Indian Institute of Information Technology Vadodara

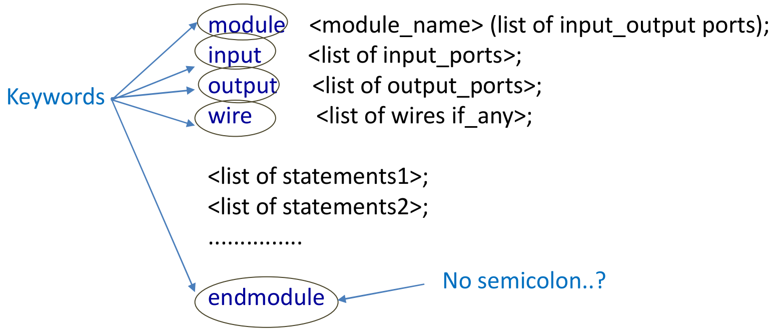
CS/IT 429

Test Bench

Dr. Yash Agrawal
Visiting Faculty, IIIT Vadodara
Associate Professor, DA-IICT Gandhinagar

Functionality Module

Anatomy of a Module in Verilog HDL



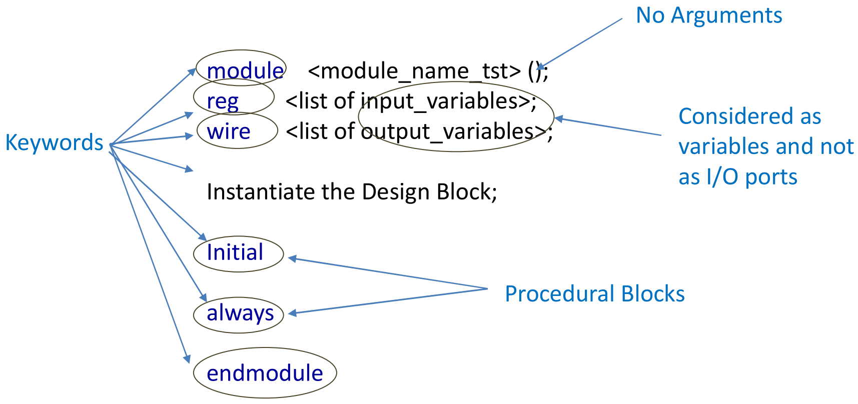
```
module    <module_name> (list of input_output ports);
input     <list of input_ports>;
output     <list of output_ports>;
wire        <list of wires if_any>;

<list of statements1>;
<list of statements2>;
...............

endmodule
```

Keywords

No semicolon..?

Test Bench

Anatomy of Test Bench



```
module   <module_name_tst> ();
reg    <list of input_variables>;
wire    <list of output_variables>;

Instantiate the Design Block;

Initial

always

endmodule
```

No Arguments

Keywords

Considered as variables and not as I/O ports

Procedural Blocks

Contd…

Test Bench Application

- To check functionality of module (provide test values to the system).
- To generate signals.

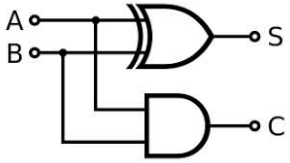
Contd…

Test Bench Application

- To check functionality of module (provide test values to the system).
- To generate signals.

Contd…

Test Bench: Half Adder and Test Bench



Test Bench1

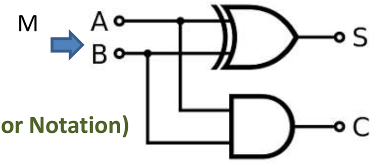
```
module half_adder (s, c, a, b);
input a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
t=0  s1 = 1'b0;    c1 =  1'b0;
t=2  s1 = 1'b1;    c1 =  1'b0;
t=4  s1 = 1'b1;    c1 =  1'b0;
t=6  s1 = 1'b0 ;   c1 =  1'b1 ;
```

```
module half_adder_tst1 ();
reg a1, b1;
wire s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0  a1 = 1'b0;    b1 = 1'b0;
#2  a1 = 1'b0;    b1 = 1'b1;
#2  a1 = 1'b1;    b1 = 1'b0;
#2  a1 = 1'b1;    b1 = 1'b1;
end
endmodule
```

Contd…

Test Bench: Half Adder and Test Bench



Test Bench2 (using Vector Notation)

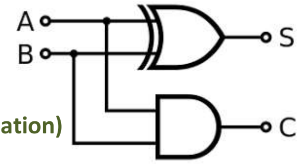
```
module half_adder (s, c, a, b);
input a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
t=0  s1 = 1'b0;    c1 =  1'b0;
t=2  s1 = 1'b1;    c1 =  1'b0;
t=4  s1 = 1'b1;    c1 =  1'b0;
t=6  s1 = 1'b0 ;   c1 =  1'b1 ;
```

```
module half_adder_tst2 ();
reg [1:0] M;
wire s1, c1;
half_adder  h1(s1, c1, M[0], M[1]);
initial
begin
#0  M[0] = 1'b0;    M[1] = 1'b0;
#2  M[0] = 1'b0;    M[1] = 1'b1;
#2  M[0] = 1'b1;    M[1] = 1'b0;
#2  M[0] = 1'b1;    M[1] = 1'b1;
end
endmodule
```

Contd…

Test Bench: Half Adder and Test Bench



Test Bench3 (using Hierarchy Notation)

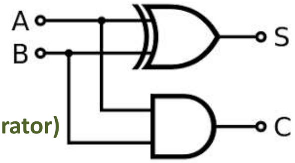
```
module half_adder (s, c, a, b);
input a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
t=0  s1 = 1'b0;    c1 =  1'b0;
t=2  s1 = 1'b1;    c1 =  1'b0;
t=4  s1 = 1'b1;    c1 =  1'b0;
t=6  s1 = 1'b0 ;   c1 =  1'b1 ;
```

```
module half_adder_tst3 ();
reg a1, b1;
wire s1, c1;
half_adder  h1(.s(s1), .c(c1), .a(a1, .b(b1));
initial
begin
#0  a1 = 1'b0;    b1 = 1'b0;
#2  a1 = 1'b0;    b1 = 1'b1;
#2  a1 = 1'b1;    b1 = 1'b0;
#2  a1 = 1'b1;    b1 = 1'b1;
end
endmodule
```

Contd…

Test Bench: Half Adder and Test Bench



Test Bench4 (using Concatenation Operator)

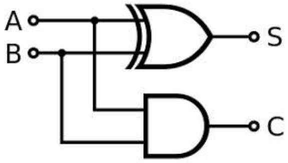
```
module half_adder (s, c, a, b);
input a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
t=0  s1 = 1’b0;    c1 =  1’b0;
t=2  s1 = 1’b1;    c1 =  1’b0;
t=4  s1 = 1’b1;    c1 =  1’b0;
t=6  s1 = 1’b0 ;   c1 =  1’b1 ;
```

```
module half_adder_tst4 ();
reg a1, b1;
wire s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0  {a1, b1} = {1’b0, 1’b0};
#2  {a1, b1} = {1’b0, 1’b1};
#2  {a1, b1} = {1’b1, 1’b0};
#2  {a1, b1} = {1’b1, 1’b1};
end
endmodule
```

Contd…

1. Test Bench: 2-bit 2-input Half Adder



Test Bench

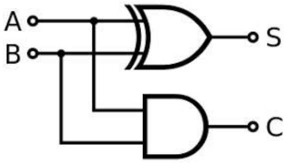
```
module half_adder (s, c, a, b);
input [1:0] a, b;
output [1:0] s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
t=0  s1 = 2’b00;    c1 =  2’b00;
t=2  s1 = 2’b10;    c1 =  2’b00;
t=4  s1 = 2’b10;    c1 =  2’b00;
t=6  s1 = 2’b00 ;   c1 =  2’b10;
```

```
module half_adder_tst1 ();
reg [1:0] a1, b1;
wire [1:0] s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0  a1 = 2’b00;    b1 = 2’b00;
#2  a1 = 2’b00;    b1 = 2’b10;
#2  a1 = 2’b10;    b1 = 2’b00;
#2  a1 = 2’b10;    b1 = 2’b10;
end
endmodule
```

Contd…

2. Test Bench: 2-bit Module; 1-bit Test Bench



Test Bench

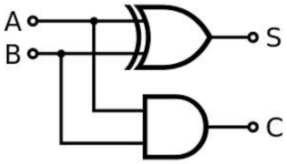
```
module half_adder (s, c, a, b);
input [1:0] a, b;
output [1:0] s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
t=0  s1 = 1’b0;    c1 =  1’b0;
t=2  s1 = 1’b1;    c1 =  1’b0;
t=4  s1 = 1’b1;    c1 =  1’b0;
t=6  s1 = 1’b0;    c1 =  1’b1;
```

```
module half_adder_tst1 ();
reg a1, b1;
wire s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0  a1 = 1’b0;    b1 = 1’b0;
#2  a1 = 1’b0;    b1 = 1’b1;
#2  a1 = 1’b1;    b1 = 1’b0;
#2  a1 = 1’b1;    b1 = 1’b1;
end
endmodule
```

Contd…

3. Test Bench: 1-bit Module; 2-bit Test Bench



Test Bench

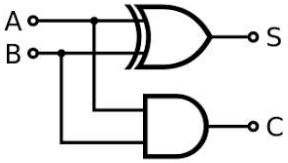
```
module half_adder (s, c, a, b);
input  a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
t=0  s1 = 2’bz0;    c1 =  2’bz0;
t=2  s1 = 2’bz0;    c1 =  2’bz0;
t=4  s1 = 2’bz0;    c1 =  2’bz0;
t=6  s1 = 2’bz0;    c1 =  2’bz0;
```

```
module half_adder_tst1 ();
reg [1:0] a1, b1;
wire [1:0] s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0  a1 = 2’b00;    b1 = 2’b00;
#2  a1 = 2’b00;    b1 = 2’b10;
#2  a1 = 2’b10;    b1 = 2’b00;
#2  a1 = 2’b10;    b1 = 2’b10;
end
endmodule
```

Contd…

4. Test Bench: 1-bit Module; 2-bit Test Bench



Test Bench

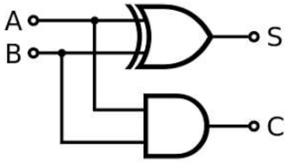
```
module half_adder (s, c, a, b);
input  a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
module half_adder_tst1 ();
reg [1:0] a1, b1;
wire [1:0] s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0  a1 = 2’b00;    b1 = 2’b00;
#2  a1 = 2’b00;    b1 = 2’b01;
#2  a1 = 2’b01;    b1 = 2’b00;
#2  a1 = 2’b01;    b1 = 2’b01;
end
endmodule
```

```
t=0  s1 = 2’bz0;    c1 =  2’bz0;
t=2  s1 = 2’bz1;    c1 =  2’bz0;
t=4  s1 = 2’bz1;    c1 =  2’bz0;
t=6  s1 = 2’bz0;    c1 =  2’bz1;
```

Contd…

5. Test Bench: 1-bit Module; 2-bit Test Bench



Test Bench

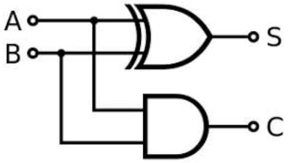
```
module half_adder (s, c, a, b);
input  [1:0] a, b;
output [1:0] s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
module half_adder_tst1 ();
reg [1:0] a1, b1;
wire [1:0] s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0  a1 = 1’b0;    b1 = 1’b0;
#2  a1 = 1’b0;    b1 = 1’b1;
#2  a1 = 1’b1;    b1 = 1’b0;
#2  a1 = 1’b1;    b1 = 1’b1;
end
endmodule
```

```
t=0  s1 = 2’b00;    c1 =  2’b00;
t=2  s1 = 2’b01;    c1 =  2’b00;
t=4  s1 = 2’b01;    c1 =  2’b00;
t=6  s1 = 2’b00;    c1 =  2’b01;
```

Contd…

Concept of `timescale



Test Bench

```
module half_adder (s, c, a, b);
input  a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

```
module half_adder_tst2 ();
reg a1, b1;
wire s1, c1;
half_adder  h1(s1, c1, a1, b1);
initial
begin
#0          a1 = 1’b0;    b1 = 1’b0;
#0.003      a1 = 1’b0;    b1 = 1’b1;
#0.0044     a1 = 1’b1;    b1 = 1’b0;
#0.0037     a1 = 1’b1;    b1 = 1’b1;
end
endmodule
```

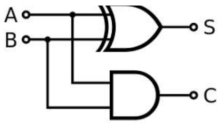
```
t=0       s1 = 1’b0;    c1 =  2’b0;
t=3ps     s1 = 1’b1;    c1 =  2’b0;
t=7ps     s1 = 1’b1;    c1 =  2’b0;
t=11ps    s1 = 1’b0;    c1 =  2’b1;
```

Contd…

Test Bench Application

- To check functionality of module (provide test values to the system).
- To generate signals.

Contd…



Test Bench using ‘always’ Block

```
module half_adder (s, c, a, b);
input  a, b;
output s, c;
assign   s = a ^ b;
assign   c = a & b;
endmodule
```

Test Bench1

```
module half_adder_tst3 ();
reg a1, b1;
wire s1, c1;
half_adder  h1(s1, c1, a1, b1);

initial
begin
#0  a1 = 1’b1;    b1 = 1’b1;
end

always
#2 b1 = ~b1;

always
#4 a1 = ~a1;

endmodule
```

Test Bench2

```
module half_adder_tst4 ();
reg a2, b2;
wire s2, c2;
half_adder  h2(s2, c2, a2, b2);

initial
begin
#0  a2 = 1’b1;    b2 = 1’b1;
end

always
begin
#2 b2 = ~b2;
#4 a2 = ~a2;
end

endmodule
```

Contd…

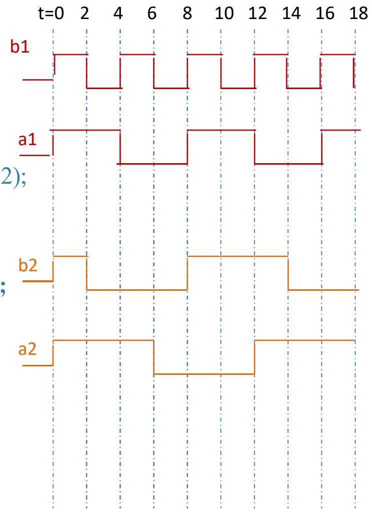
Test Bench using ‘always’ Block

Test Bench1

```
module half_adder_tst3 ();
reg a1, b1;
wire s1, c1;
half_adder  h1(s1, c1, a1, b1);

initial
begin
#0  a1 = 1’b1;    b1 = 1’b1;
end

always
#2 b1 = ~b1;

always
#4 a1 = ~a1;

endmodule
```

Test Bench2

```
module half_adder_tst4 ();
reg a2, b2;
wire s2, c2;
half_adder  h2(s2, c2, a2, b2);

initial
begin
#0  a2 = 1’b1;    b2 = 1’b1;
end

always
begin
#2 b2 = ~b2;
#4 a2 = ~a2;
end

endmodule
```


t=0
2
4
6
8
10
12
14
16
18
b1
a1
b2
a2