Click to add title

- Interface
- Internal

Modeling the Internal of a Module

- 1.Structural Style
  - -Gate Level
  - -Switch Level

2.Dataflow Style

-Module is specified as a set of continuous assignments statements

3.Behavioral or algorithmic style

4.Mixed style

-modelling Large Designs

Port Declaration

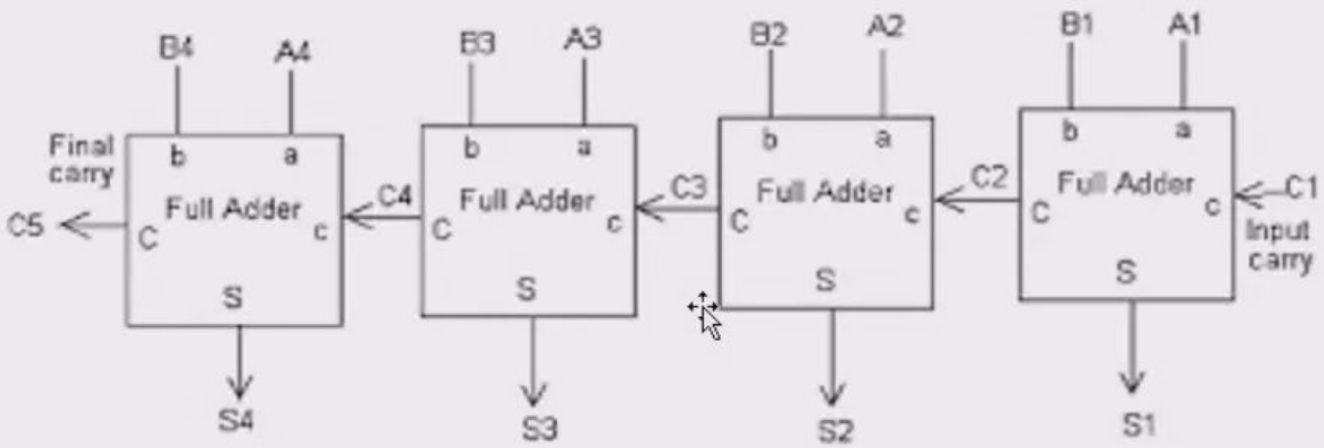
- Interface signals of any Verilog HDL module can be of three types
  - Input
  - output
  - Inout
- The complete interface of a module is to divide it into three parts
  - Port list
  - Port Declaration
  - Data type declaration of each type

Port List style

```
module adder(x,y,c_in,sum,c_out)
  Input [3:0] x,y;
  Input c_in;
  output [3:0] sum;
  output c_out;
  reg [3:0] sum;
  reg c_out;
```

This style of port declaration is known as port list style

Click to add title



Click to add title

```
module adder(x,y,c_in,sum,c_out)
   Input [3:0] x,y;
   Input c_in;
   output  reg [3:0] sum ;
   output  reg c_out;

module adder(input [3:0]x,y, input c_in, output  reg [3:0] sum
   output  reg c_out );
```

You

Port Connection Rules

- Named association
- Positional association

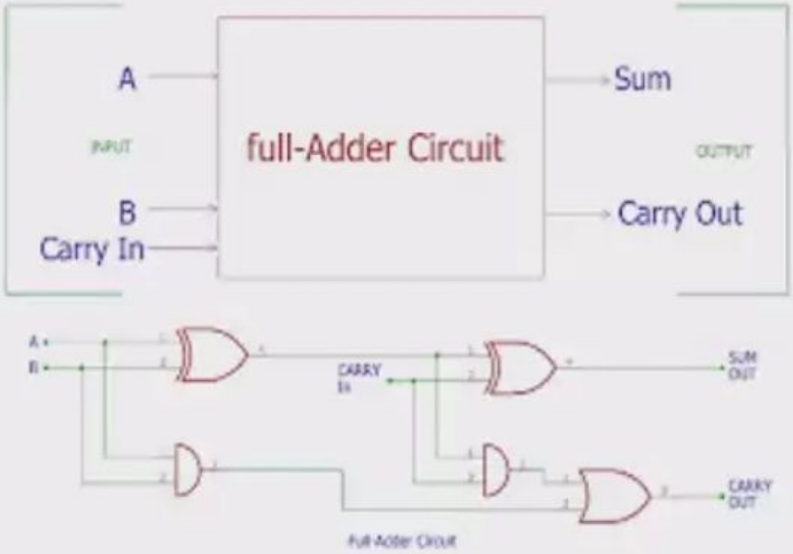
Coding Styles

- A module cannot be declared within another module
- A module can instantiate other modules
- A module instantiation must have a module identifier (instance name) except built in primitives,gate and switch primitives and user defined primitives
- Named association at the top level modules to avoid confusions

Port connection Rules

```
module half adder(x,y,s,c);
Input x,y;
Output ,c;
//half adder body//
//instantiate primitive gates
xor xor1(s,x,y);          ( s,x,y)-Positional association
and and1 (c,x,y);      and1 (instance name is optional)
endmodule
```

Click to add title



Port connection Rules

```
module half adder(x,y,s,c);
Input x,y;
Output ,c;
//half adder body//
//instantiate primitive gates
xor xor1(s,x,y);          ( s,x,y)-Positional association
and and1 (c,x,y);      and1 (instance name is optional)
endmodule
```

Click to add title

```
module full _adder(x,y,cin,s,cout);
Input x,y,cin;
output s,cout;
wire s1,c1,c2;
// full adder body//
// instantiate the half adder
half adder ha_1(x,y,s1,c1);          // (x,y,s1,c1)-Positional  association
half adder ha_2(.x(cin), .y(s1), .s(s), .c(c2)); //ha_2(instance name is necessary)
Or (cout,c1,c2);                     //(.x(cin), .y(s1), .s(s), .c(c2))-Named association
endmodule
```

Click to add title

```
module full _adder(x,y,cin,s,cout);
Input x,y,cin;
output s,cout;
wire s1,c1,c2;
// full adder body//
// instantiate the half adder
half adder ha_1(x,y,s1,c1);          // (x,y,s1,c1)-Positional  association
half adder ha_2(.x(cin), .y(s1), .s(s), .c(c2)); //ha_2(instance name is necessary)
Or (cout,c1,c2);                     //(.x(cin), .y(s1), .s(s), .c(c2))-Named association
endmodule
```

Structural Modelling

- Structural Modelling at gate level

The Half adder instantiates two gate primitives

The Full adder instantiates two half modules and one gate primitive

The Four bit adder is constructed by four full adders instances

Gate level description of half adder

```
module half adder(x,y,s,c);
Input x,y;
Output s,c;
//half adder body
// instantiate primitive gates
 xor(s,x,y);
and(c,x,y);
endmodule
```

You

Gate level description of Full adder

```
module full adder(x,y,cin,s,cout);
input x,y ,cin;
output s,cout;
Wire s1,c1,c2;
//full adder body
//instantiate the half adder
half _adder ha_1(x,y,s1,c1);
half_adder ha_2(cin,s1,s,c2);
Or(cout,c1,c2)
endmodule
```

Gate level description of 4 bit adder

```
module 4_bit_adder(x,y,c_in,sum,c_out);
Input [3:0] x,y;
Input c_in;
output [3:0] sum;
output c_out;
Wire c1,c2,c3;
//4_bit adder body
// instantiate the full adder
```

Click to add title

```
full_adder fa_1(x[0],y[0],c_in,sum[0],c1);
full_adder fa_2 (x[1],y[1],c1,sum[1],c2);
full _adder fa_3(x[2],y[2],c2,sum[2],c3);
full _adder fa_4(x[3],y[3],c3,sum[3],c_out);
endmodule
```

Dataflow Modelling

- Continuous assignment
- Keyword used"assign"

syntax

assign[delay] 1_value=expression;

✓Expression is evaluated and result is assigned to 1_value after the specified delay

✓Default –zero delay

You

Click to add title

```
module full_adder_dataflow(x,y,c_in,sum,c_out);
//  I/O port declarations
Input x,y,c_in;
output sum,c_out;
//specify the function of full adder

assign #5 {c_out,sum} =x+y+c_in;
endmodule
```

Behavioral Modelling

- Two procedural constructs
  - Initial
  - Always

Initial statement used to set up initial values of variable data types

Always are used to model combinational and sequential logic

Mixed Style Modelling

```
module full_adder_mixed_style(x,y,c_in,s,c_out);
Input x,y,c_in;
output s,c_out;
reg c_out;
wire s1,c1,c2;
//structural modeling of HA 1
xor xor_ha1(s1,x,y);
and and_ha1(c1,x,y);
```

You

Click to add title

```
//dataflow modelling of HA2

assign s=c_in^s1;
assign c2=c_in &s1;
//behavioral modelling of output or gate

always @(c1,c2)
c_out=c1|c2;
endmodule
```

You