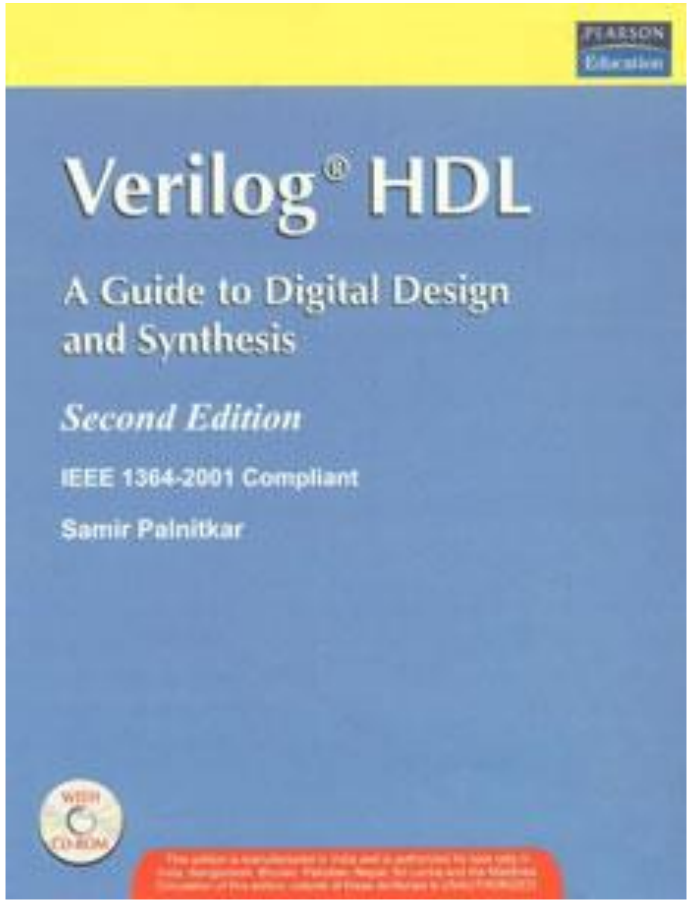
# Verilog HDL

MODULES AND PORTS

## Reference

Verilog HDL: A Guide to Digital Design and Synthesis, 2e, Samir Palnitkar

**Chapters 4**

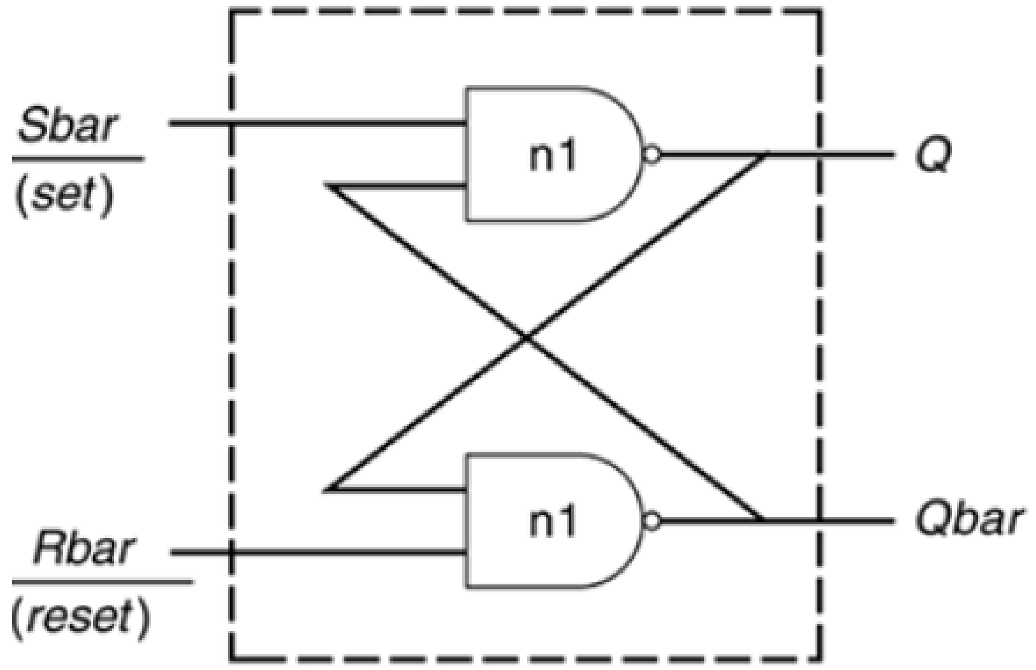


## Verilog Modules

- So far we looked at how modules are defined and instantiated.

- Now lets look at the internals of a module.

## Components of a module : an example



An SR Latch

## Components of a module : an example

```
// Module name and port list
// SR_latch module
module SR_latch(Q, Qbar, Sbar, Rbar);

    //Port declarations
    output Q, Qbar;
    input Sbar, Rbar;

    // Instantiate lower-level modules // In this case,
    instantiate Verilog primitive nand gates // Note, how
    the wires are connected in a cross-coupled fashion.
    nand n1(Q, Sbar, Qbar);
    nand n2(Qbar, Rbar, Q);

// endmodule statement
endmodule
```

## Components of a module : an example

```
// Module name and port list : Stimulus module
module Top;
    // Declarations of wire, reg, and other variables
    wire q, qbar; reg set, reset;

    // Instantiate lower-level modules. In this case, instantiate SR_latch
    // Feed inverted set and reset signals to the SR latch
    SR_latch m1(q, qbar, ~set, ~reset);

    initial // Behavioural block,initial
    begin
        $monitor($time, " set = %b, reset= %b, q= %b\n",set,reset,q);
        set = 0; reset = 0;
        #5 reset = 1;
        #5 reset = 0;
        #5 set = 1;
        end

// endmodule statement
endmodule
```

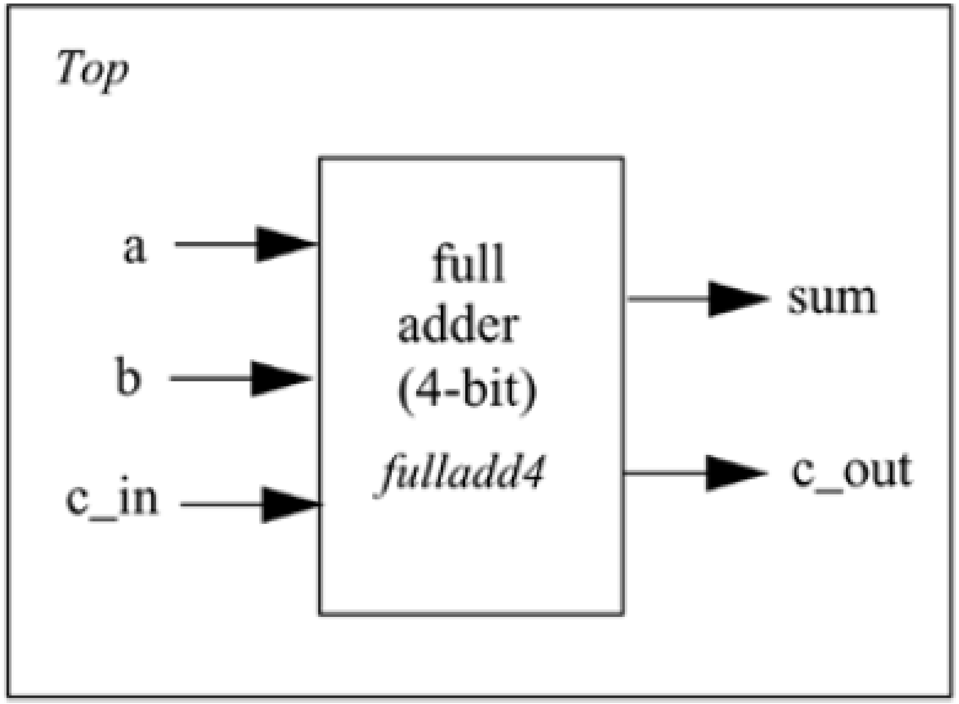
## Ports / terminals

- Ports provide the interface by which a module can communicate with its environment.
  - For example, the input/output pins of an IC chip are its ports.
- The environment can interact with the module only through its ports.
- The internals of the module are not visible to the environment.
  - Provides flexibility to the designer : The internals of the module can be changed without affecting the environment as long as the interface is not modified.

## List of Ports

- A module definition contains an optional list of ports.
- If the module does not exchange any signals with the environment, there are no ports in the list.

## Example : I/O Ports for Top and Full Adder



- module *Top* is a top-level module
  - no communication with the environment
  - no ports
- The module *fulladd4* is instantiated below *Top*
  - Does addition for its environments and hence does communication
  - has ports

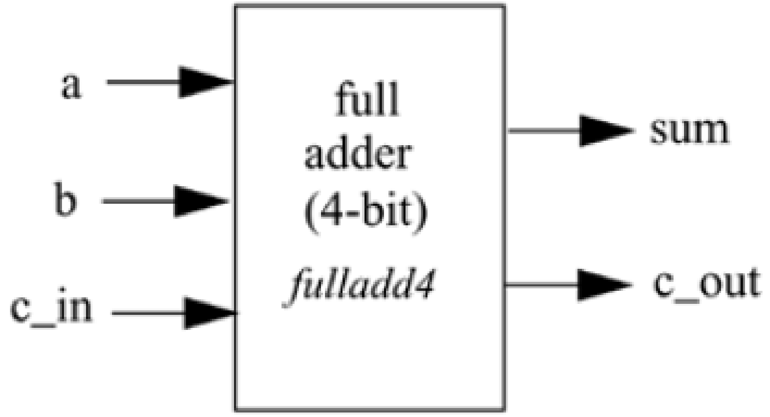
```
module fulladd4(sum, c_out, a, b, c_in); //Module with a list of ports
module Top; // No list of ports, top-level module in simulation
```

## Port declaration

- All ports in the list of ports must be declared in the module.

| Verilog Keyword | Type of Port |
|---|---|
| input | Input port |
| output | Output port |
| inout | Bidirectional port |

# Port declaration : an example

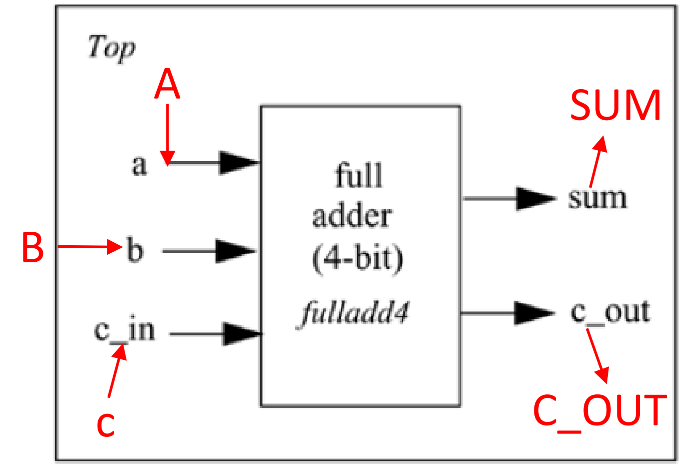


```
module fulladd4(sum, c_out, a, b, c_in);

   //Begin port declarations section
   output[3:0] sum;
   output c_cout;
   input [3:0] a, b;
   input c_in;
   //End port declarations section

   ... <module internals> ...

endmodule
```

- Note that all port declarations are implicitly declared as wire in Verilog.

# Connecting ports to external signals

- The signals to be connected must appear in the module instantiation in the same order as the ports in the port list in the module definition.



```
module Top; //Declare connection variables
       reg [3:0] A,B; reg C_IN;
       wire [3:0] SUM; wire C_OUT;
       //Instantiate fulladd4, call it fa_ordered.
       //Signals are connected to ports in order (by positio
       fulladd4 fa_ordered(SUM, C_OUT, A, B, C_IN);
       ... <stimulus> ...
endmodule

module fulladd4(sum, c_out, a, b, c_in);
    output[3:0] sum; output c_cout;
    input [3:0] a, b; input c_in;
    ... <module internals> ...
endmodule
```