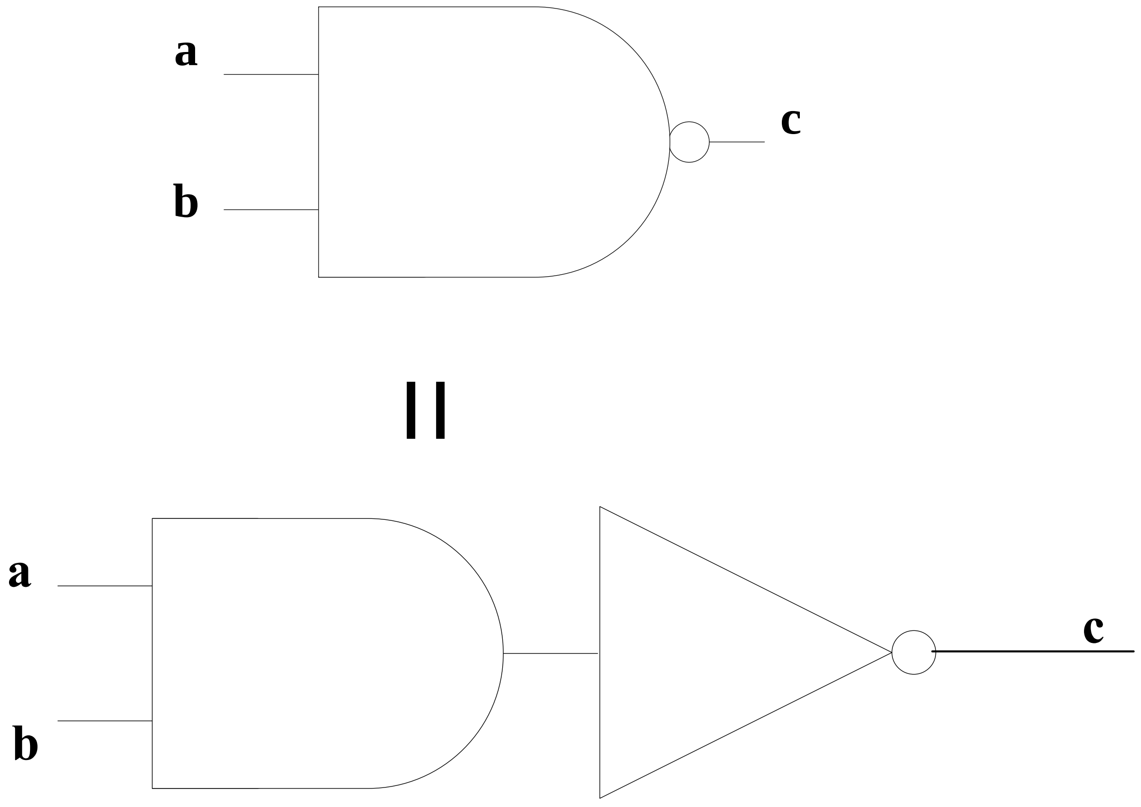
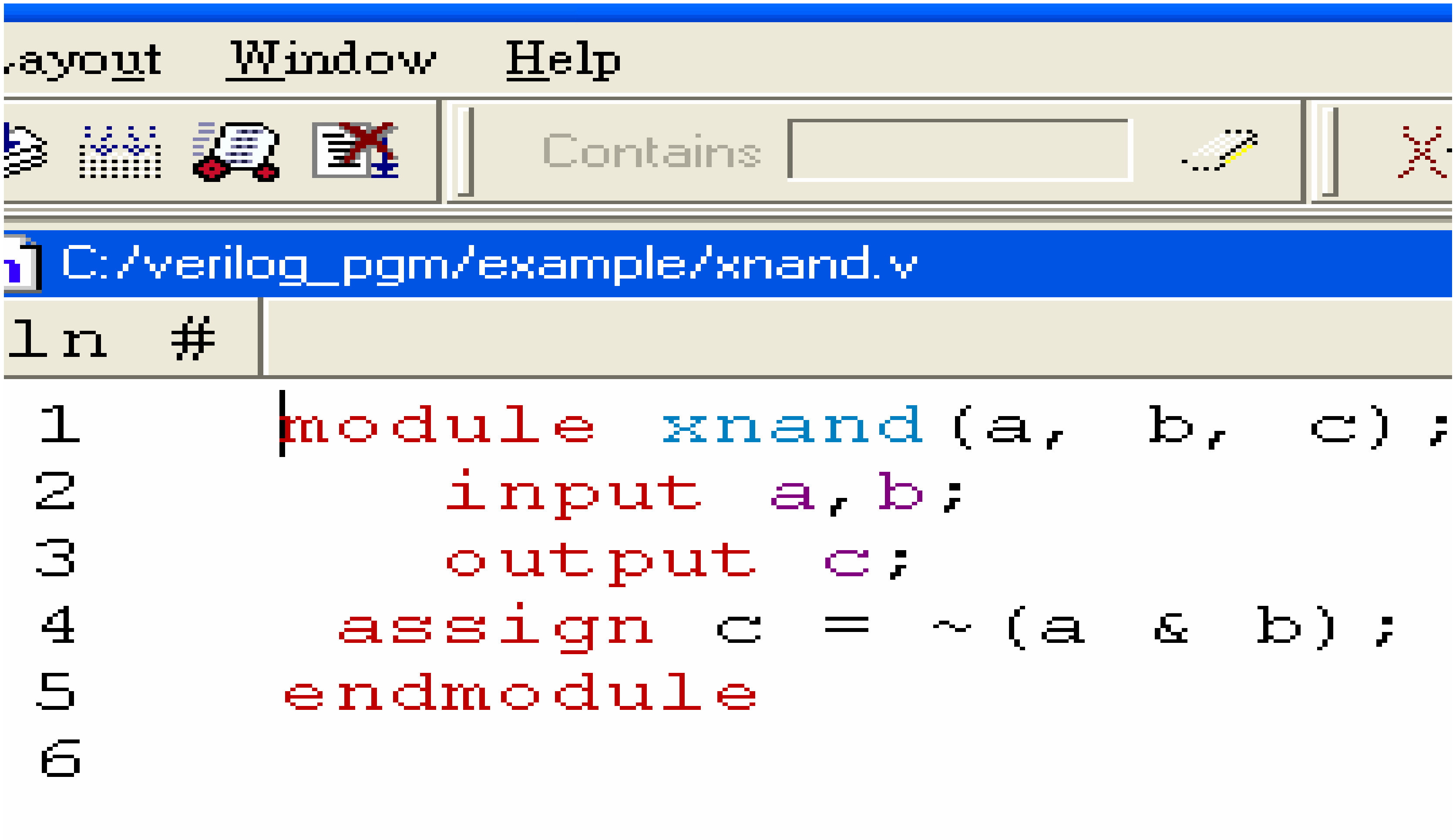
Verilog class 1

NAND

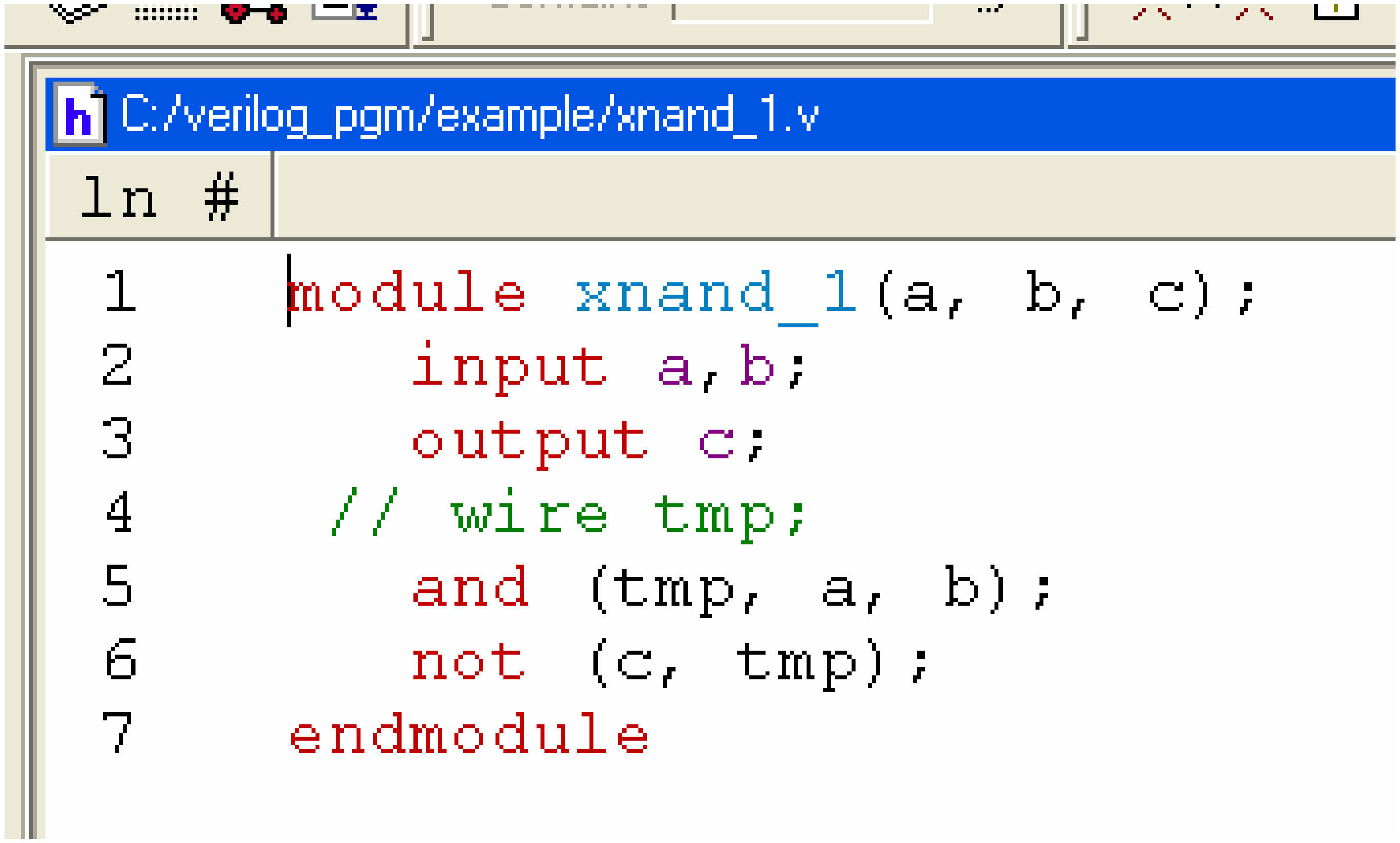


NAND: Behavior(行爲)



```
module xnand(a, b, c);
     input a,b;
     output c;
  assign c = ~(a & b);
endmodule
```

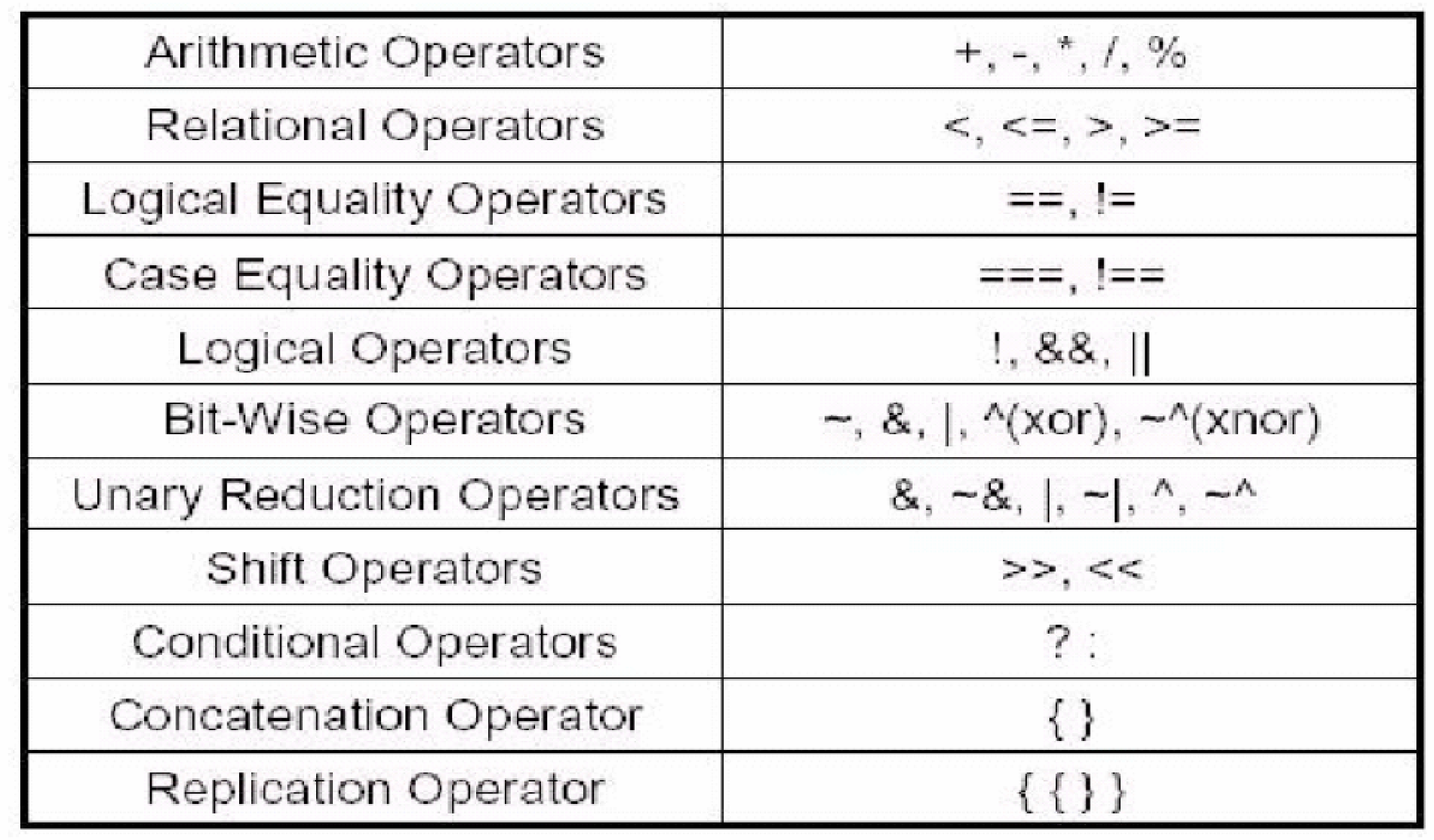
NAND: Structure(結構)


C:/verilog_pgm/example/xnand_1.v

ln #

```
module xnand_1(a, b, c);
   input a,b;
   output c;
 // wire tmp;
   and (tmp, a, b);
   not (c, tmp);
endmodule
```

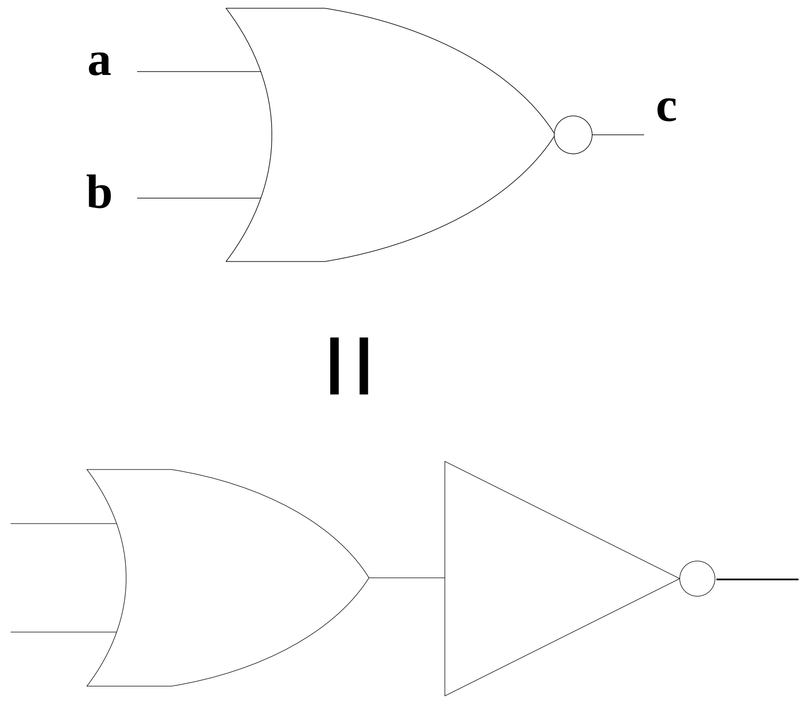
運算子(Operators)



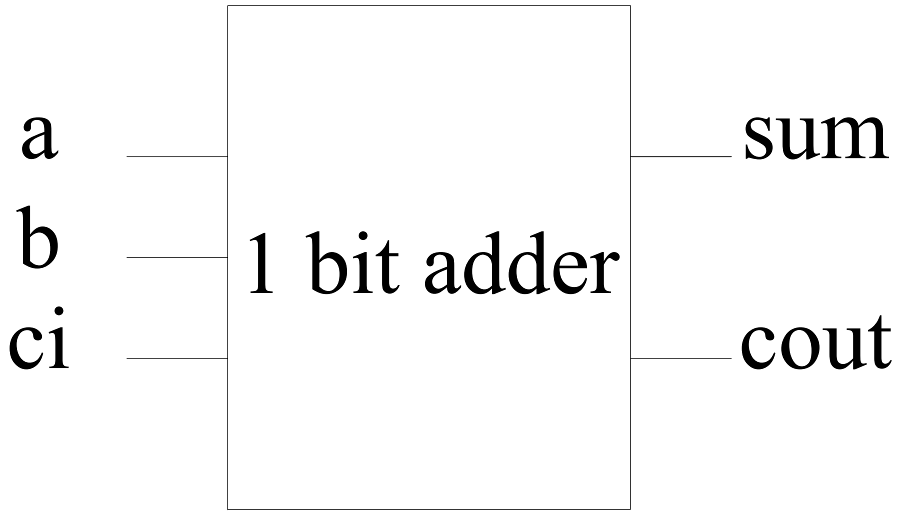
Arithmetic Operators	+, -, *, /, %
Relational Operators	<, <=, >, >=
Logical Equality Operators	==, !=
Case Equality Operators	===, !==
Logical Operators	!, &&, ||
Bit-Wise Operators	~, &, |, ^(xor), ~^(xnor)
Unary Reduction Operators	&, ~&, |, ~|, ^, ~^
Shift Operators	>>, <<
Conditional Operators	?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

Home Work #1

- 使用 or 閘建構 nor
- 使用Simulation驗證電路正確



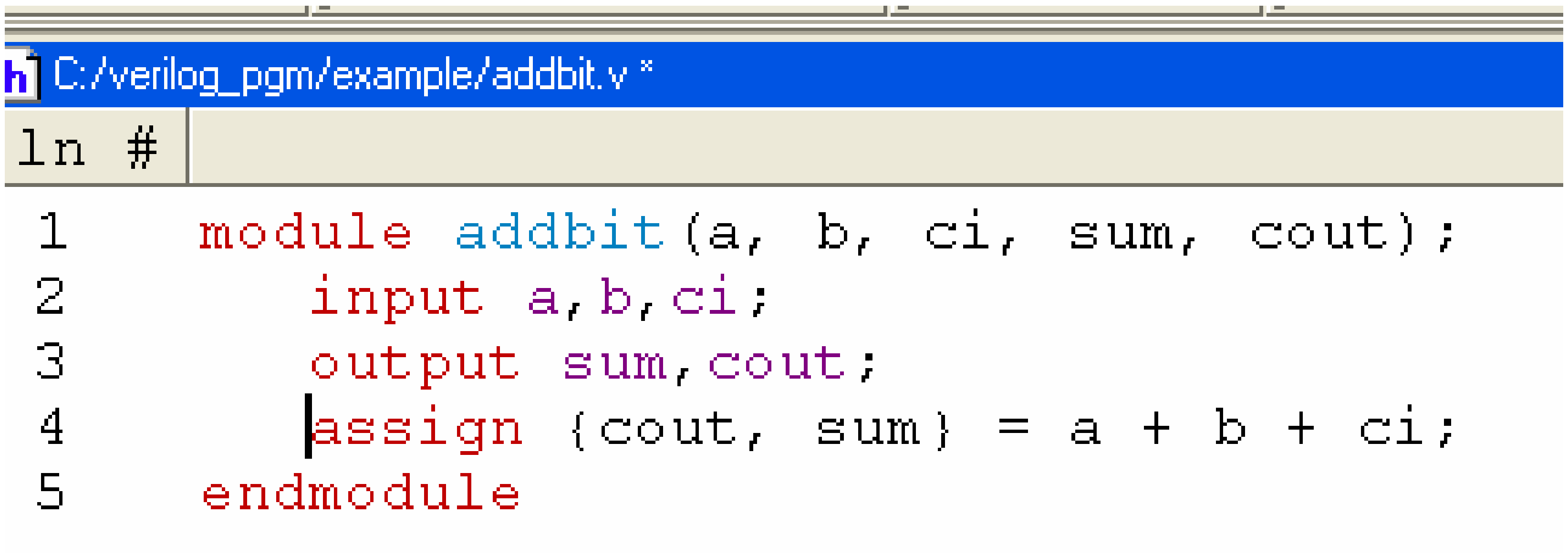
One bit adder



$$sum = a \otimes b \otimes ci$$

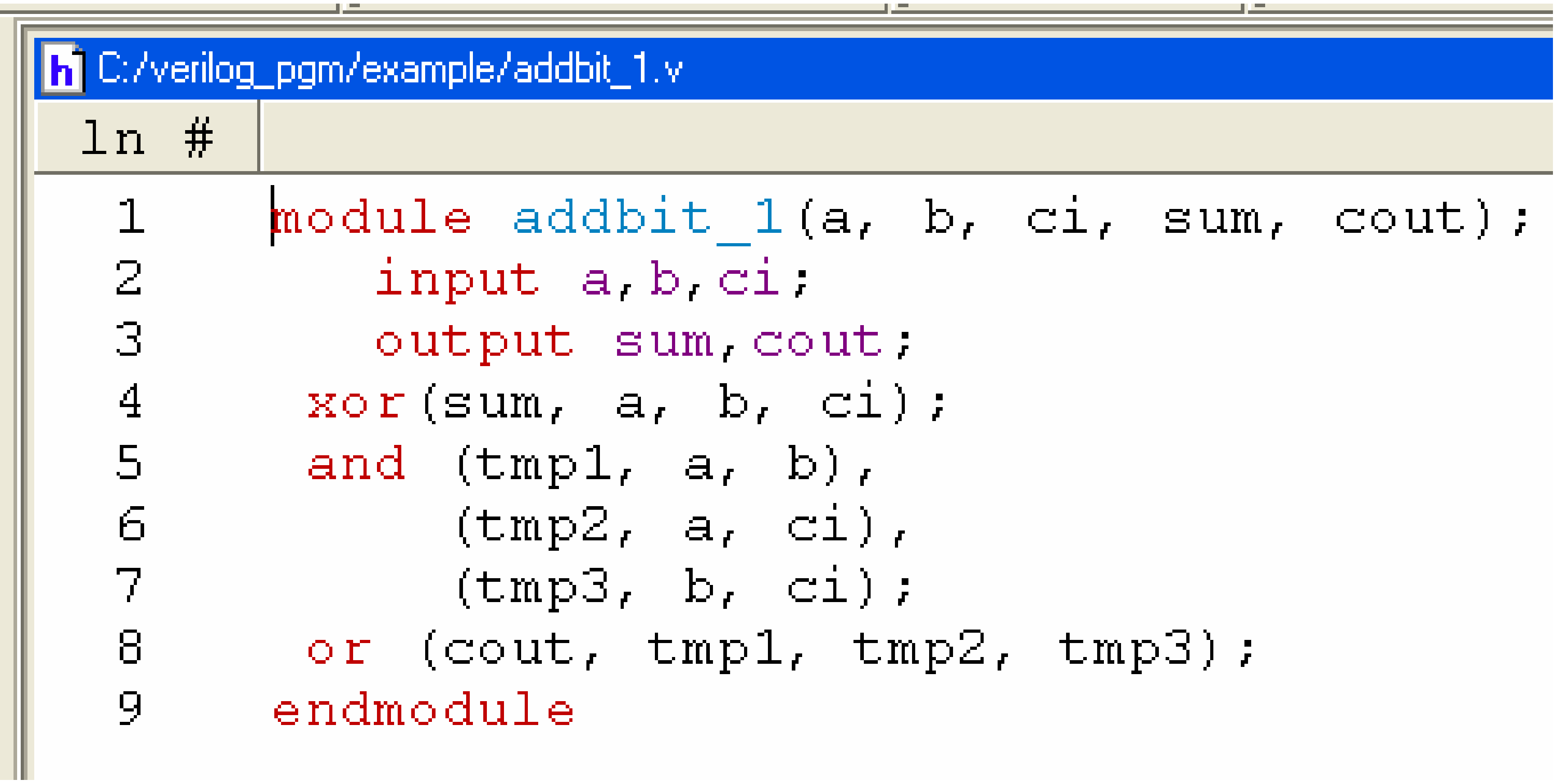
$$cout = a \bullet b + a \bullet ci + b \bullet ci$$

One bit adder: Behavior(行爲)



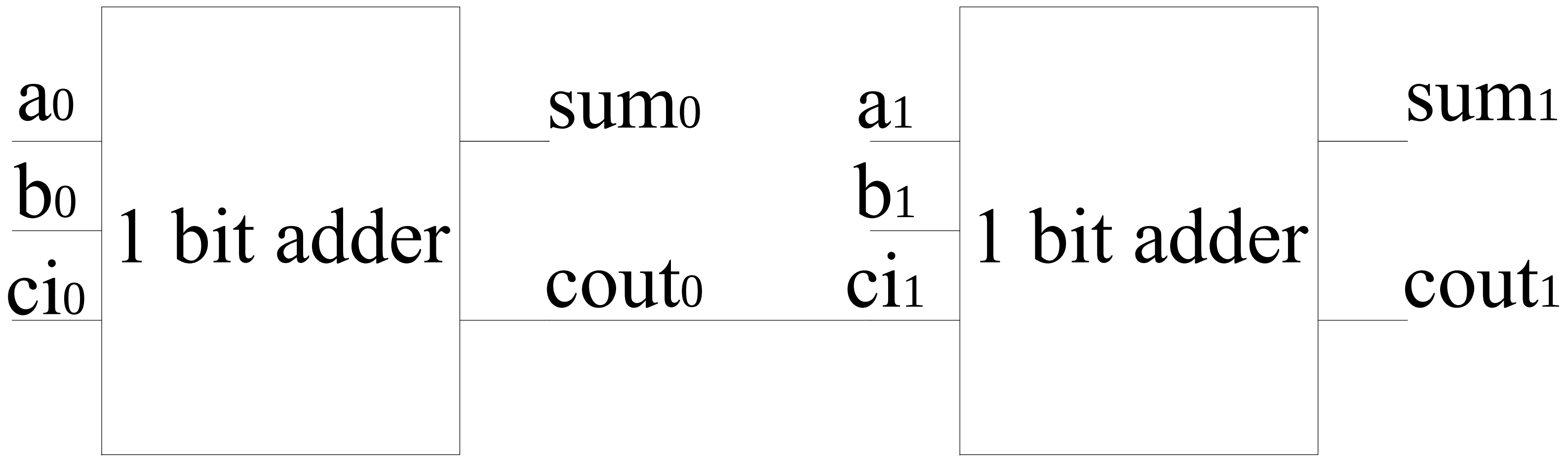
```
module addbit(a, b, ci, sum, cout);
    input a,b,ci;
    output sum,cout;
    assign {cout, sum} = a + b + ci;
endmodule
```

One bit adder: Structure(結構)

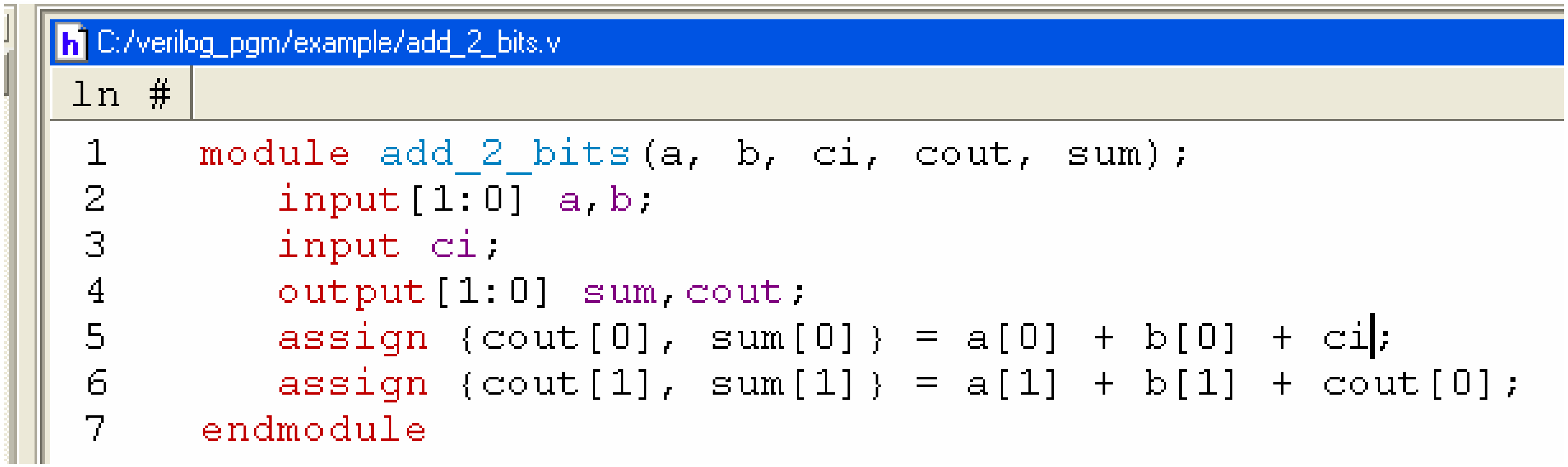

C:/verilog_pgm/example/addbit_1.v

```
module addbit_1(a, b, ci, sum, cout);
     input a,b,ci;
     output sum,cout;
  xor(sum, a, b, ci);
  and (tmp1, a, b),
      (tmp2, a, ci),
      (tmp3, b, ci);
  or (cout, tmp1, tmp2, tmp3);
endmodule
```

Two bit adder

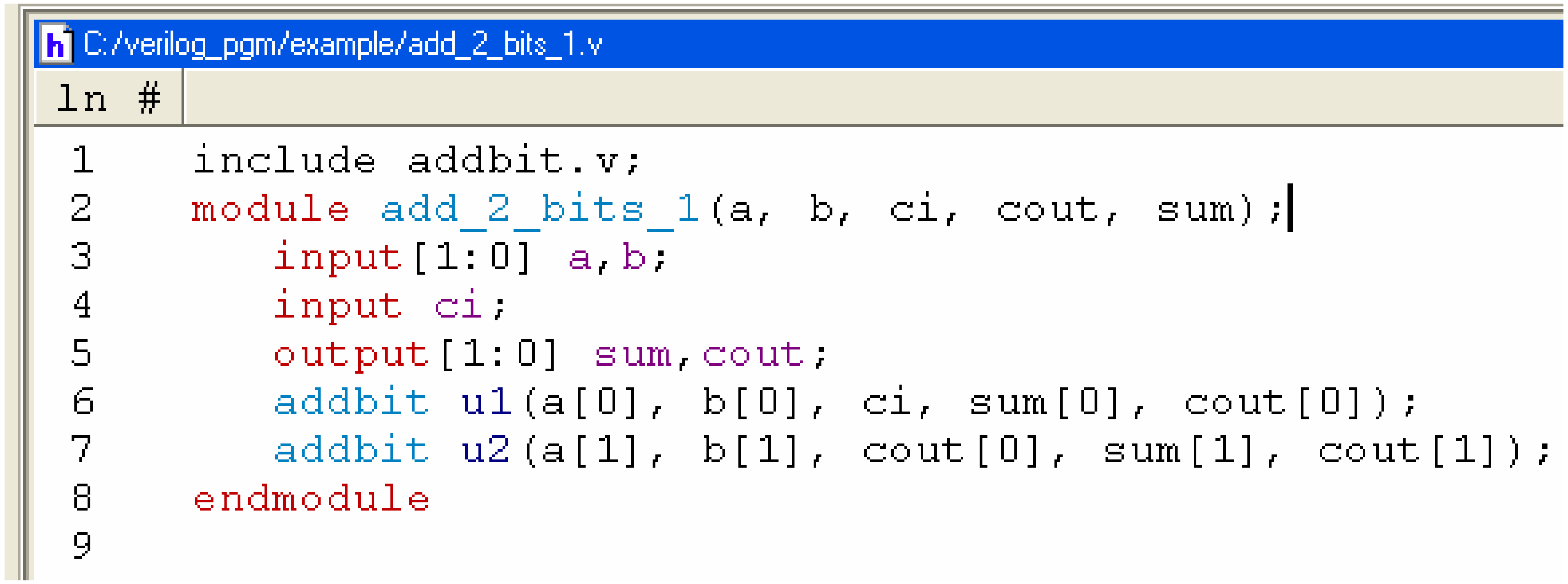


Two bit adder: Behavior(行爲)



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

Two bit adder : Structure(結構)



C:/verilog_pgm/example/add_2_bits_1.v

ln #

```
include addbit.v;
module add_2_bits_1(a, b, ci, cout, sum);
   input[1:0] a,b;
   input ci;
   output[1:0] sum,cout;
   addbit u1(a[0], b[0], ci, sum[0], cout[0]);
   addbit u2(a[1], b[1], cout[0], sum[1], cout[1]);
endmodule

```

HW#2

- 建構 4位元加法器
- 使用Simulation驗證電路正確