HARDWARE DESCRIPTION LANGUAGE

Digital Design (with an introduction to the Verilog HDL) 6th Edition,
M. Morris Mano, Michael D. Ciletti

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING,
KANCHEEPURAM

- Dr. Kalpana Settu
  Assistant Professor
  ECE, IIITDM Kancheepuram

HDL

- A hardware description language (HDL) is a computer-based language that describes the hardware of digital systems in a textual form. It resembles an ordinary computer programming language, such as C, but is specifically oriented to describing hardware structures and the behavior of logic circuits.
- It can be used to represent logic diagrams, truth tables, Boolean expressions, and complex abstractions of the behavior of a digital system.
- One way to view an HDL is to observe that it describes a relationship between signals that are the inputs to a circuit and the signals that are the outputs of the circuit.

➢ There are two standard HDLs that are supported by the IEEE: VHDL and Verilog (IEEE: Institute of Electrical and Electronics Engineers).

➢ Why Verilog, not VHDL? Easier!!

Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL)

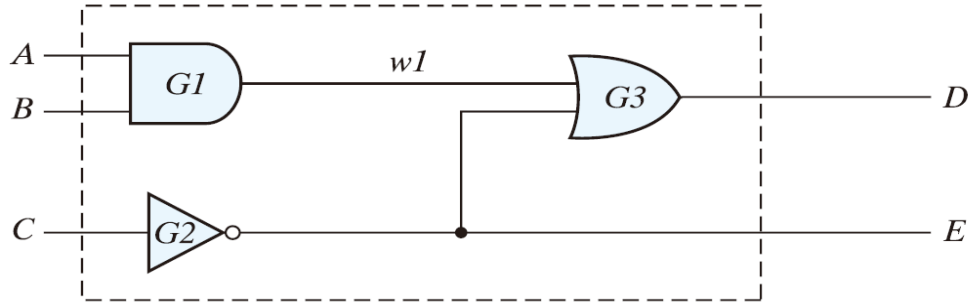
Three Modeling Styles in Verilog

- Gate-level Modeling (also known as Structural modeling) describes a circuit by specifying its gates and how they are connected with each other.
- Dataflow Modeling is used mostly for describing the Boolean equations of combinational logic and uses continuous assignment statements with the keyword "**assign**".
- Behavioral Modeling uses procedural assignment statements with the keyword "**always**". It is used mostly to describe sequential circuits.

Gate-level Modeling

Module Description

- **Keywords:** module, endmodule, input, output, wire, and, or, not




```
// Verilog model of circuit of Figure 3.35. IEEE 1364–1995 Syntax
module Simple_Circuit (A, B, C, D, E);
  output      D, E;
  input       A, B, C;
  wire        w1;

  and         G1 (w1, A, B); // Optional gate instance name
  not         G2 (E, C);
  or          G3 (D, w1, E);
endmodule
```

names given to modules, variables

gate instance (output, input)

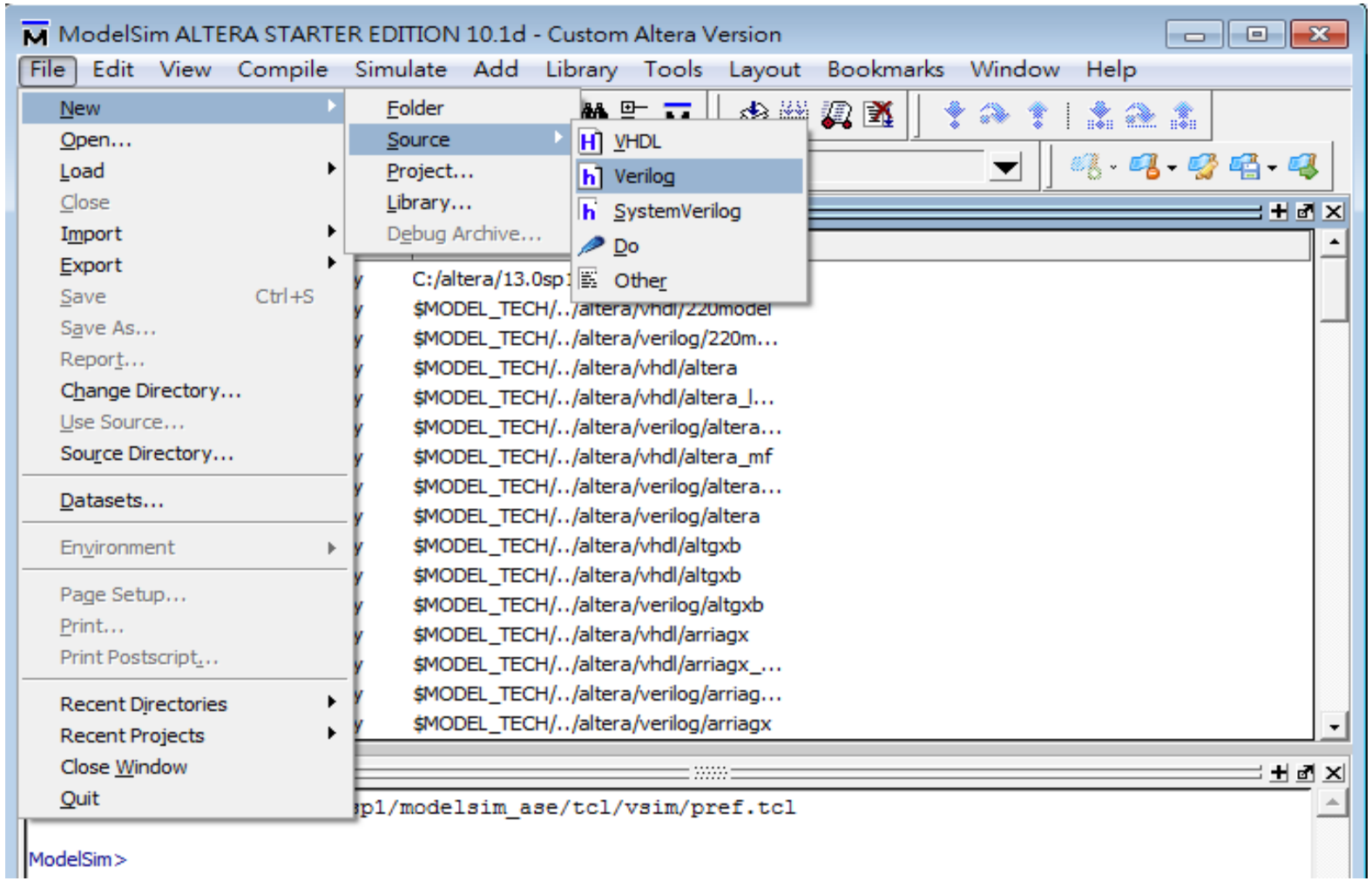
input and output specify which of the ports are inputs and which are outputs

module and endmodule → start and end of the declaration (description) of the module

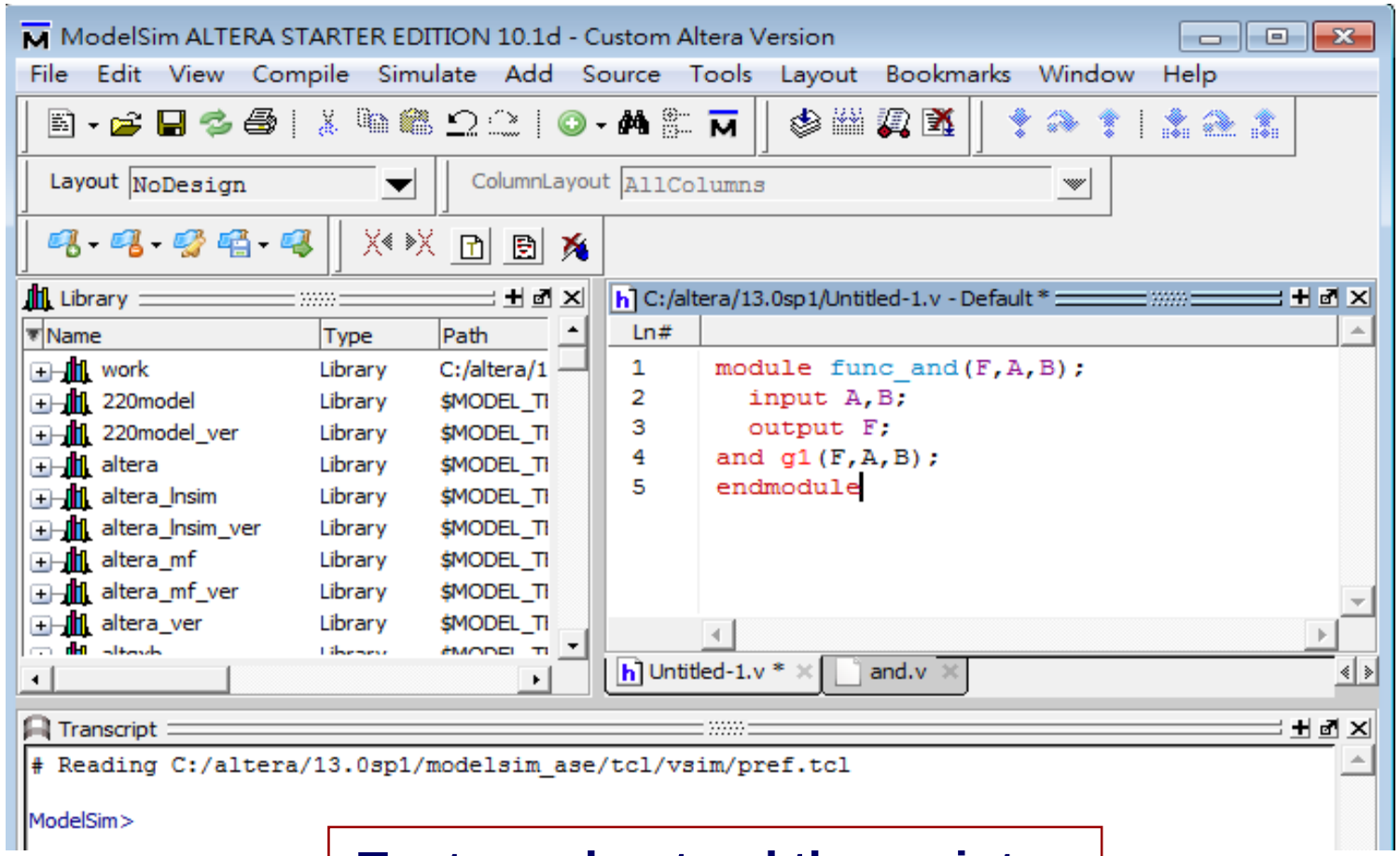
wire → internal connection

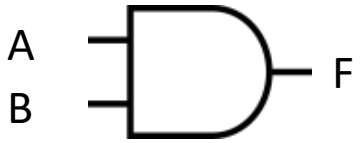
and, or, not → primitive gates (basic functional block used in Verilog HDL)

Create Verilog Script



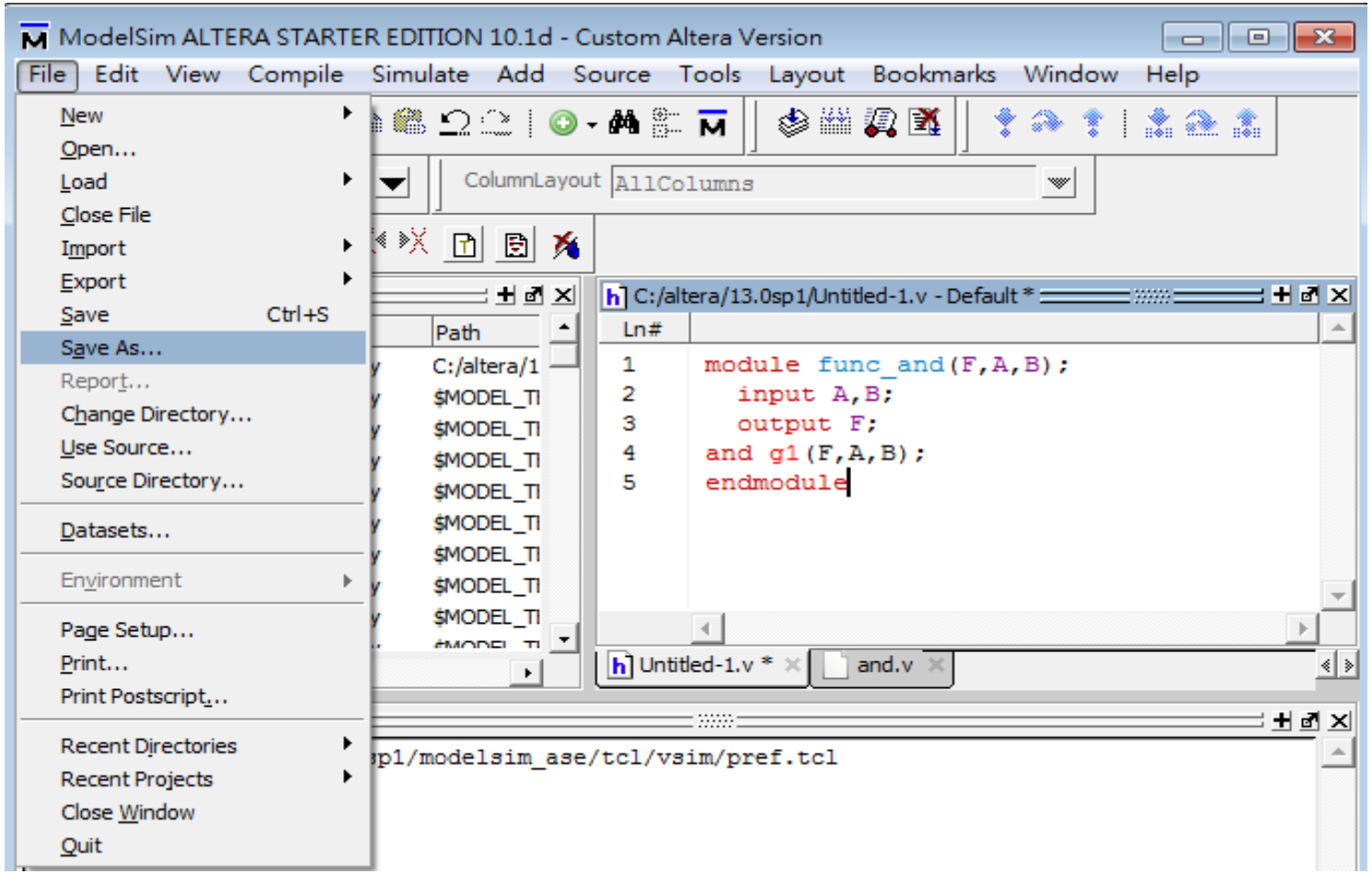
Verilog Script





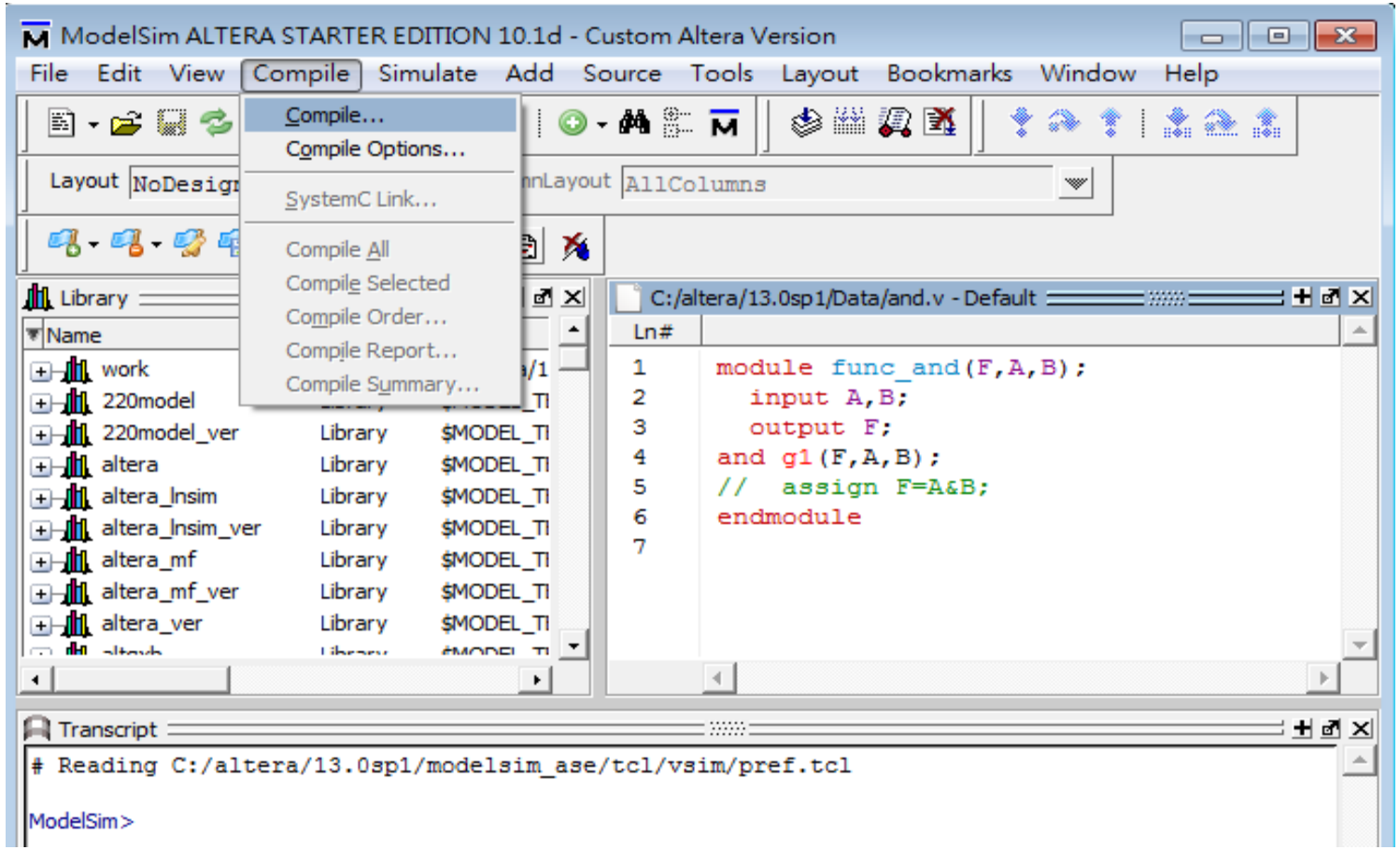
Try to understand the script

Save Your Script

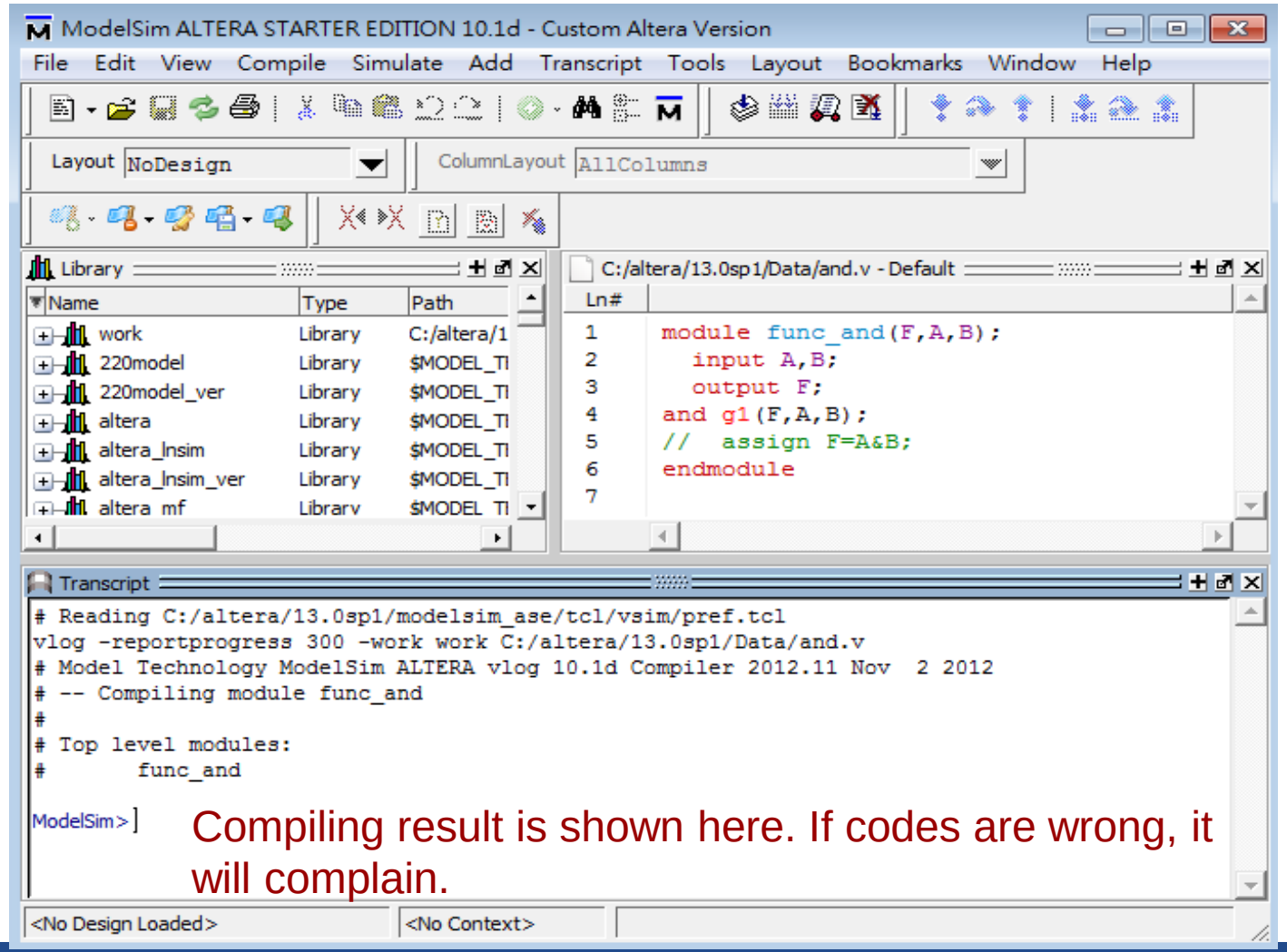


You can create a folder called “Data” and save the file named “and.v” in the folder

Compile Your Script



Compile Your Script



Test Bench

- In order to simulate a circuit with an HDL, it is necessary to apply inputs to the circuit so that the simulator will generate an output response.
- An HDL description that provides the stimulus to a design is called a *test bench*.
- In its simplest form, a test bench is a module containing a signal generator and an instantiation of the model that is to be verified.

Create a Test Bench

C:/intelFPGA_pro/17.0/data/test_bench_2var.v - Default

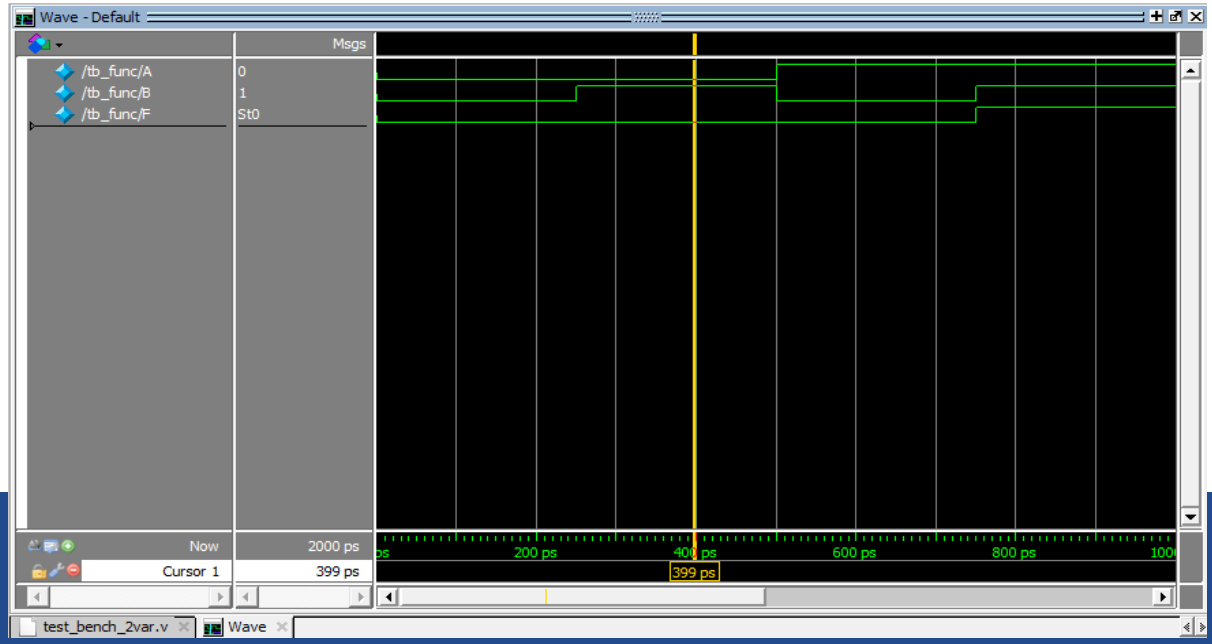
```
Ln#
1   `include "C:/intelFPGA_pro/17.0/data/and.v"
2           module tb_func;
3           reg A = 1'b0;
4           reg B = 1'b0;
5           wire F;
6           func_and M1 (F,A,B);
7
8   initial
9           begin
10          #2000;
11          $stop;
12          end
13  initial
14          begin
15          #250 A = 1'b0; B = 1'b1;
16          #250 A = 1'b1; B = 1'b0;
17          #250 A = 1'b1; B = 1'b1;
18          end
19  endmodule
20
```

reg → register
1'b0 → 1 bit, binary code, value 0

#2000 means run 2000 time units and then stop (total simulation generates waveforms over an interval of 2000 ps)

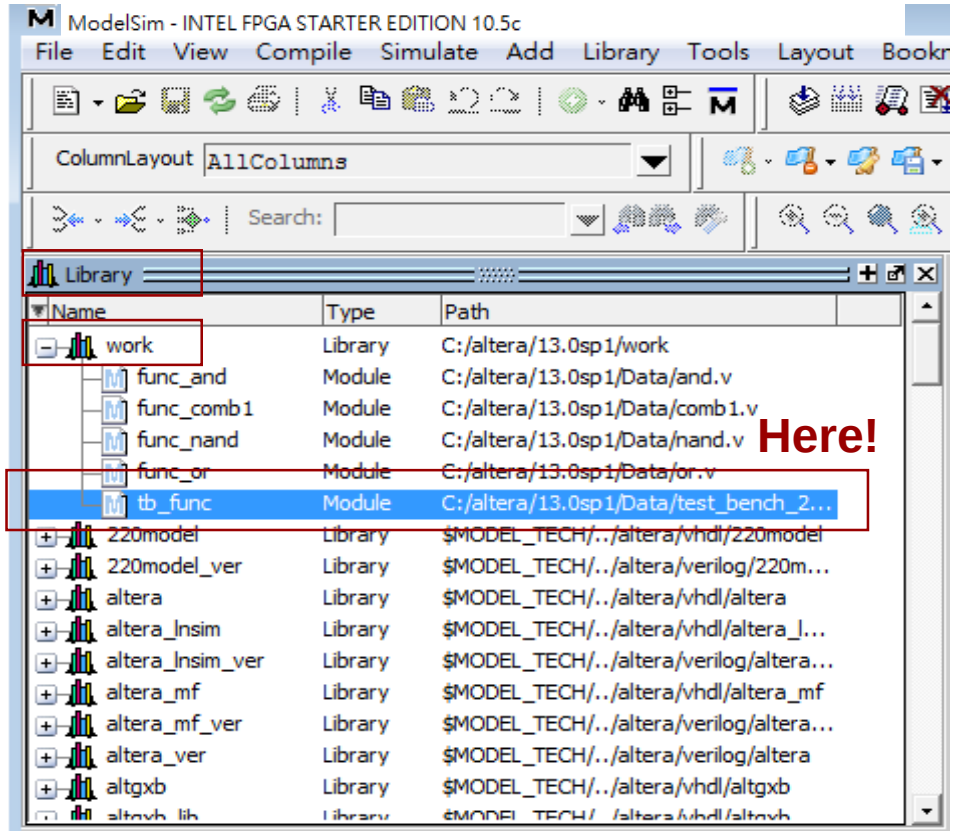
After 250 time units A is 0 and B is 1

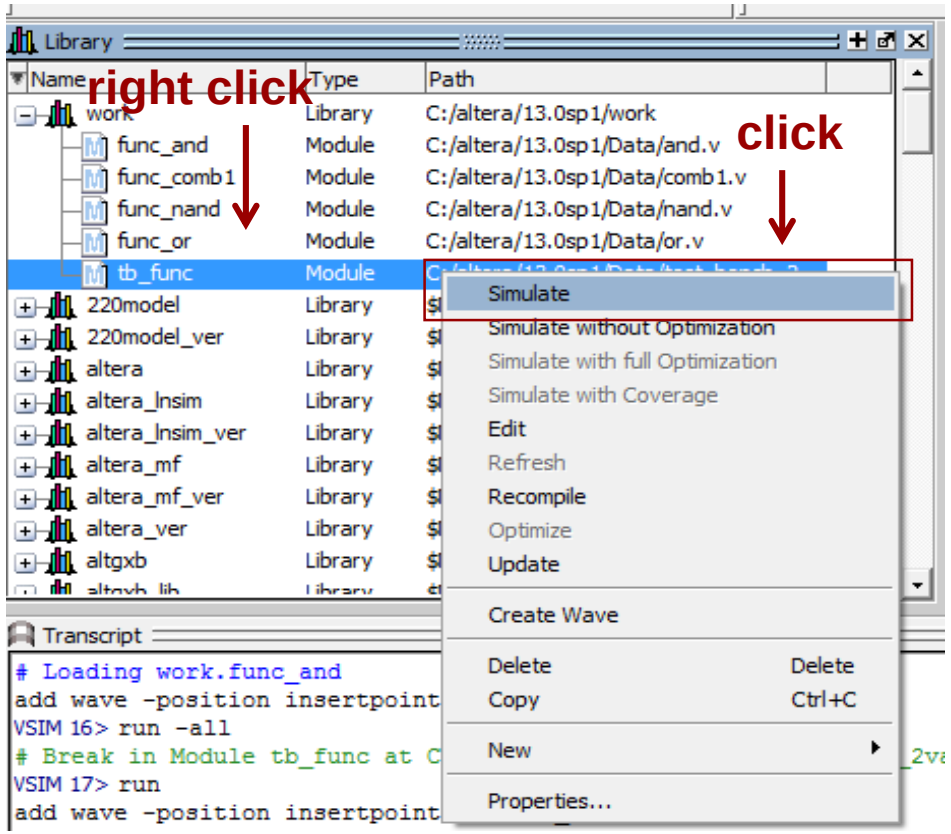
- **Understand the script**
- **Save it as "test_bench_2var.v"**



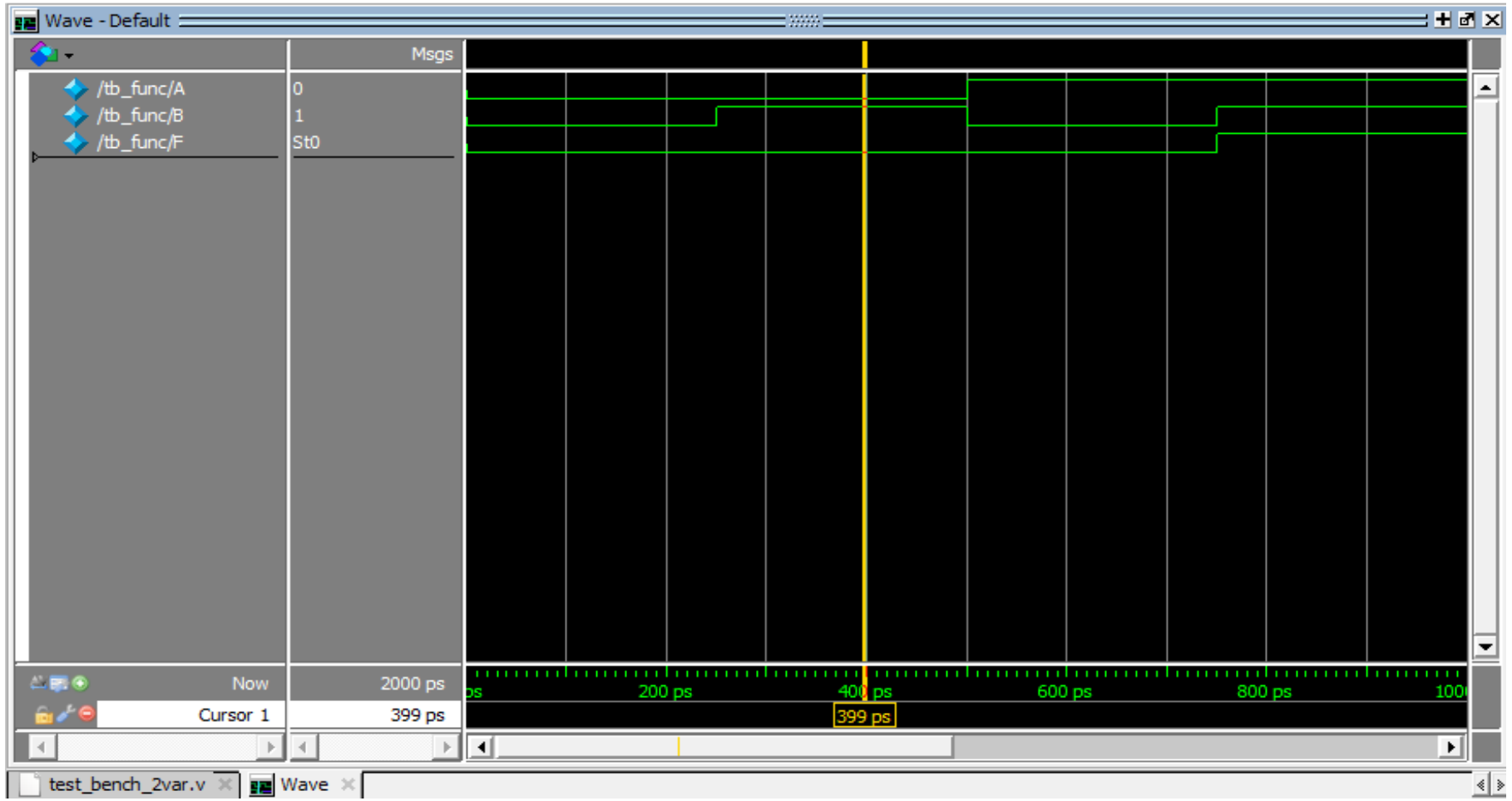
Simulate the Test Bench

- After compiling, go to Library “work” and find the module called “tb_func”, which you defined in your test bench.
- Right click tb_func and click on “simulate”.





Waveforms


Wave - Default
Msgs
/tb_func/A
/tb_func/B
/tb_func/F
0
1
St0
Now
2000 ps
Cursor 1
399 ps
200 ps
600 ps
800 ps
399 ps
test_bench_2var.v
Wave

Data Flow Modeling

- A Boolean function can be directly described in Verilog. You do not need to care about gate-level logic.
- Here is an example of how do you write an “AND” operation in Verilog using the command “assign”.

```
Ln#
1   module func_and(F,A,B);
2     input A,B;
3     output F;
4     assign F=A&B;
5   endmodule
6
```

- The operator “&” means “AND” operation.

Example:

$$F = B'C + BC'A' + (A \oplus C)$$

```
assign F = (~B&C)|(B&~C&~A)|(A^C);
```

Behavioral Modeling

- Behavioral Modeling uses procedural assignment statements with the keyword “always”. It is used mostly to describe sequential circuits.

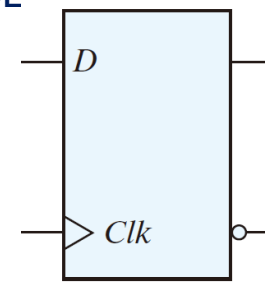
always @(A or B or select)

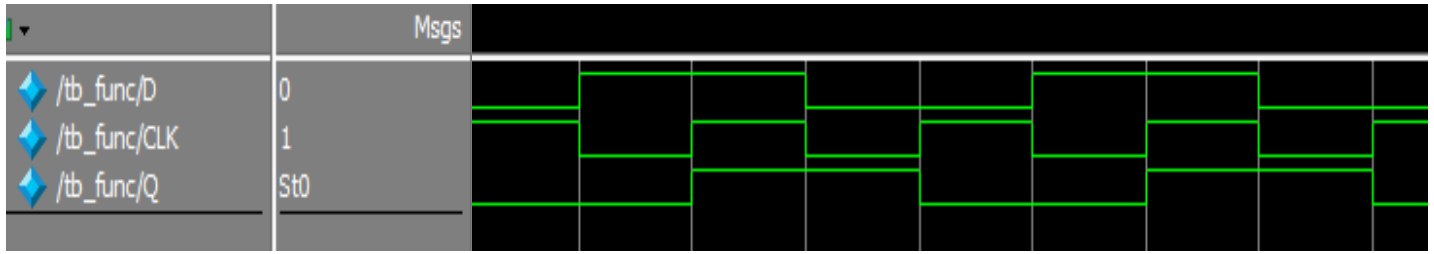
The procedural assignment statements inside the always block are executed every time there is a change in any of the variables listed after the @ symbol.

➢ **DFF Module**

```
1  module DFF(Q,D,CLK);
2    input D,CLK;
3    output Q;
4    reg Q;
5  always @(posedge CLK)
6  Q<=D;
7  endmodule
```

always @(posedge CLK)→ This statement will initiate execution of the procedural statements in the associated always block if a change occurs in posedge CLK





Verilog simulation

- To complete a verilog simulation, we need to do following steps:

1. Create a file for a main logic Verilog script, named “xxx.v”.
2. Create a test bench file, also named “zzz.v”, to test your script.
3. Compile each script → check if scripts are correct.
4. Simulate the test bench.
5. Add Wave and Run to evaluate timing signals.
6. Use wave function to check the results.

The End

Reference:

1. **Digital Design (with an introduction to the Verilog HDL) 6th Edition, M. Morris Mano, Michael D. Ciletti**