Digital Design with Verilog

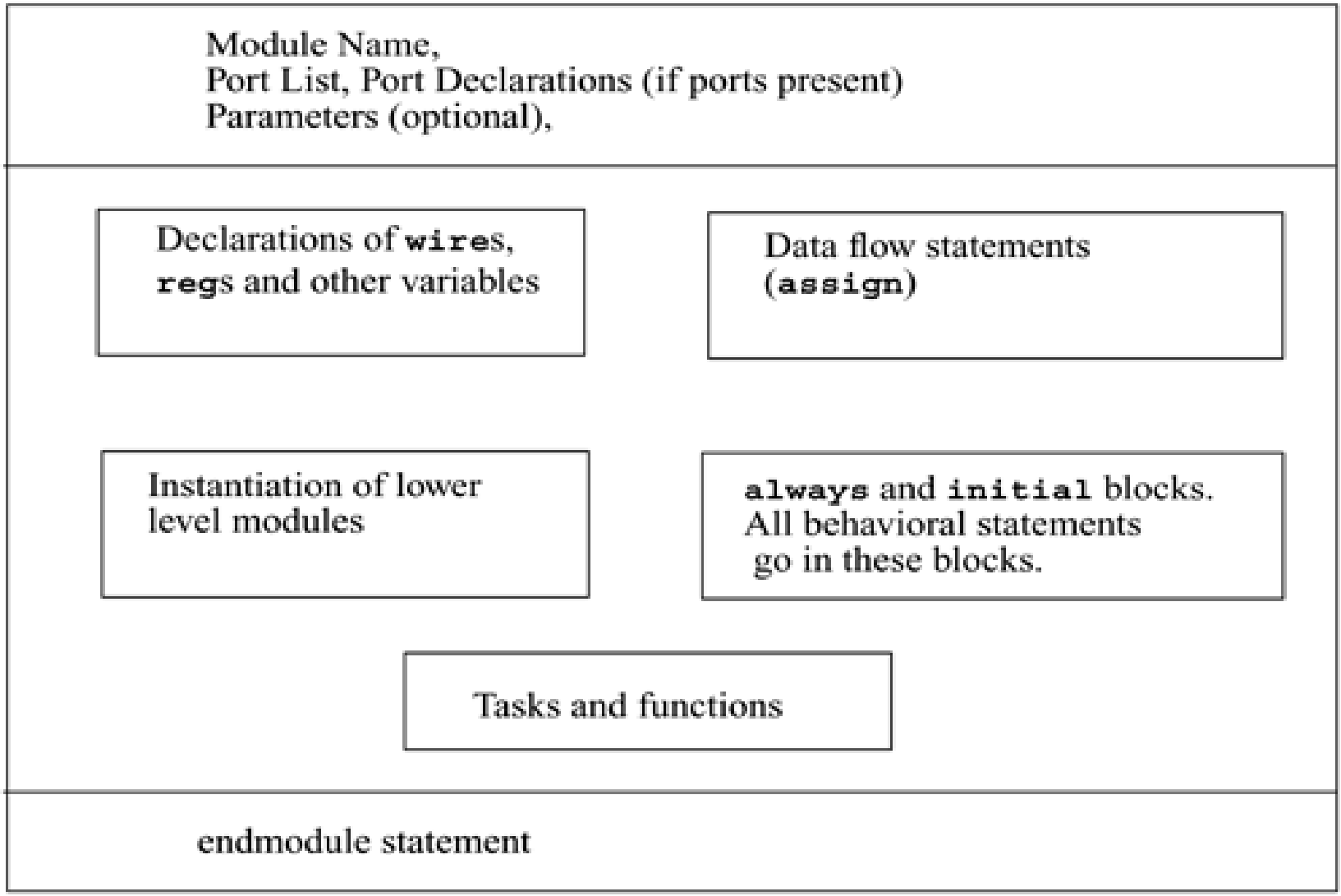
Verilog

Lecture 6: Modules and Ports

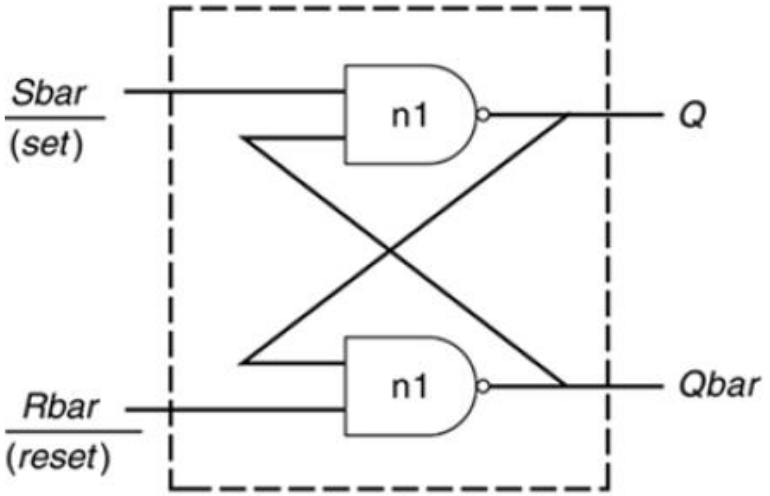
Learning Objectives

- Identify the components of a Verilog module definition
- Understand how to define the port list for a module and declare it in Verilog.
- Describe the port connection rules in a module instantiation.
- Understand how to connect ports to external signals, by ordered list, and by name.

Components of a Verilog Module



Modules (Contd.)




```
// 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
```

Modules (Contd.)

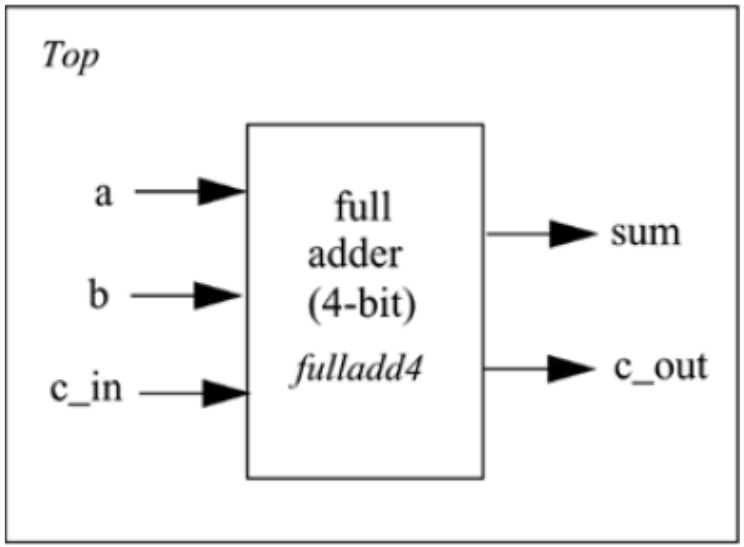
```
// Module name and port list
// Stimulus module
module Top;
// Declarations of wire, reg, and other variables
wire q, qbar;
reg set, reset;
// Instantiate lower-level modules
// In this case, instantiate SR_latch
// Feed inverted set and reset signals to the SR latch
SR_latch m1(q, qbar, ~set, ~reset);
// Behavioral block, initial
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

- 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 are not visible to the environment.
- Ports are also referred to as “terminals”

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.




```
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.
- Port declarations –input, output, inout
  - Port declarations are implicitly declared as **wire**

```
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
```

Port Declaration (Contd.)

- Port declarations for DFF

```
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
```

- Ports of the type input and inout cannot be declared as reg because reg variables store values and input ports should not store values but simply reflect the changes in the external signals they are connected to.

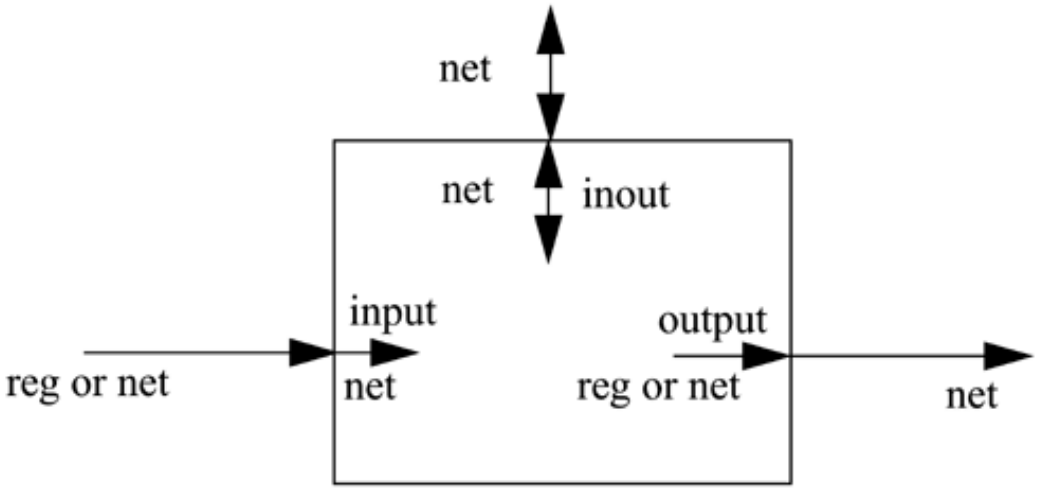
Port Declaration (Contd.)

- This syntax avoids the duplication of naming the ports in both the module definition statement and the module port list definitions.

```
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
```

Port Connection Rules

- One can visualize a port as consisting of two units, one unit that is internal to the module and another that is external to the module.
  - The internal and external units are connected.
  - There are rules governing port connections when modules are instantiated within other modules.



Port Connection Rules (Contd.)

- Unconnected ports
  - Verilog allows ports to remain unconnected.

```
fulladd4 fa0(SUM, , A, B, C_IN);

 // Output port c_out is unconnected
```

Port Connection Rules (Contd.)

- Example of illegal port connection

```
module Top;
//Declare connection variables
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 connected to a register variable SUM in module Top.
.
.
<stimulus>
.
.
endmodule
```

Connecting Ports to External Signals

- Connecting 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.
- Connecting ports by name
  - For large designs where modules have, say, 50 ports, 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, rather than by position.

Connection 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 port by name

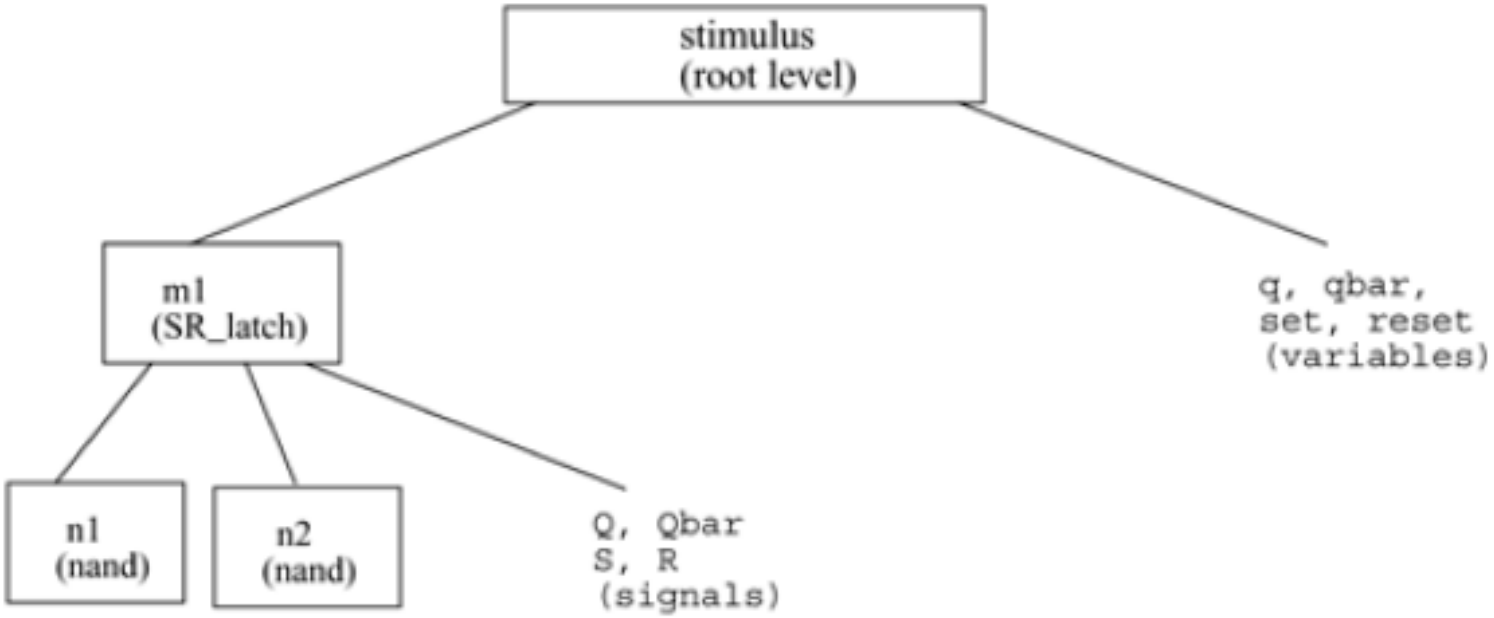
```
// Instantiate module fa_byname and connect signals to ports by name
fulladd4 fa_byname(.c_out(C_OUT), .sum(SUM), .b(B), .c_in(C_IN),
.a(A),);

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

- Another advantage of connecting ports by name is that as long as 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.

Hierarchical Names

- 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.



Hierarchical Names (Contd.)

- Example

stimulus	stimulus.q
stimulus.qbar	stimulus.set
stimulus.reset	stimulus.m1
stimulus.m1.Q	stimulus.m1.Qbar
stimulus.m1.S	stimulus.m1.R

- Each identifier in the design is uniquely specified by its hierarchical path name. To display the level of hierarchy, use the special character %m in the $display task.

References

- Chapter 4, Verilog HDL by Samir Palnitkar, Second Edition.

Thank you !!!