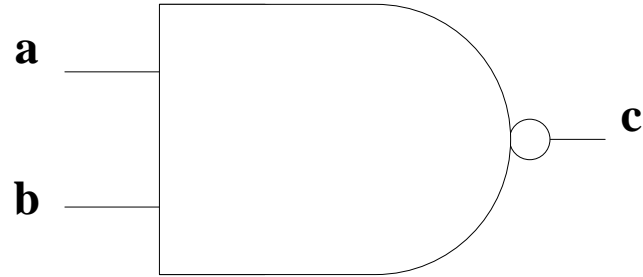
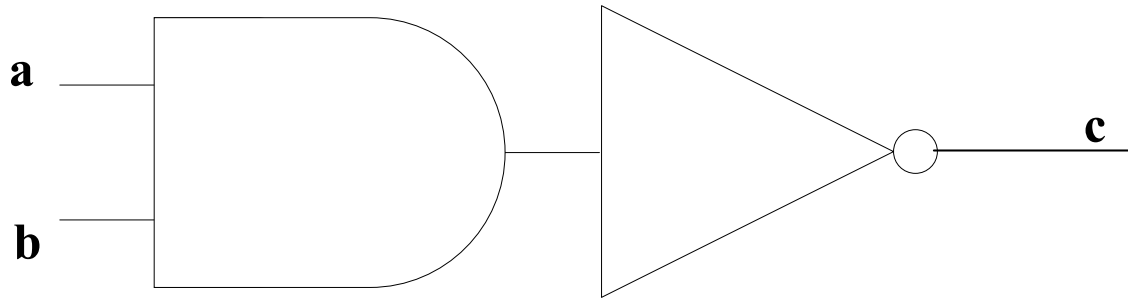


Verilog class 1

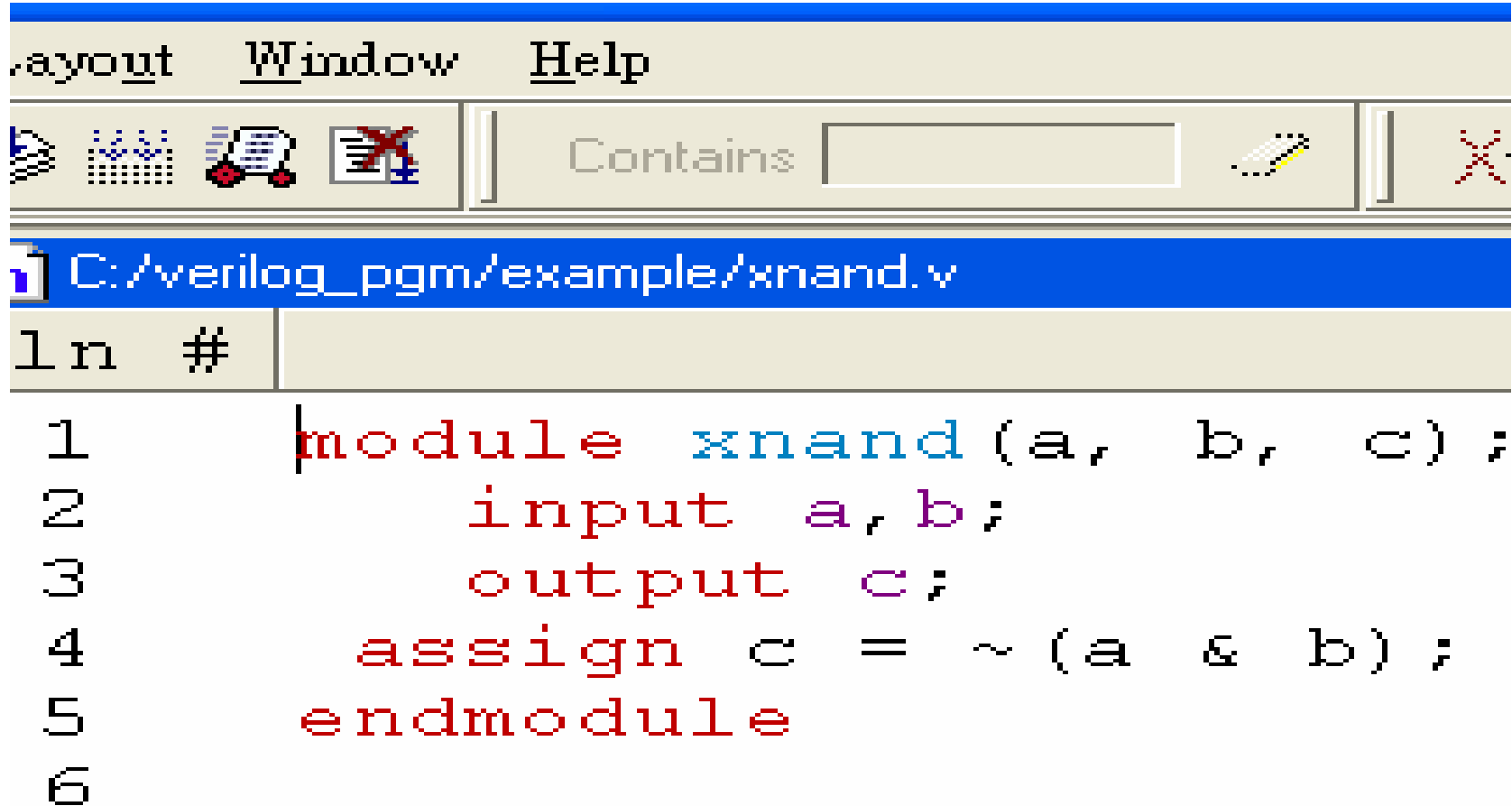
NAND



||



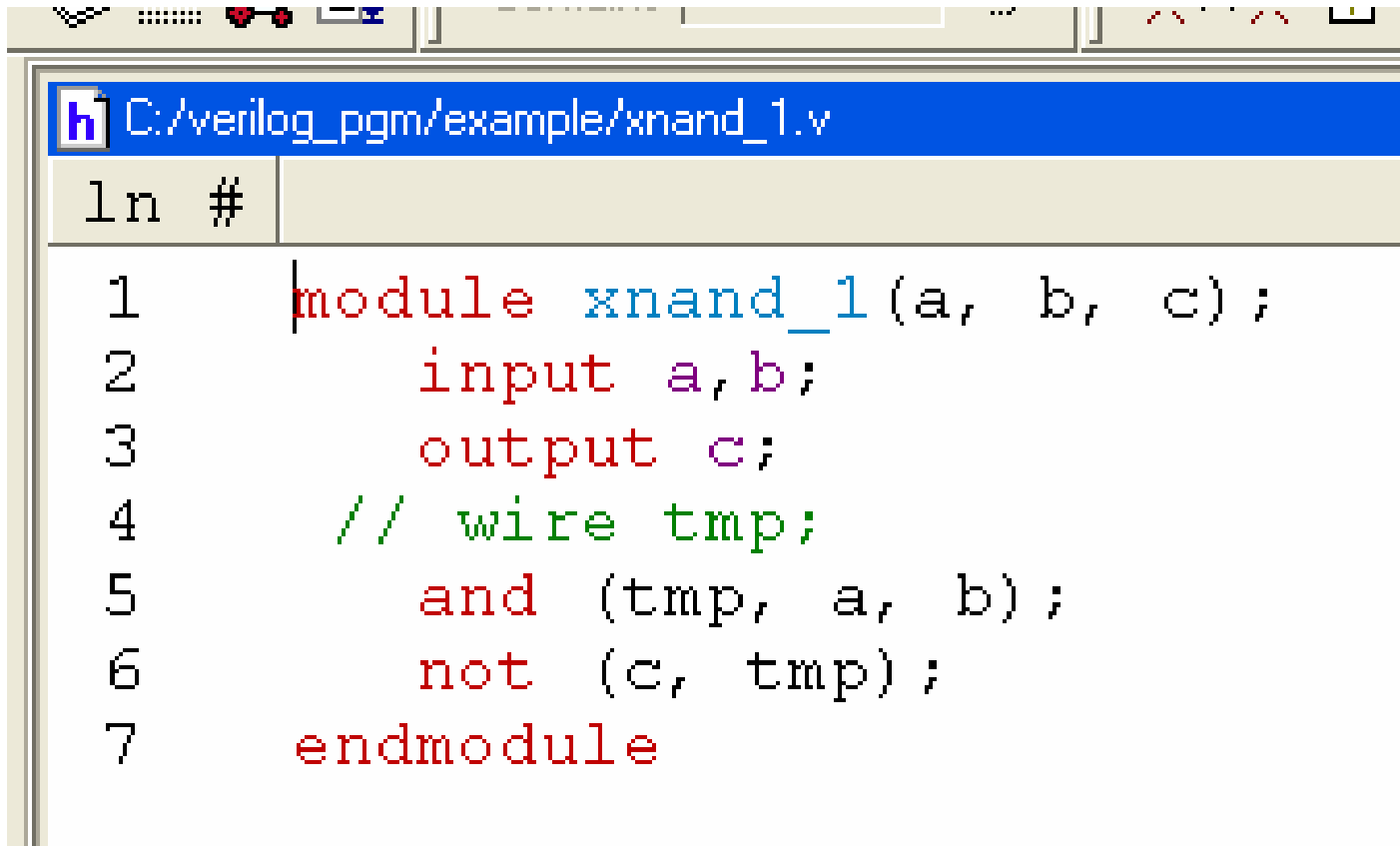
NAND: Behavior(行爲)



The image shows a screenshot of a Verilog editor window. The title bar contains the text "Layout Window Help". Below the title bar is a toolbar with several icons, including a grid, a document with a red X, and a pencil. The main window displays the following Verilog code:

```
1  module xnand(a, b, c);  
2      input a, b;  
3      output c;  
4      assign c = ~(a & b);  
5  endmodule  
6
```

NAND: Structure(結構)



The image shows a screenshot of a Verilog code editor window. The title bar indicates the file path is C:/verilog_pgm/example/xnand_1.v. The editor displays a Verilog module definition for a NAND gate. The code is as follows:

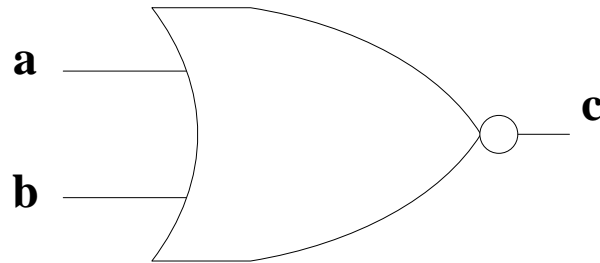
```
ln #  
1  module xnand_1(a, b, c);  
2      input a, b;  
3      output c;  
4      // wire tmp;  
5      and (tmp, a, b);  
6      not (c, tmp);  
7  endmodule
```

運算子(Operators)

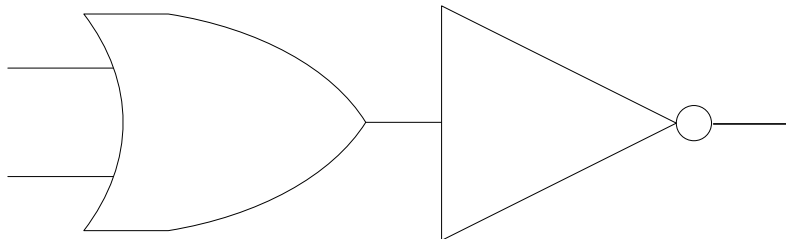
Arithmetic Operators	+, -, *, /, %
Relational Operators	<, <=, >, >=
Logical Equality Operators	==, !=
Case Equality Operators	===, !==
Logical Operators	!, &&,
Bit-Wise Operators	~, &, , ^(xor), ~^(xnor)
Unary Reduction Operators	&, ~&, , ~ , ^, ~^
Shift Operators	>>, <<
Conditional Operators	? :
Concatenation Operator	{ }
Replication Operator	{ { } }

Home Work #1

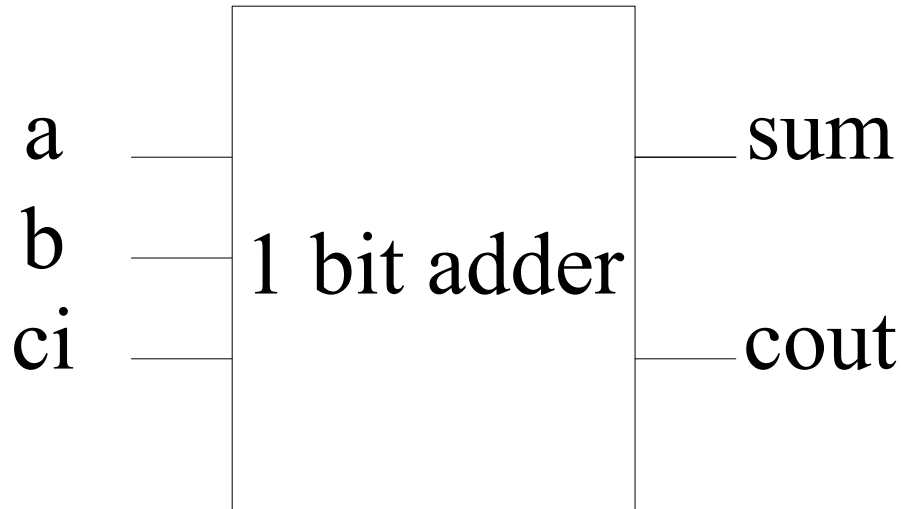
- 使用 or 閘建構 nor
- 使用Simulation驗證電路正確



||



One bit adder



$$\text{sum} = a \otimes b \otimes ci$$

$$\text{cout} = a \cdot b + a \cdot ci + b \cdot ci$$

One bit adder: Behavior(行爲)

C:/verilog_pgm/example/addbit.v *

ln #

```
1  module addbit (a, b, ci, sum, cout);  
2      input a, b, ci;  
3      output sum, cout;  
4      assign {cout, sum} = a + b + ci;  
5  endmodule
```

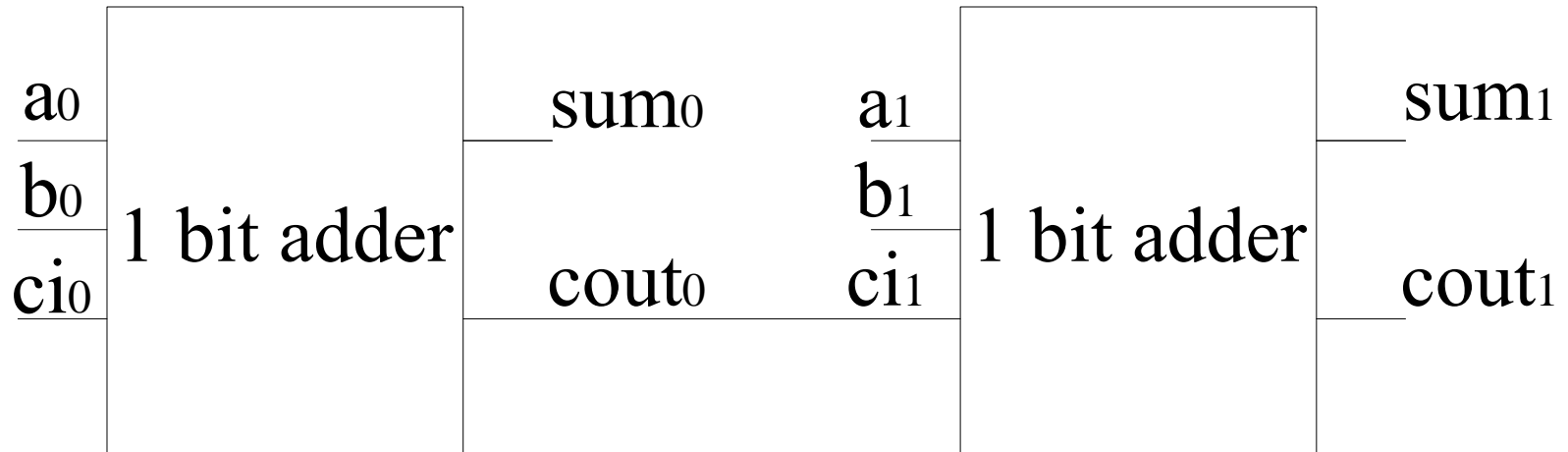

One bit adder: Structure(結構)

h C:/verilog_pgm/example/addbit_1.v

ln	#
----	---

1	<code>module addbit_1(a, b, ci, sum, cout);</code>
2	<code>input a, b, ci;</code>
3	<code>output sum, cout;</code>
4	<code>xor(sum, a, b, ci);</code>
5	<code>and (tmp1, a, b),</code>
6	<code>(tmp2, a, ci),</code>
7	<code>(tmp3, b, ci);</code>
8	<code>or (cout, tmp1, tmp2, tmp3);</code>
9	<code>endmodule</code>

Two bit adder



Two bit adder: Behavior(行爲)

C:/verilog_pgm/example/add_2_bits.v

ln #

```
1  module add_2_bits(a, b, ci, cout, sum);
2      input [1:0] a, b;
3      input ci;
4      output [1:0] sum, cout;
5      assign {cout[0], sum[0]} = a[0] + b[0] + ci;
6      assign {cout[1], sum[1]} = a[1] + b[1] + cout[0];
7  endmodule
```

Two bit adder : Structure(結構)

C:/verilog_pgm/example/add_2_bits_1.v

```
ln #
1   include addbit.v;
2   module add_2_bits_1(a, b, ci, cout, sum);
3       input [1:0] a,b;
4       input ci;
5       output [1:0] sum,cout;
6       addbit u1(a[0], b[0], ci, sum[0], cout[0]);
7       addbit u2(a[1], b[1], cout[0], sum[1], cout[1]);
8   endmodule
9
```

HW#2

- 建構 4位元加法器
- 使用Simulation驗證電路正確