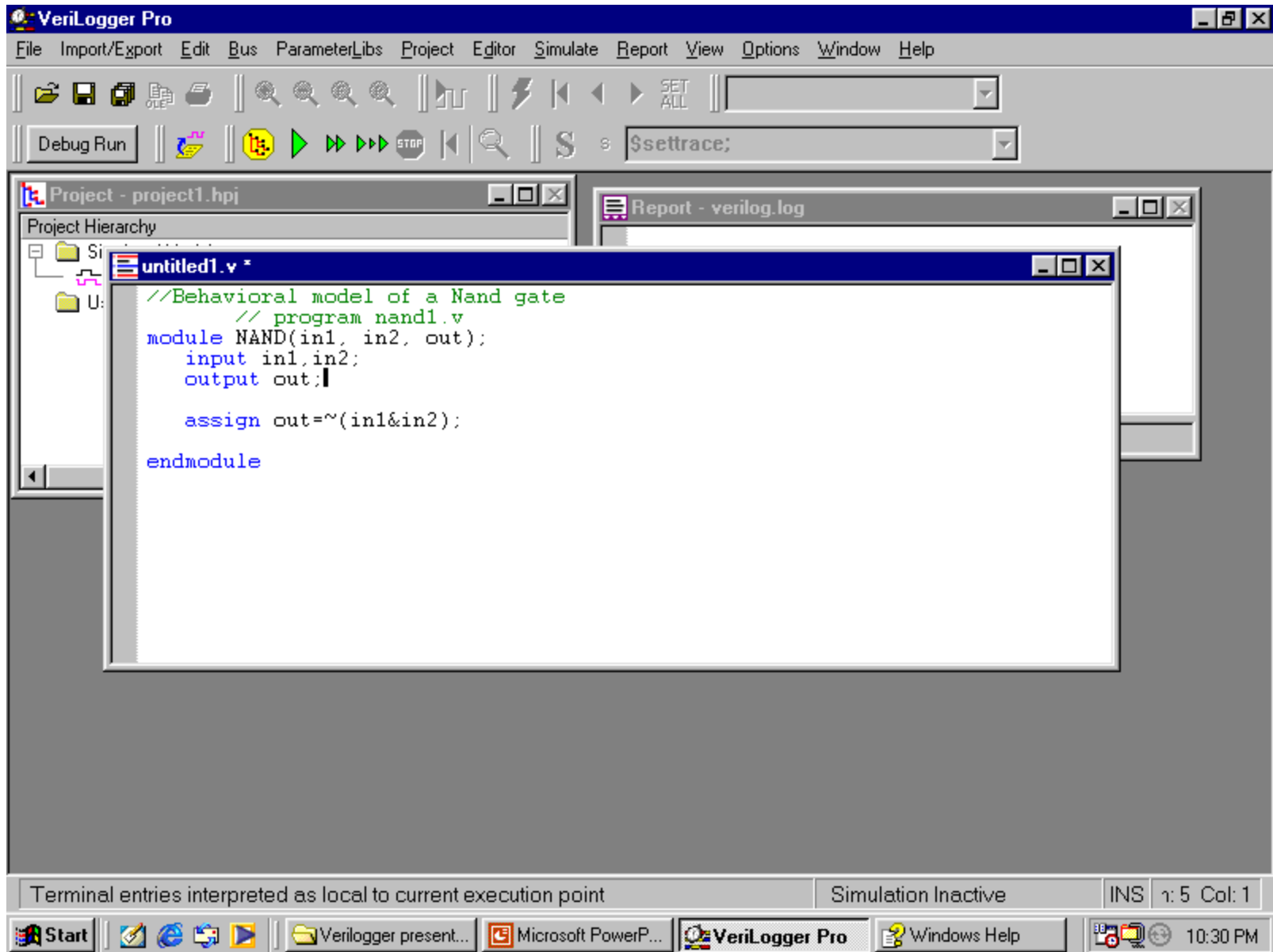




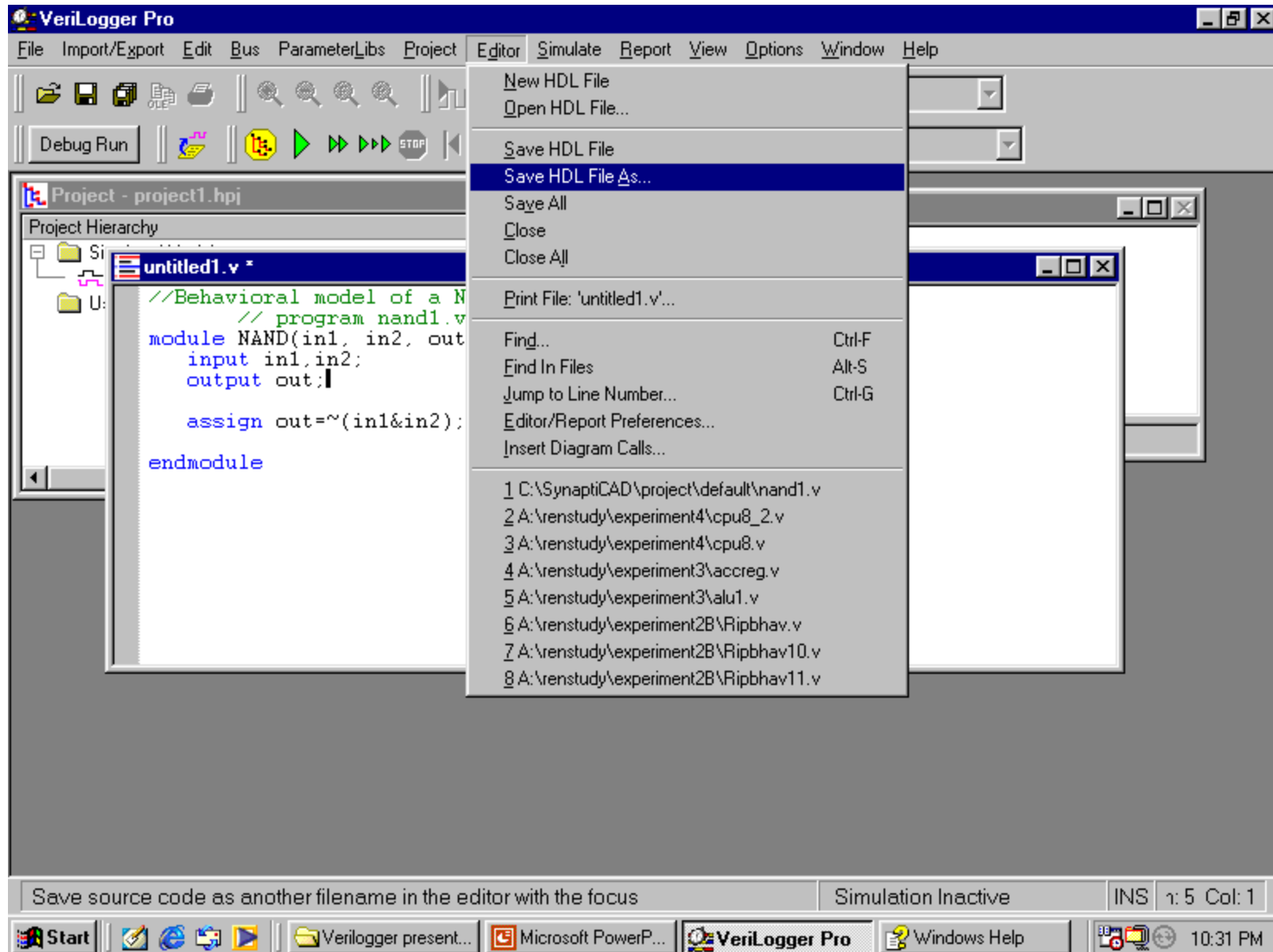
How to create a HDL file?

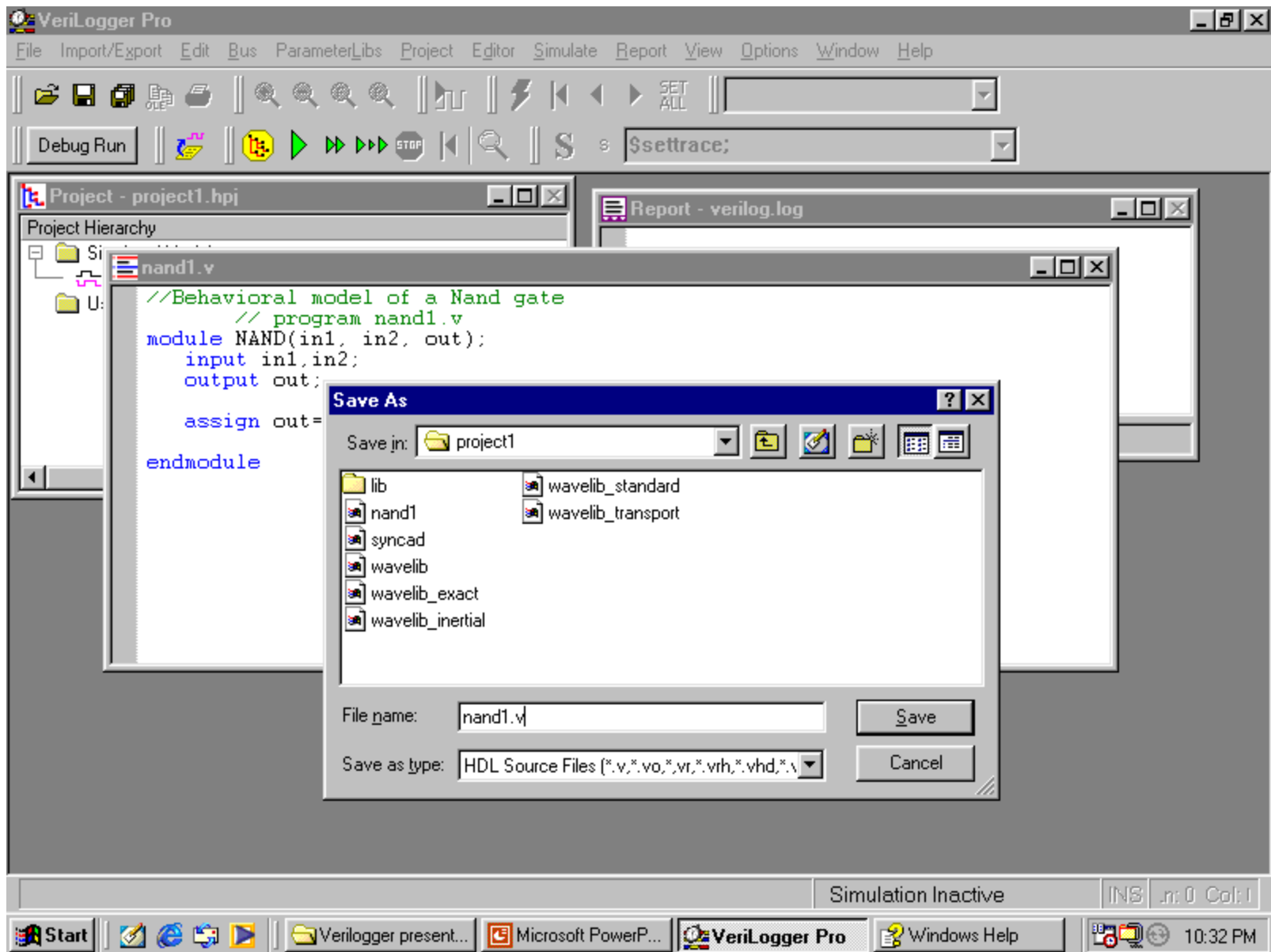


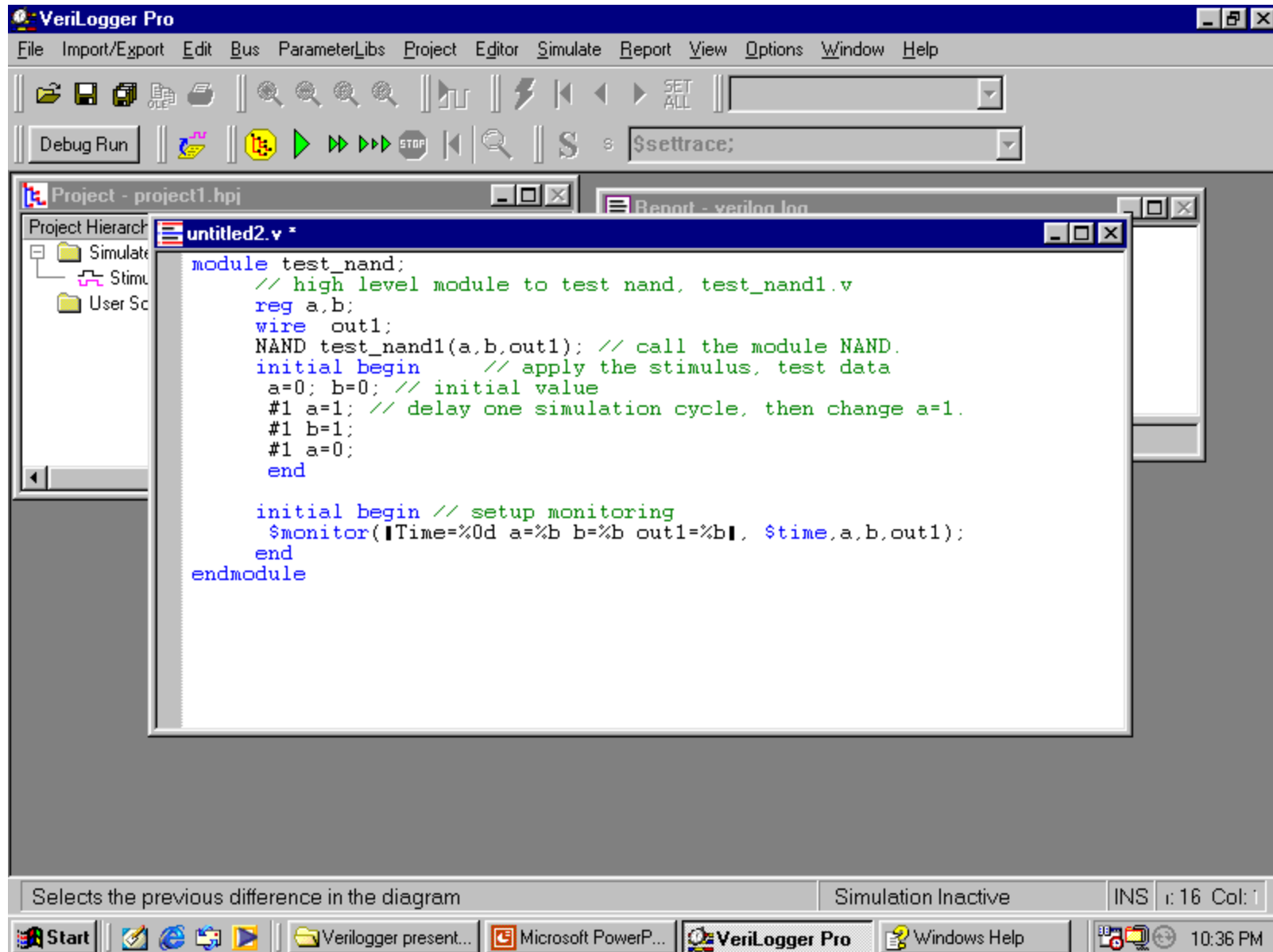


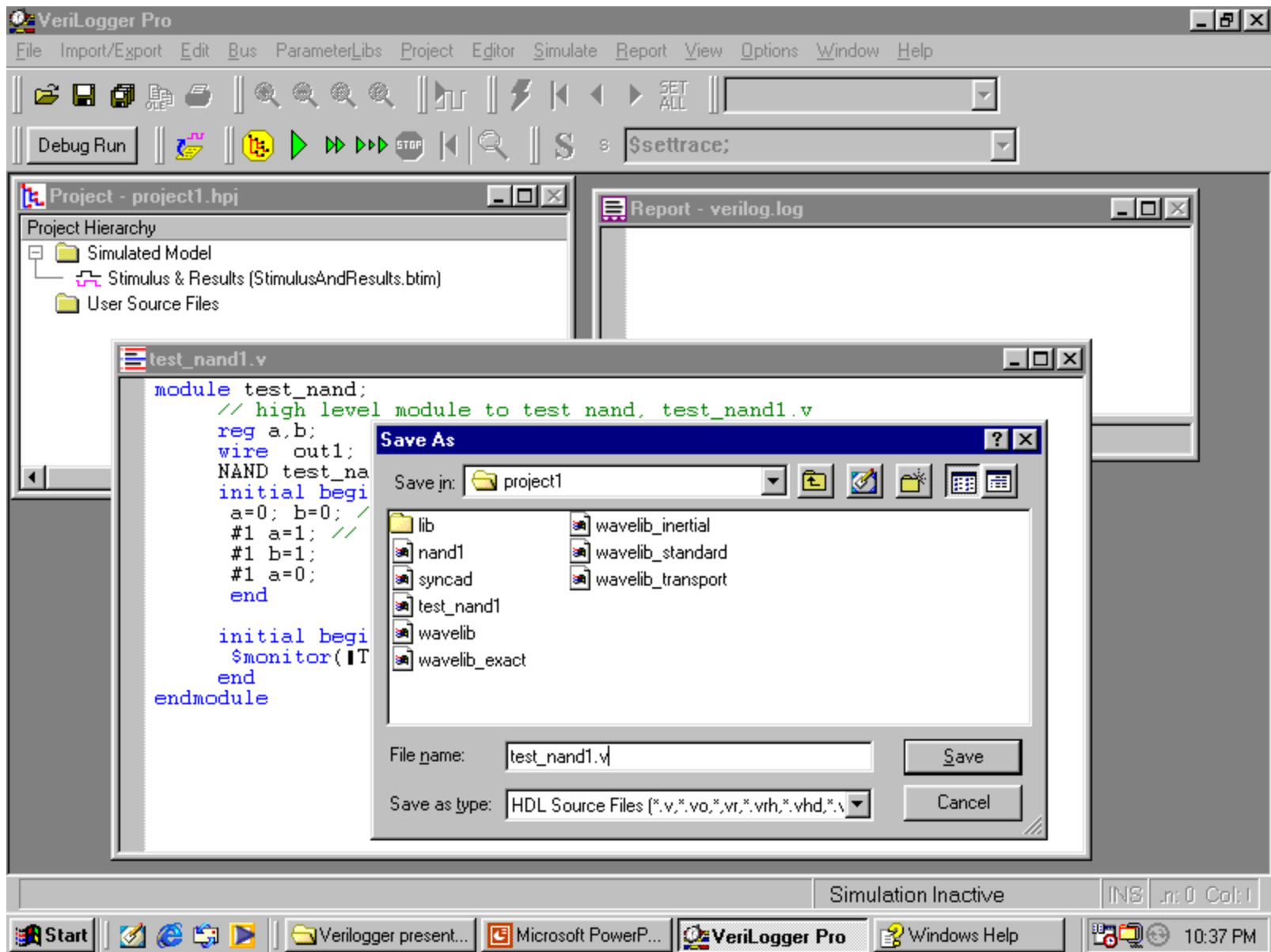


How to save the HDL file?

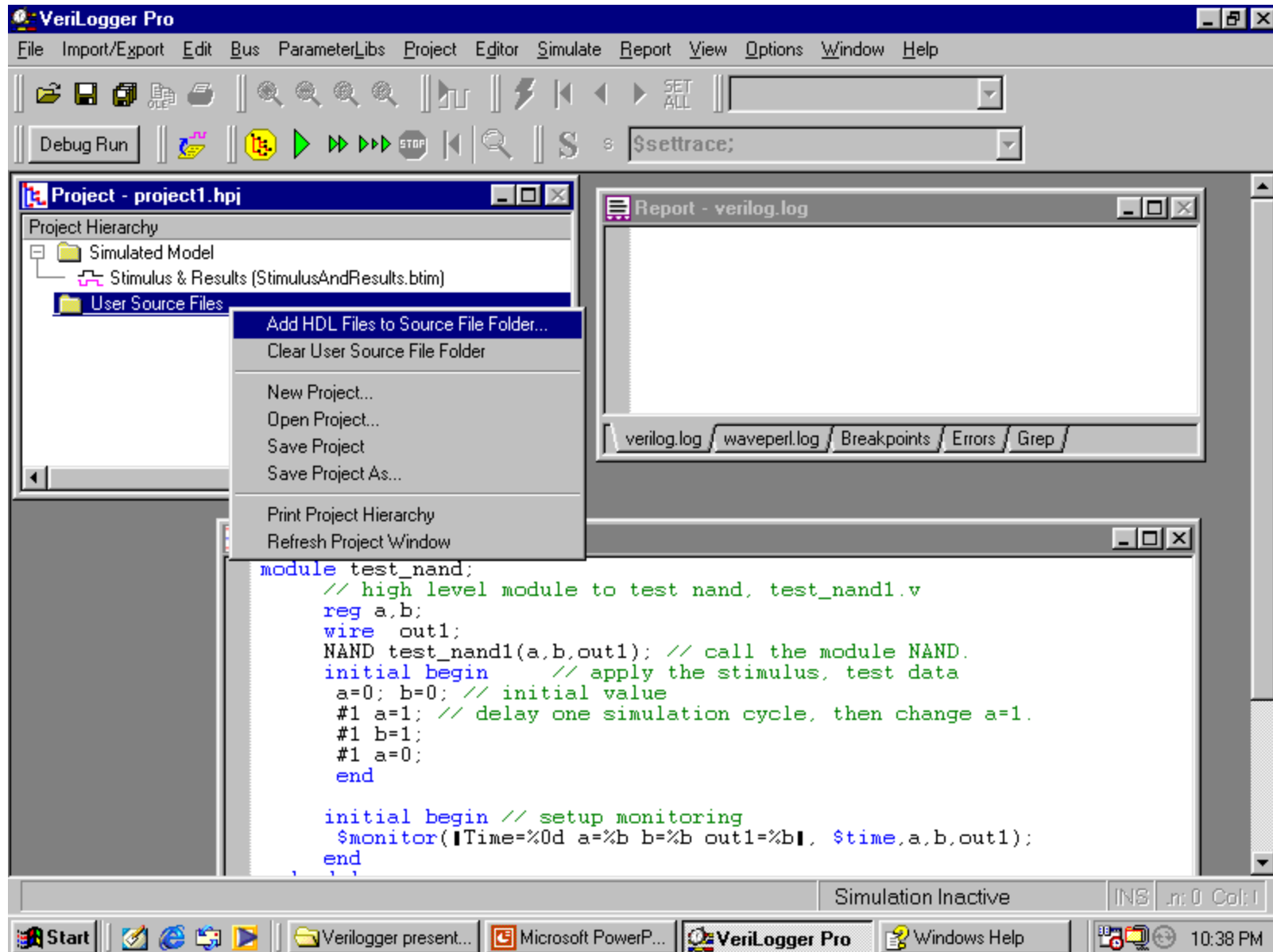


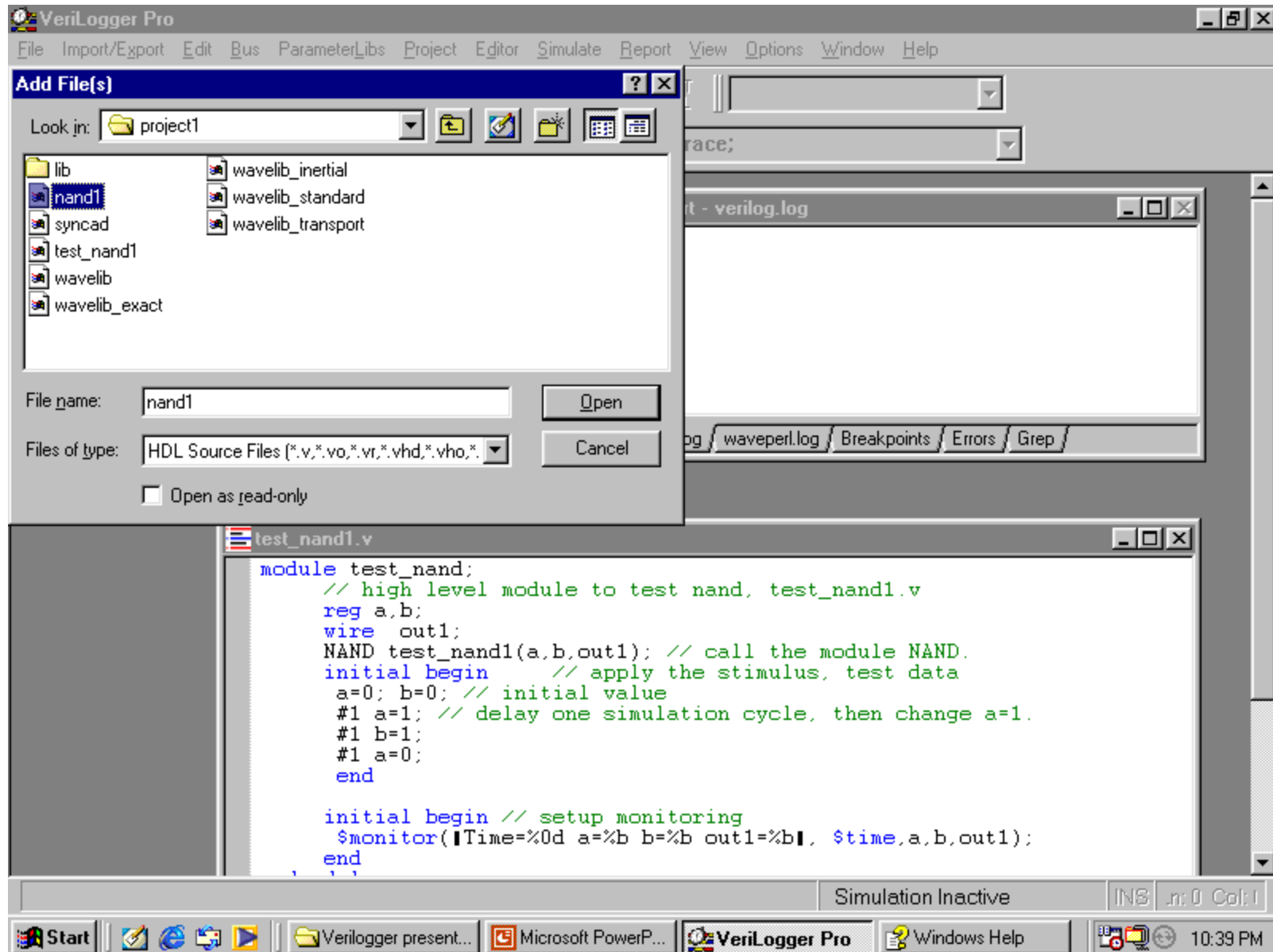


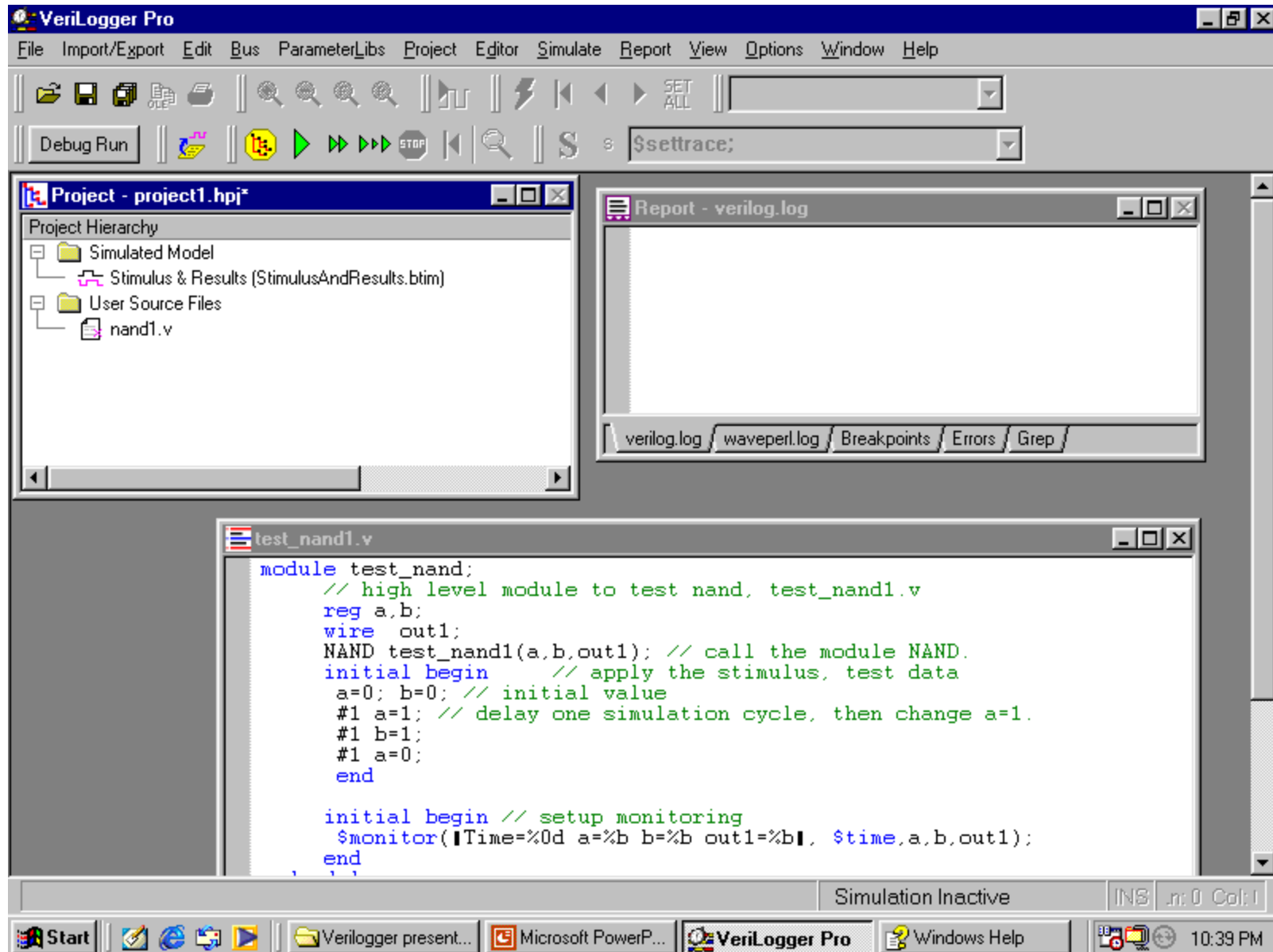


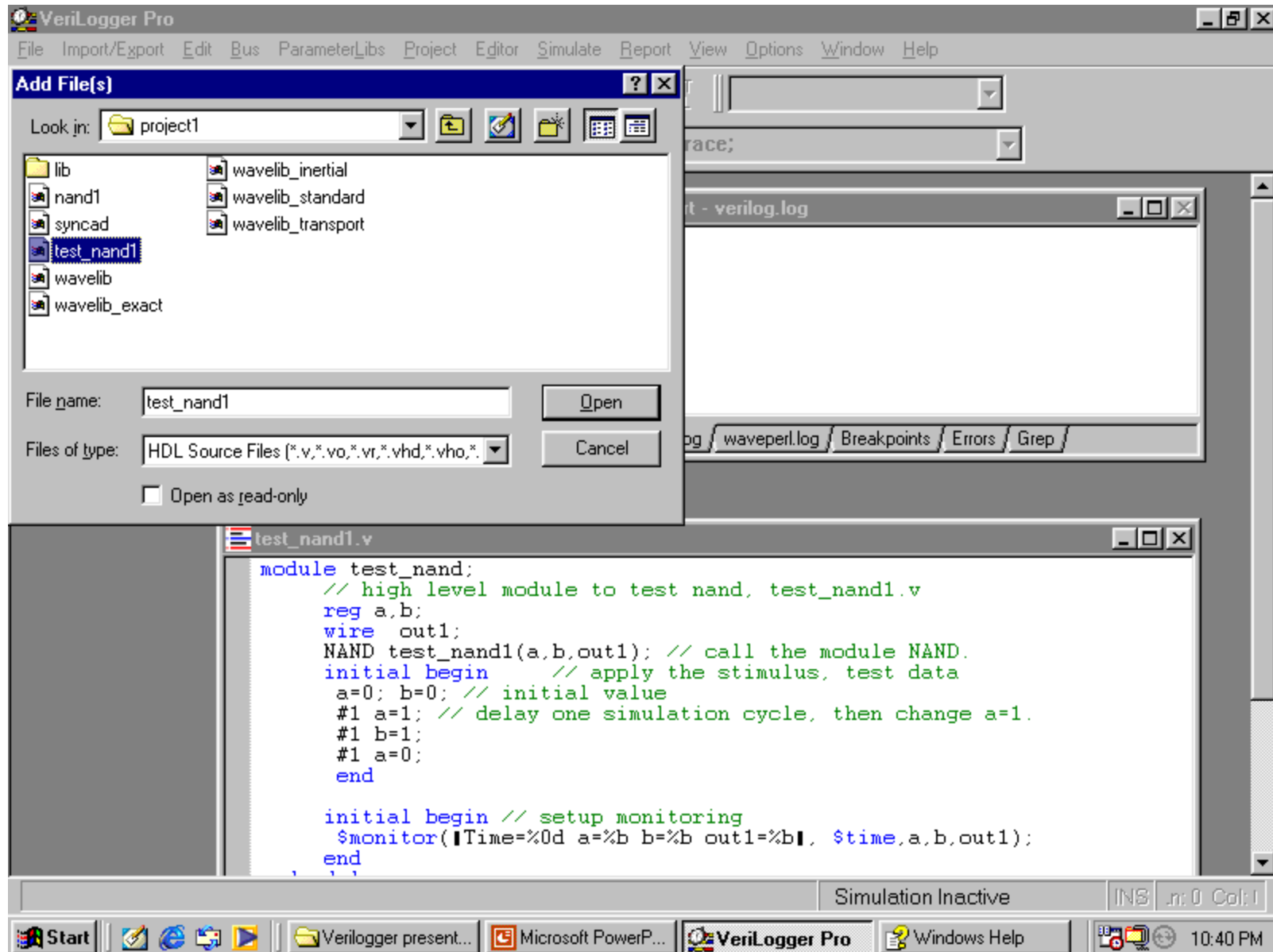


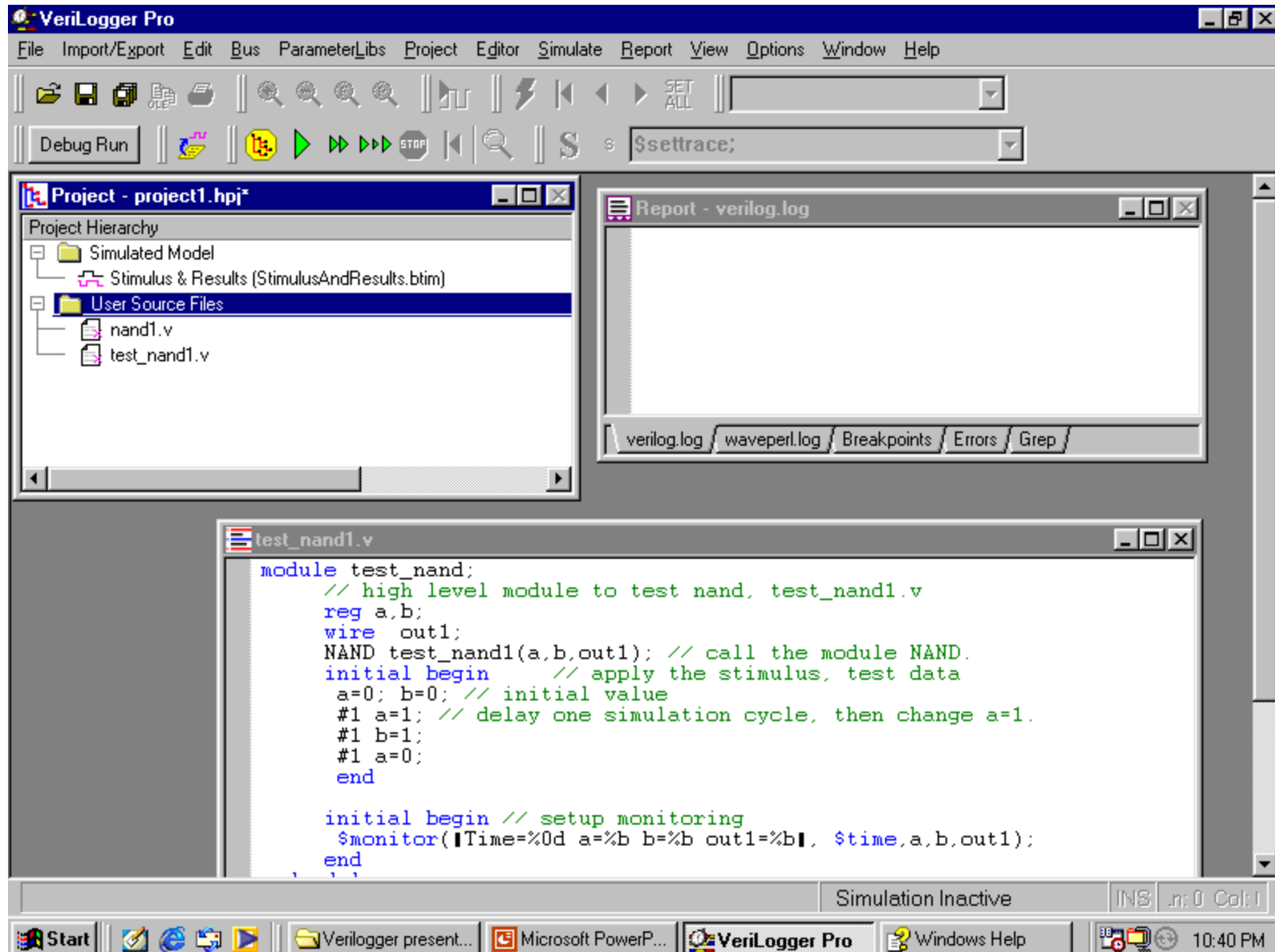
How to add a source HDL file
to a Project(project1)



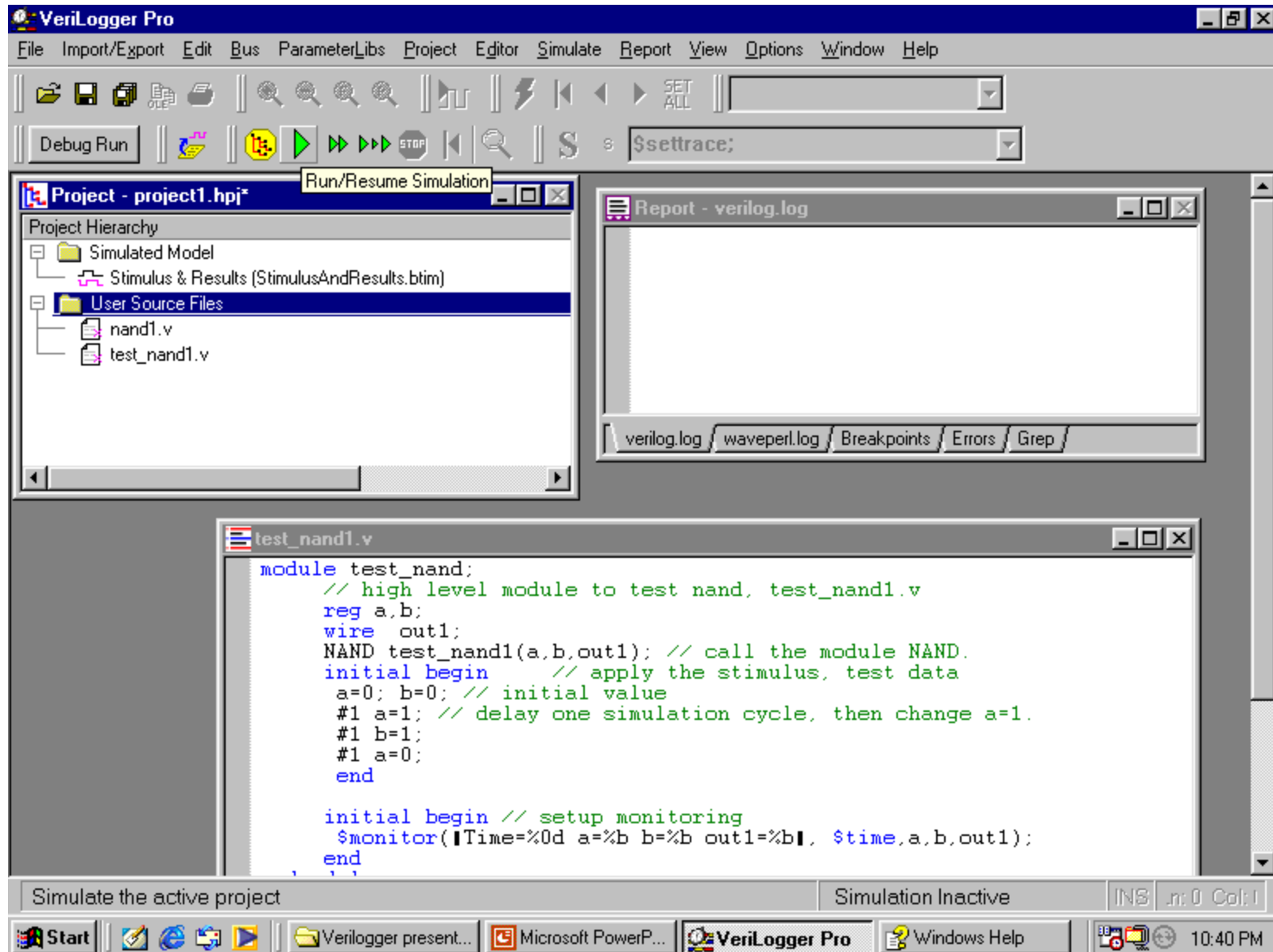








Now, Ready to run the
program!



VeriLogger Pro

File Import/Export Edit Bus ParameterLibs Project Editor Simulate Report View Options Window Help

Debug Run \$settrace;

Project - project1.hpj*

Project Hierarchy

- Simulated Model
 - <<<test_nand>>
 - Stimulus & Results (StimulusAndResults.btim)
- User Source Files
 - nand1.v
 - test_nand1.v

Report - verilog.log

```

Time=1 a=1 b=0 out1=1
Time=2 a=1 b=1 out1=0
Time=3 a=0 b=1 out1=1
0 Errors, 0 Warnings
Compile time = 0.000000, Load time = 0.0000
Normal exit
  
```

verilog.log / waveperl.log / Breakpoints / Errors / Grep

test_nand1.v

Diagram - StimulusAndResults.btim*

Add Signal Add Bus Delay Setup Sample HIGH LOW TRI VAL INVal WHI WLO HEX

Add Clock Add Spacer Hold Text Marker

0.000ps 0.000ps 0ns 50ns 100ns 150ns 200ns

test_nand.a

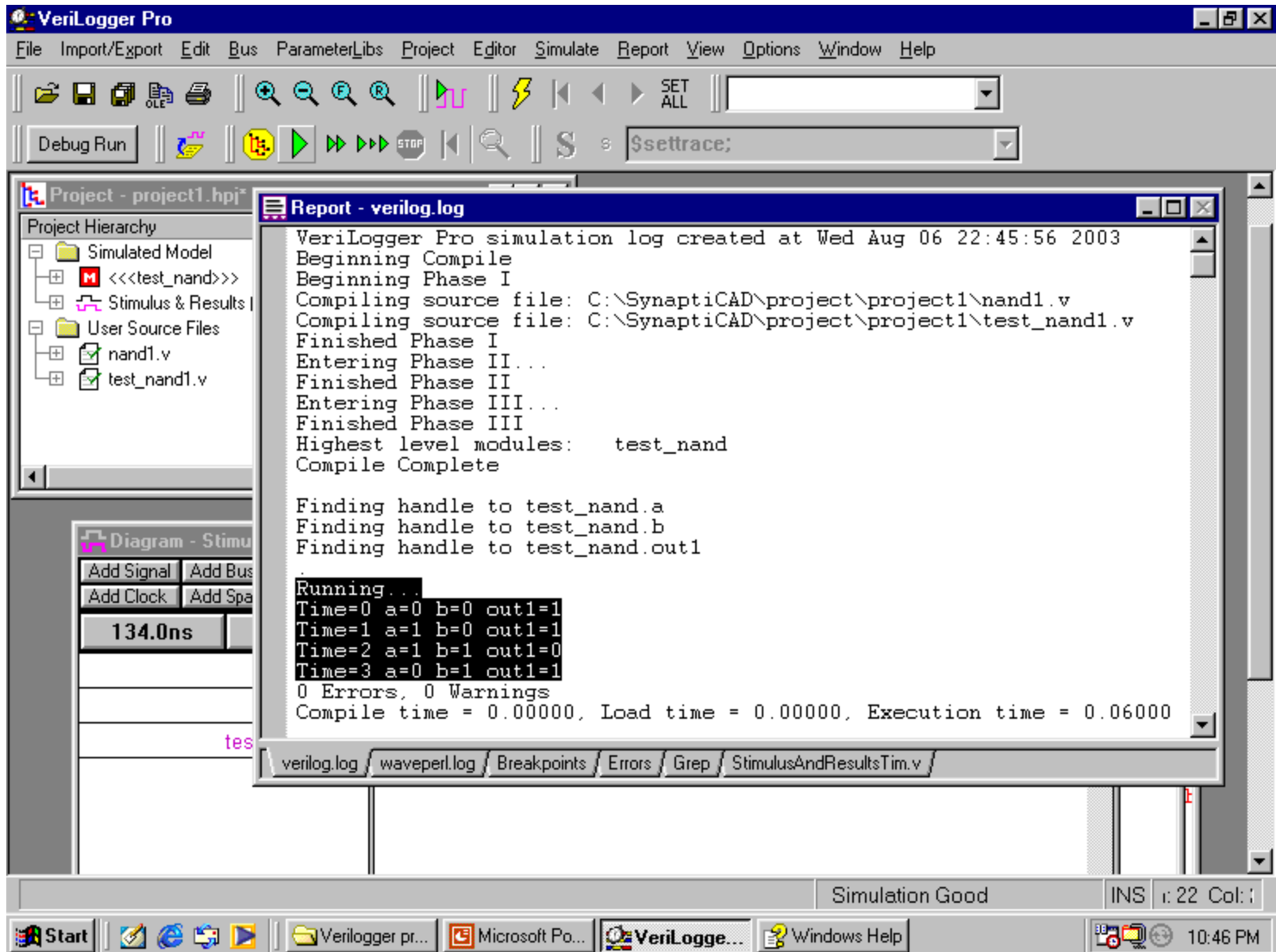
test_nand.b

test_nand.out1

Evaluating StimulusAndResults.btim... Simulation Good INS .n: 0 Col: 1

Start Verilogger pr... Microsoft Po... VeriLogge... Windows Help 10:46 PM

The Report Window of Verilogger. (all the simulation information is in this window)



Example of gate NAND

- Simulation report from Verilog-Report window.

Running...

Time=0 a=0 b=0 out1=1

Time=1 a=1 b=0 out1=1

Time=2 a=1 b=1 out1=0

Time=3 a=0 b=1 out1=1

0 Errors, 0 Warnings

Compile time = 0.00000, Load time = 0.00000,
Execution time = 0.06000

Normal exit

Diagram window of Simulation result

VeriLogger Pro

File Import/Export Edit Bus ParameterLibs Project Editor Simulate Report View Options Window Help

Debug Run \$settrace;

Project - project1.hpj*

Project Hierarchy

- Simulated Model
 - <<<test_nand>>
- Stimulus & Results
- User Source Files
 - nand1.v
 - test_nand1.v

Report - verilog.log

```

VeriLogger Pro simulation log created at Wed Aug 06 22:45:56 2003
Beginning Compile
Beginning Phase I
Compiling source file: C:\SynaptiCAD\project\project1\nand1.v
Compiling source file: C:\SynaptiCAD\project\project1\test_nand1.v
Finished Phase I
Entering Phase II...
Finished Phase II
Entering Phase III...
  
```

Diagram - StimulusAndResults.btim*

Add Signal Add Bus Delay Setup Sample HIGH LOW TRI VAL INVal WHI WLO HEX

Add Clock Add Spacer Hold Text Marker

97.00ns 23.00ns 0ns 50ns 100ns 150ns 200ns

test_nand.a

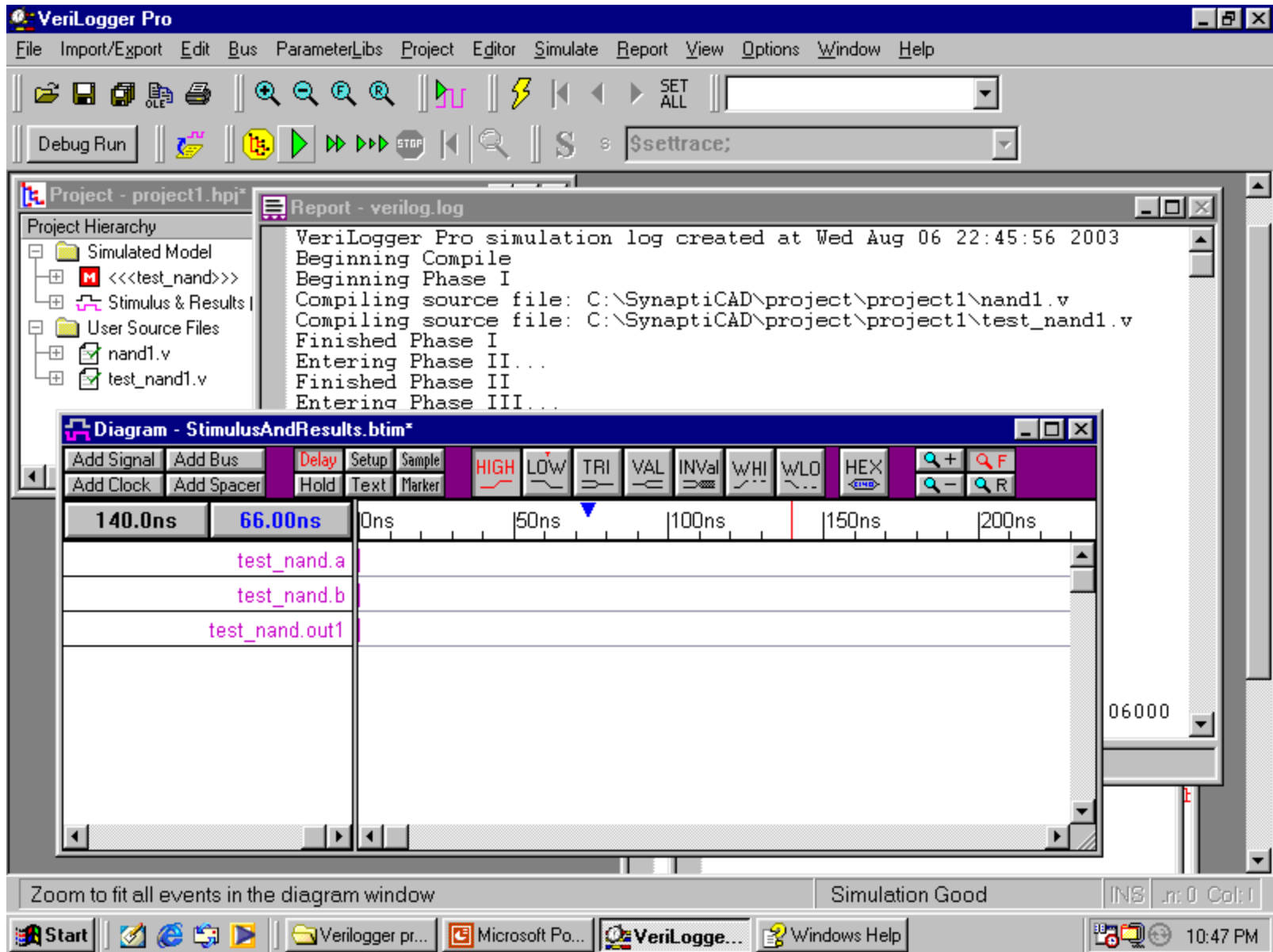
test_nand.b

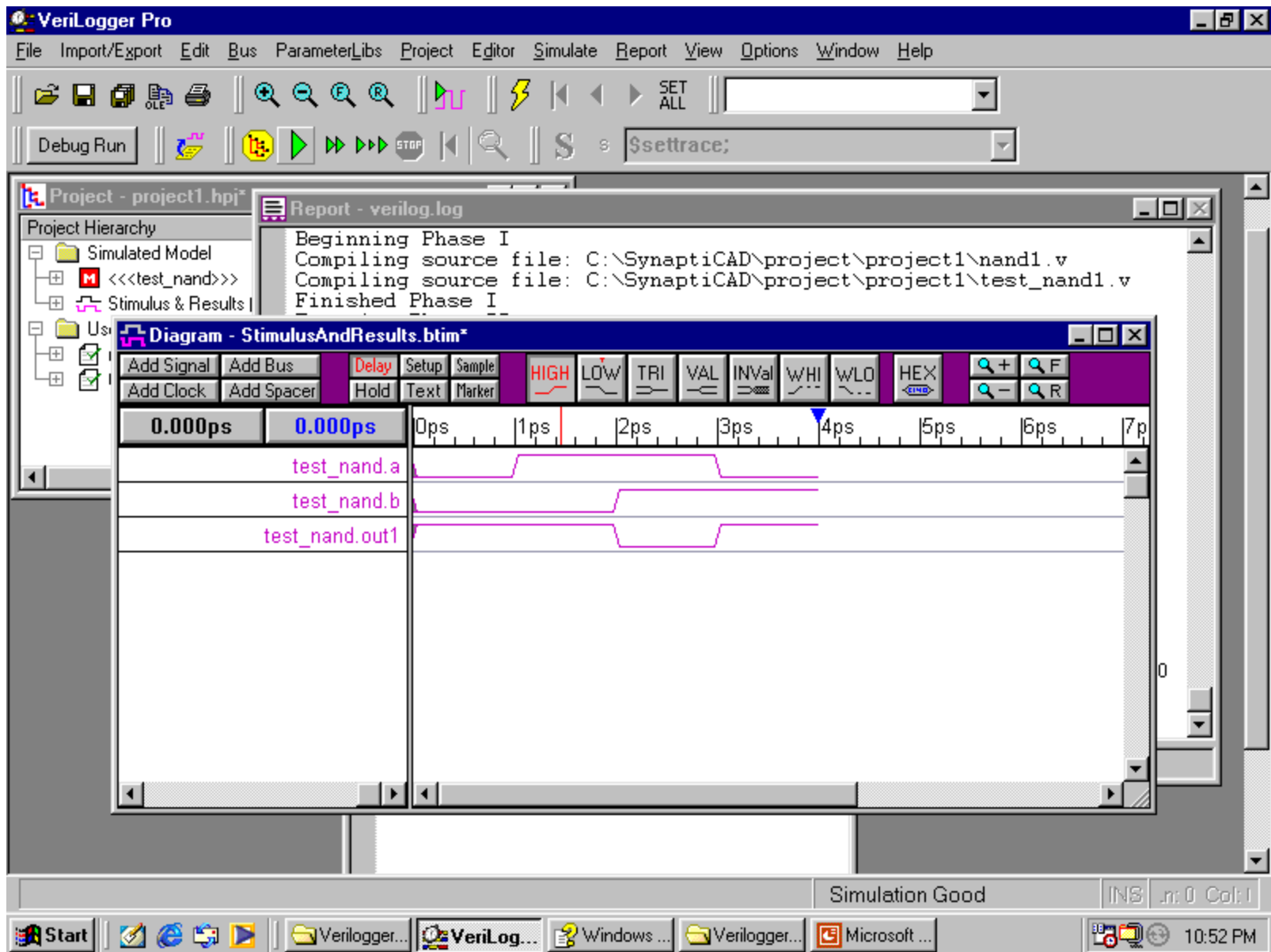
test_nand.out1

06000

Click to draw waveform displayed on red state button Simulation Good INS n: 0 Col: 1

Start Verilogger pr... Microsoft Po... VeriLogge... Windows Help 10:47 PM





How to copy the diagram to Microsoft Word!

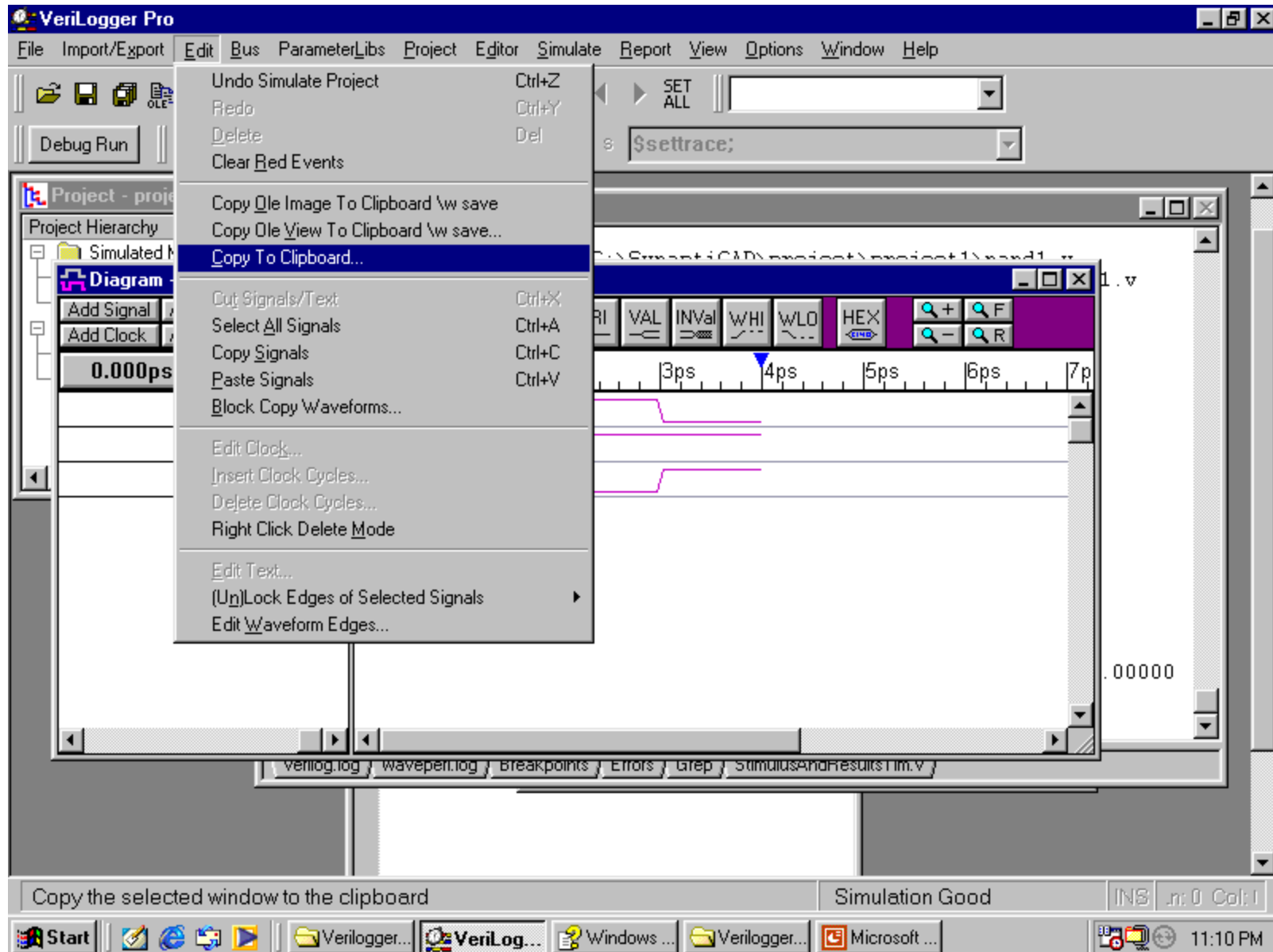
Example of gate NAND

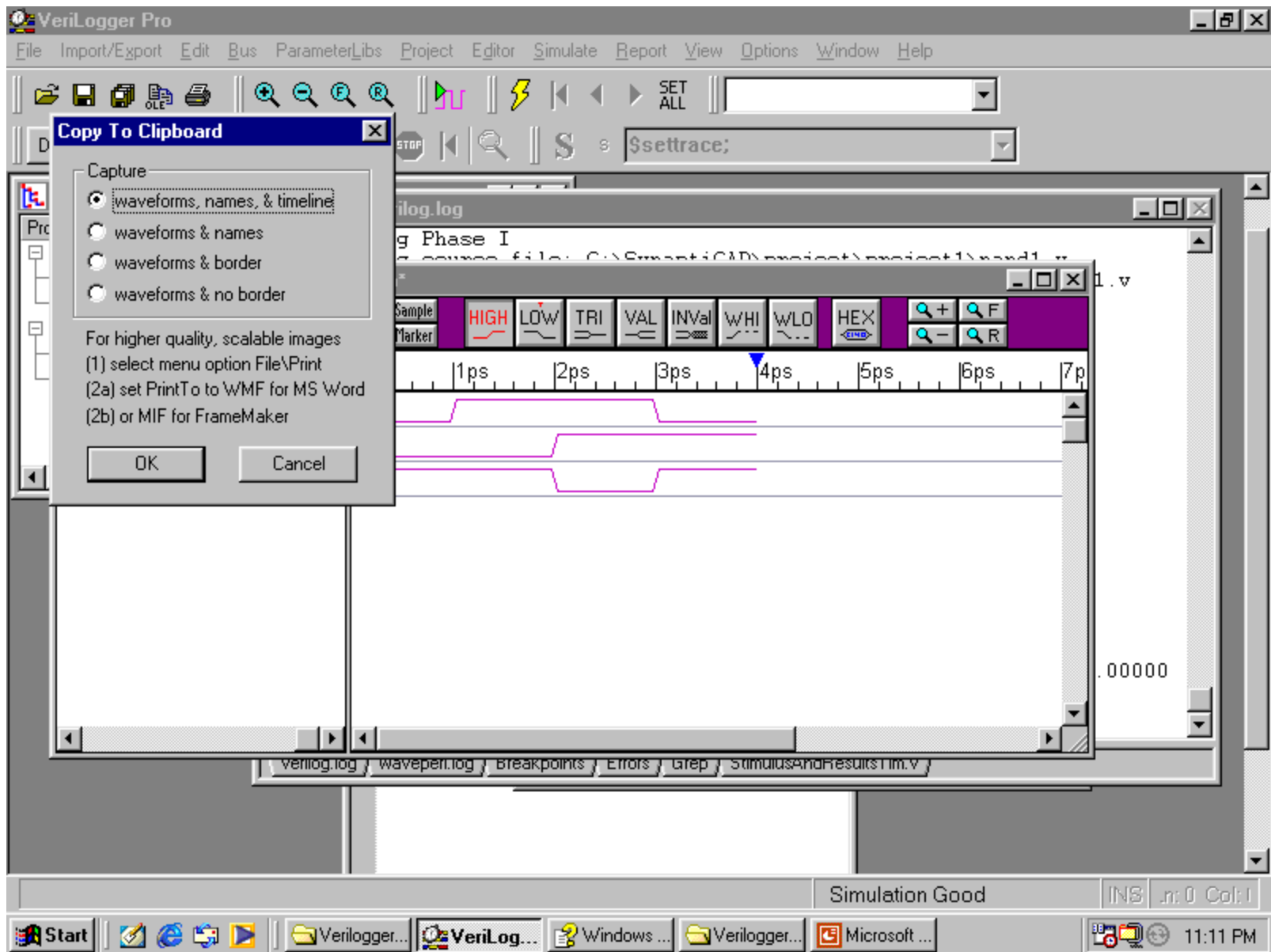
- Wave from Verilog diagram.
- Verilog windows

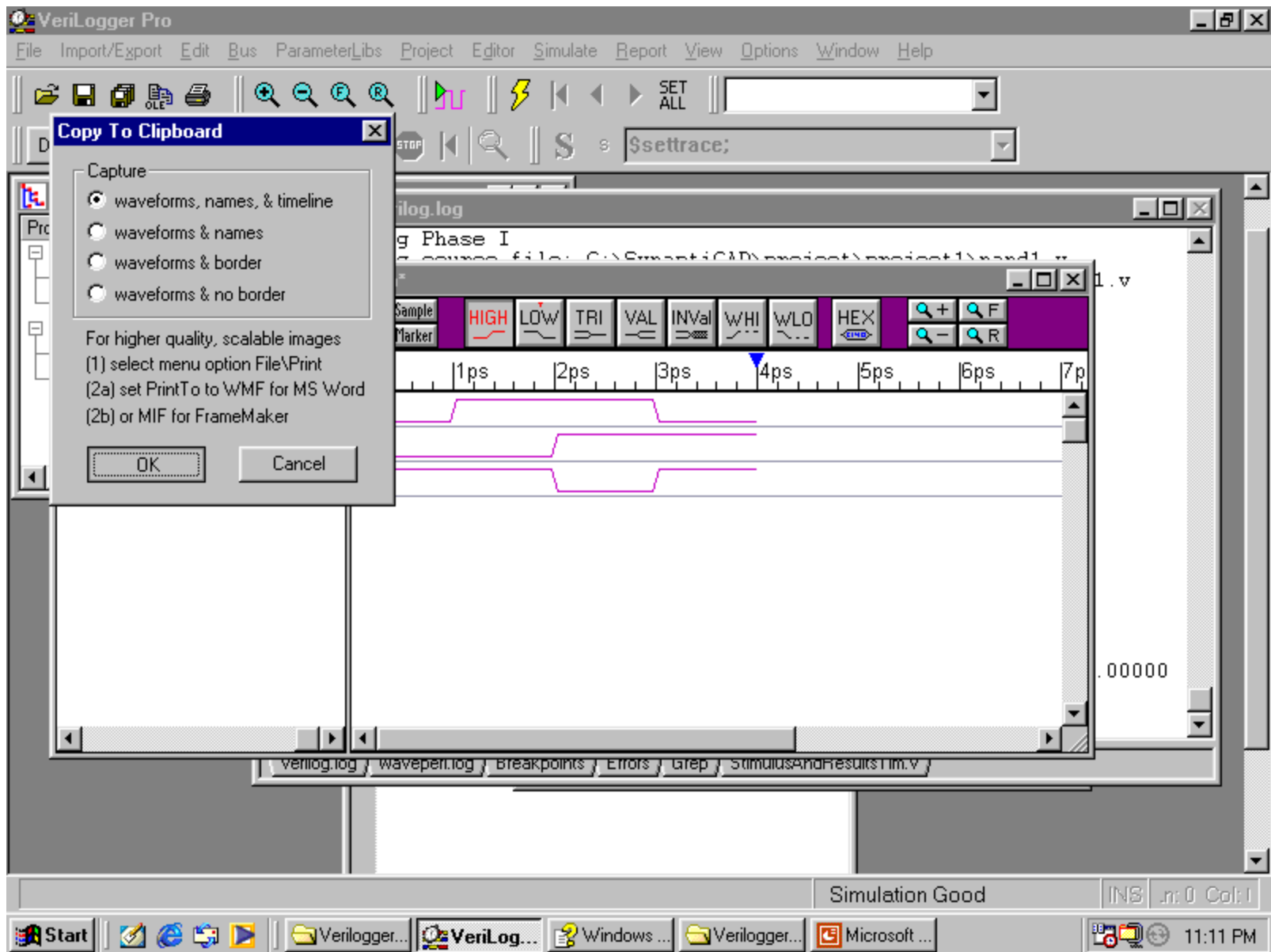
Activate the the diagram windows

Method 1: [File] -> [Print Diagram] -> Print to: [WMF
Metafile[MS Word];

Method 2: [edit]→ [copy to clipboard]→select “wave
form, name and time line”→select “ok”→
then you can paste the diagram to anywhere you
want.







You can paste the diagram here!

