Textbook: Verilog® HDL 2nd. Edition
Samir Palnitkar
Prentice-Hall, Inc.

E-mail: kevinsu@yuntech.edu.tw
Chapter 4

Modules and Ports
4.1 Modules
4.2 Ports
4.3 Hierarchical Names
4.1 Modules

4.2 Ports

4.3 Hierarchical Names
### Distinct Parts of a Module in Verilog

<table>
<thead>
<tr>
<th>Module Name, Port Lists, Port Declaration (if ports present), Parameters (optional),</th>
<th>Declaration of wires, regs and other variables,</th>
<th>Data flow statements (assign)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantiation of lower level modules</td>
<td>always and initial blocks. All behavior statements go in these blocks.</td>
<td></td>
</tr>
<tr>
<td>Tasks and functions</td>
<td>endmodule statement</td>
<td></td>
</tr>
</tbody>
</table>
# Module Example: SR Latch

This example illustrates the different components of a module. The module name and port list is `SR_latch` module.

```verilog
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   //endmodule statement
```

The stimulus module is also included.

```verilog
module Top;
wire q, qbar;   // Declarations of wire, reg and other variables
reg set, reset;
// Instantiate lower level modules
// In this case, instantiate SR_latch
SR_latch l1(q, qbar, ~set, ~reset);
initial   // Behavioral 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;
    #5 set = 0; reset = 1;
    #5 set = 0; reset = 0;
    #5 set = 1; reset = 0;
end
endmodule   // endmodule statement
```
4.1 Modules

**SR Latch Simulation Results**
4.2 Ports

4.1 Modules
4.2 Ports
4.3 Hierarchical Names
4.2 Ports

- Ports provide the interface by which a module can communicate with its environment.
- The environment can interact with the module only through its ports.
- The internals of the module can be changed without affecting the environment as long as the interface is not modified.
- Ports are also referred to as terminals.
4.2 Ports

- **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.
  - Consider a 4-bit full adder that is instantiated inside a top-level module *Top*.
  - Example 4-bit full adders:
4.2 Ports

- Port Example for 4-bit Full Adder

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
## 4.2 Ports

### Port Declaration

- Ports can be declared as follows:

<table>
<thead>
<tr>
<th>Verilog Keyword</th>
<th>Type of Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>Input Port</td>
</tr>
<tr>
<td>output</td>
<td>Output Port</td>
</tr>
<tr>
<td>inout</td>
<td>Bidirectional Port</td>
</tr>
</tbody>
</table>

- **Example of `fulladd4`:**

  ```verilog
  module fulladd4 (sum, c_out, a, b, c_in);
  //Begin port declarations section
  output [3:0] sum;
  output c_out;
  input [3:0] a, b;
  input c_in; //End port declaration section
  ... <module internals> ...
  endmodule
  ```
All port declarations are implicitly declared as `wire` in Verilog.

If output ports hold their value, they must be declared as `reg`.

For example of DFF, we want the output `q` to retain its value until the next clock edge.

The port declarations for DFF will look as next slice:
4.2 Ports

- **Port Declaration** (Cont.)

module DFF(q, d, clk, reset);
output q;
reg q; //Output port q holds value, therefore it is declared as reg
input d, clk, reset;

:::

endmodule
4.2 Ports

- **Port Declaration** (Cont.)

- **input** and **inout** cannot be declared as reg because reg variables store values and input ports should not store values.

- **fulladd4** declaration in ANSI C Style:

```verbatim
gcd module fulladd4 (output reg [3:0] sum,
   output reg c_out,
   input   [3:0] a, b, //wire by default
   input   c_in; // wire by default

   ...
<module internals>

   ...
endmodule
```
4.2 Ports

Port Connection Rules

- **Width Matching**: It is legal to connect internal and external items of different sizes.

- **Mismatching of Width**: Warning Message!!
Example of Illegal Port Connection

- This problem is rectified if the variable `SUM` is declared as a net (`wire`).

```verbatim
module Top
  // Declare connection variable
  reg [3:0] A, B;
  reg C_IN;
  reg [3:0] SUM;
  wire C_OUT

  // Instantiate fulladd4, call it fa0
  fulladd4 fa0(SUM, C_OUT, A, B, C_IN);
  // Illegal connection because output port sum in module fulladd4 is connect to a register variable SUM in module Top
  <stimulus> ...
endmodule
```
4.2 Ports

- Connecting Ports to External Signals
  1. Connecting by ordered list
  2. Connecting ports by name
4.2 Ports

- **Connecting Ports by Ordered List**
  - Connecting by ordered list is the most intuitive method for most beginners.
  - 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.
4.2 Ports

Example for Connecting Ports by Ordered List

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 position)
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
Connecting Ports by Name

- For large designs where modules have remembering the order of the ports in the module definition is impractical and error-prone.
- Verilog provides the capability to connect external signals to ports by the port names.
- You can specify the port connections in any order as long as the port name in the module definition correctly matches the external signal.
- Unconnected ports can be dropped.
- Advantage of connecting ports by name is that the port name is not changed, the order of ports in the port list of a module can be rearranged without changing the port connections in module instantiations.
Example for Connecting Ports by Name

module Top;  // Declare connection variables
reg [3:0] A, B;
reg C_IN;
wire [3:0] SUM;
wire C_OUT;

// Instantiate module fa_byname and connect signals to ports by name
fulladd4 fa_byname (.c_out(C_OUT), .sum(SUM), .b(B), .c_in(C_IN), .a(A));

...  
  
<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

// Instantiate module fa_byname and connect signals to ports by name
fulladd4 fa_byname (.sum(SUM), .b(B), .c_in(C_IN), .a(A));
4.3 Hierarchical Names

4.1 Modules
4.2 Ports
4.3 Hierarchical Names
Hierarchical Names

- Verilog supports a hierarchical design methodology.
- A particular identifier has a unique place in the design hierarchy.
- Hierarchical name referencing allows us to denote every identifier in the design hierarchy with a unique name.
- A hierarchical name is a list of identifiers separated by dots (".") for each level of hierarchy.
- Any identifier can be addressed from any place in the design by simply specifying the complete hierarchical name of that identifier.
- The top-level module is called the root module because it is not instantiated anywhere.
4.3 Hierarchical Names

Design Hierarchy for SR Latch Simulation

Hierarchical Names for SR_Latch

- stimulus
  - q, qbar, set, reset, (variables)
  - stimulus.m1
    - stimulus.m1.Q
    - stimulus.m1.Qbar
    - stimulus.m1.S
    - stimulus.m1.R
    - stimulus.n1
    - stimulus.n2
  - stimulus.q
  - stimulus.qbar
  - stimulus.set
  - stimulus.reset
  - stimulus.m1
  - stimulus.m1.Qbar
  - stimulus.m1.R
  - stimulus.n1
“stimulus” is the top-level module (root module).

The identifiers defined in this module are q, qbar, set, and reset.

The root module instantiates m1, which is a module of type SR_latch.

The module m1 instantiates nand gates n1 and n2. Q, Qbar, S, and R are port signals in instance m1.

To display the level of hierarchy, use the special character %m in the $display task.