Learning Objectives

1. Introduction to Verilog syntax
2. Introduction to Verilog tools (iverilog, gtkwave)
3. Practices with Verilog coding, testing, and debugging

Work that needs to be handed in

1. Implement the `sc2_block` in verilog.

   Using the `sc_block` module we provide (shown on page 2 of this handout, in SVN in `sc_block.v`) implement the `sc2_block` module as described on the last page of this handout.

   Your code should be placed in the `sc2_block.v` file in SVN; **it should be the only module in that file.**

   In the file `sc2_block_tb.v`, write a test bench for `sc2_block` using `sc_block_tb.v` as a guideline.

   You should be able to compile, run, and debug your code by running:

   ```
   iverilog -o sc2 sc2_block_tb.v sc2_block.v sc_block.v
   ./sc2
   gtkwave sc2.vcd
   ```

   The final command assumes that you use the command $dumpfile("sc2.vcd"); in your test bench.

   Make sure you check in both `sc2_block.v` and `sc2_block_tb.v` into SVN to handin.

2. Test a "black box" module to determine its truth table. In your SVN repository you have a circuit called blackbox.v. This verilog module was designed to be not easily analyzed by visual inspection. You should write a test bench that allows you to discover the truth table for this module and fill in the truth table in the blackbox.tt file.

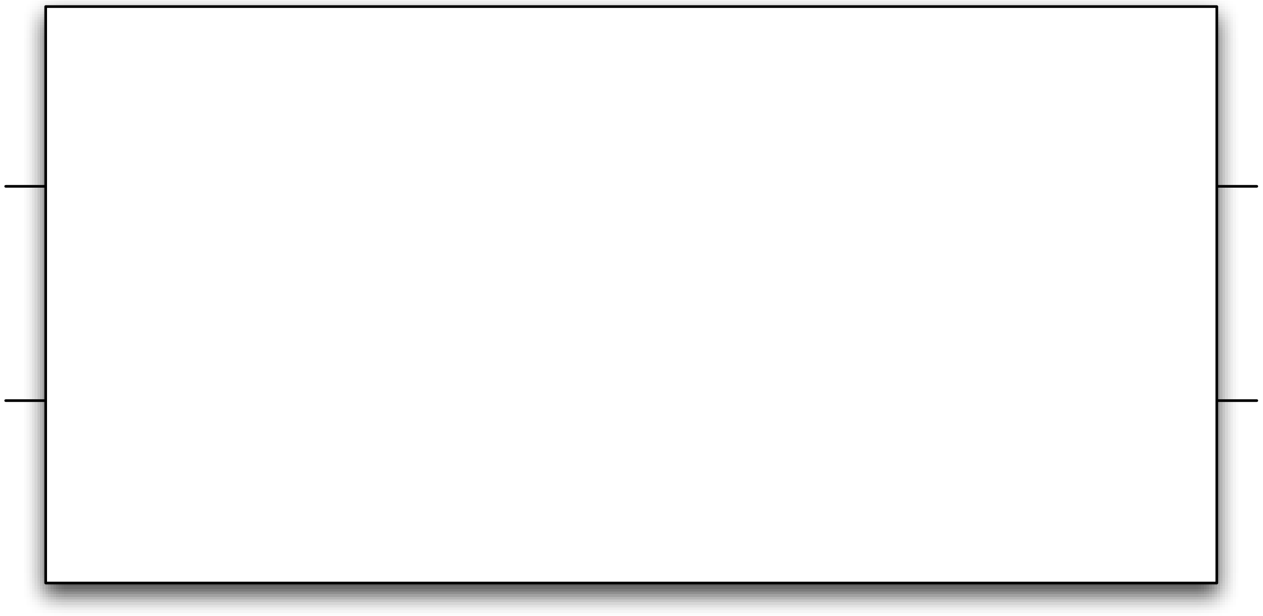
Verilog Modules

Truth Table

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

```
module sc_block(s, c, a, b);
   output s, c;
   input  a, b;
   wire   w1, w2, not_a, not_b;

   // the "c" output is just the AND of the two inputs
   and a1(c, a, b);

   // the "s" output is 1 only when exactly one of the inputs is 1
   not n1(not_a, a);
   not n2(not_b, b);
   and a2(w1, a, not_b);
   and a3(w2, b, not_a);
   or  o1(s, w1, w2);

endmodule // sc_block
```

Draw a circuit diagram for the Verilog module above:



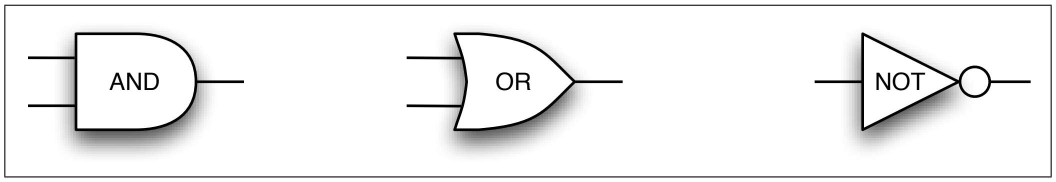
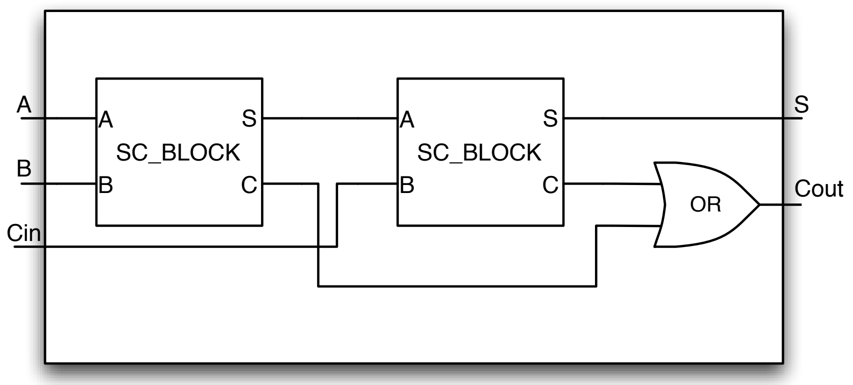


Figure 1. AND, OR, and NOT gates.

Verilog Test Bench

```
module sc_test;

   reg a = 0, b = 0;          // these are inputs to "circuit under test"
                              // use "reg" not "wire" so can assign a value
                              // we've given them initial values

   initial begin              // initial = run at beginning of simulation
                              // begin/end = associate block with initial

      $dumpfile("sc.vcd");    // name of dump file to create
      $dumpvars(0,sc_test);   // record all signals of module "sc_test"
                              // starting at time 0

      # 10                    // wait 10 time units
      a = 1;                  // change a's value
      # 10 a = 0; b = 1;
      # 10 a = 1;
      # 10
      $finish;                // end the simulation
   end

   wire s, c;                 // wires for the outputs of "circuit under test"
   sc_block sc1 (s, c, a, b); // the circuit under test


   initial
     $monitor("At time %t, a = %d b = %d s = %d c = %d",
              $time, a, b, s, c);
endmodule // test
```

Hierarchical design: building modules from modules



A	B	Cin	S	Cout
0	0	0	0	0
0	0	1		
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1		

Figure 2. The sc2_block, which is implemented using sc_blocks and its partially completed truth table.

Write the Verilog for the `sc2_block` module: