Two Hardware Description Languages

- **Verilog**
  - developed in 1984 by Gateway Design Automation
  - became an IEEE standard (1364) in 1995
  - More popular in US
- **VHDL (VHSIC Hardware Description Language)**
  - Developed in 1981 by the Department of Defense
  - Became an IEEE standard (1076) in 1987
  - More popular in Europe
- **In this course we will use Verilog**

Defining a module

- **A module is the main building block in Verilog**
- **We first need to declare:**
  - Name of the module
  - Types of its connections (input, output)
  - Names of its connections

Defining a module

```
module example (a, b, c, y);
        input a;
        input b;
        input c;
        output y;

// here comes the circuit description

endmodule
```

A question of style

The following two codes are identical

```
module test ( a, b, y );
        input a;
        input b;
        output y;

endmodule
```

```
module test ( input a,
              input b,
              output y );

endmodule
```

What if we have busses?

- **You can also define multi-bit busses.**
  - [range_start : range_end]
- **Example:**

```
input  [31:0] a;   // a[31], a[30] .. a[0]
output [15:8] b1;  // b1[15], b1[14] .. b1[8]
output [7:0]  b2;  // b2[7], b2[6] .. b1[0]
input         clk; // single signal
```

Basic Syntax

- **Verilog is case sensitive:**
  - `SomeName` and `somename` are not the same!
- **Names cannot start with numbers:**
  - `2good` is not a valid name
- **Whitespace is ignored**

```
// Single line comments start with a //

/* Multiline comments
   are defined like this */
```

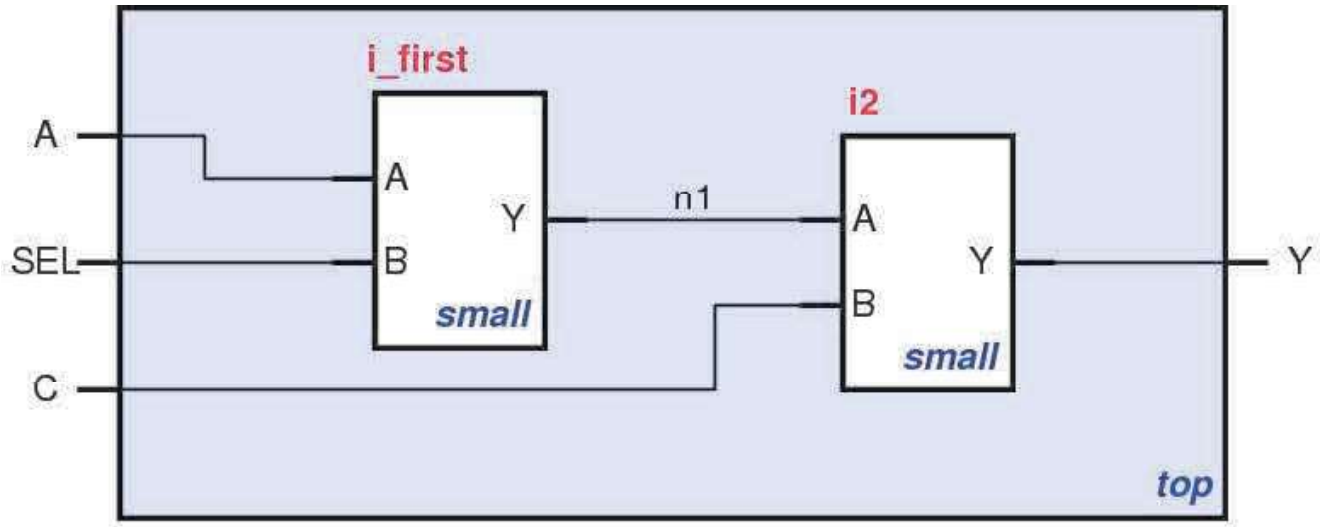
Good Practices

- **Develop/use a consistent naming style**
- **Use MSB to LSB ordering for busses (little-endian)**
  - Try using "`a[31:0]`" and not "`a[0:31]`"
- **Define one module per file**
  - Makes managing your design hierarchy easier
- **Use a file name that equals module name**
  - i.e. module TryThis is defined in a file called TryThis.v

There are Two Main Styles of HDL

- **Structural**
  - Describe how modules are interconnected
  - Each module contains other modules (instances)
  - … and interconnections between these modules
  - Describes a hierarchy
- **Behavioral**
  - The module body contains functional description of the circuit
  - Contains logical and mathematical operators
- **Practical circuits would use a combination of both**

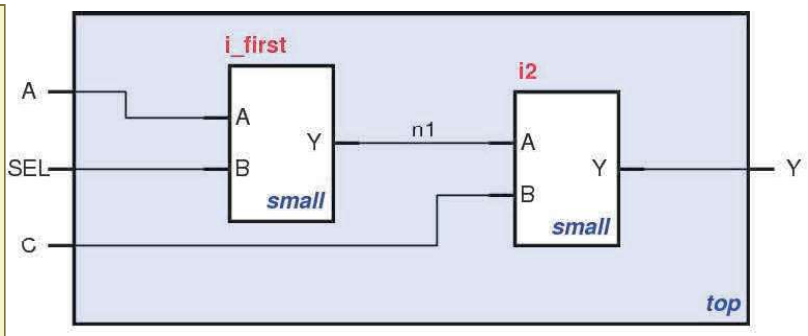
Structural HDL: Instantiating a Module



Structural HDL Example

Module Definitions

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;









endmodule
```

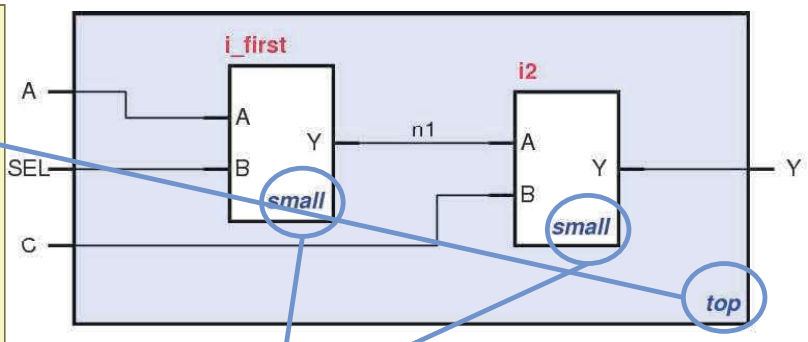



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example

Module Definitions

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;









endmodule
```

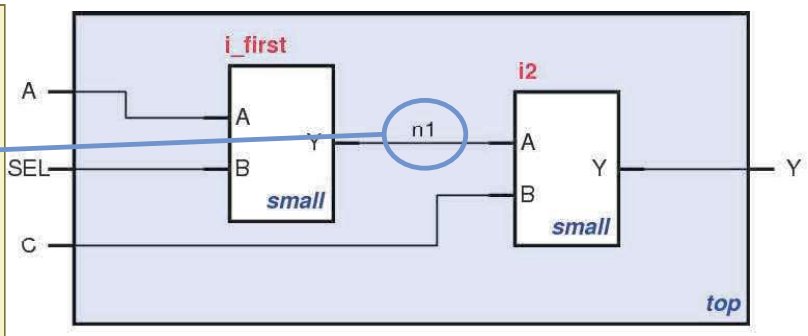



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example

Wire definitions

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;




endmodule
```

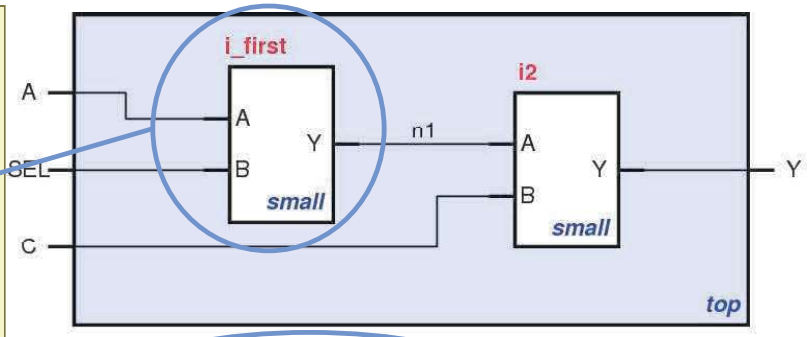



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example

Instantiate first module

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;

// instantiate small once
small i_first ( .A(A),
                .B(SEL),
                .Y(n1)   );




endmodule
```

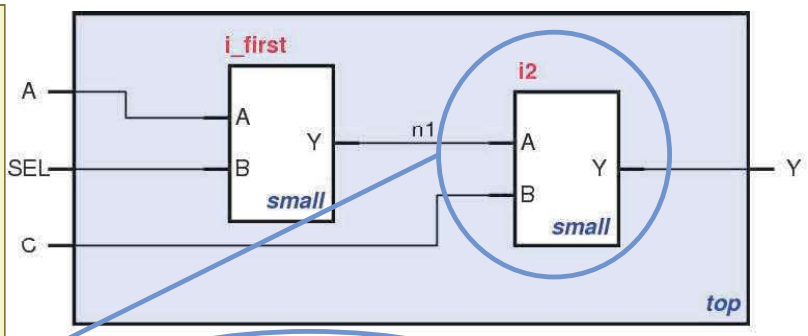



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example

Instantiate second module

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;

// instantiate small once
small i_first ( .A(A),
                .B(SEL),
                .Y(n1)   );

// instantiate small second time
small i2 ( .A(n1),
           .B(C),
           .Y(Y) );

endmodule
```

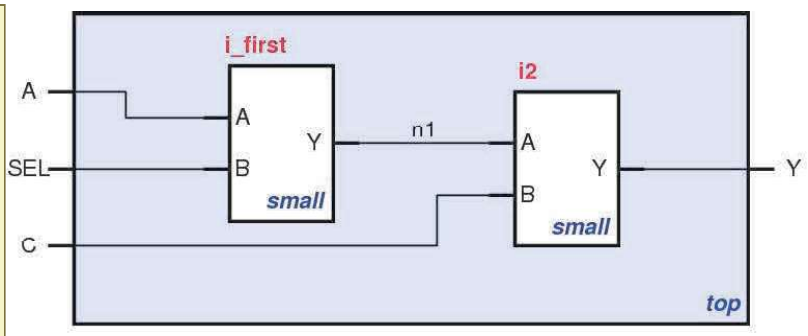



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example

Short Instantiation

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;

// alternative
small i_first ( A, SEL, n1 );

/* Shorter instantiation,
   pin order very important */

// any pin order, safer choice
small i2 ( .B(C),
           .Y(Y),
           .A(n1) );

endmodule
```




```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

What Happens with HDL code?

- **Automatic Synthesis**
  - Modern tools are able to map a behavioral HDL code into gate-level schematics
  - They can perform many optimizations
  - … however they can not guarantee that a solution is optimal
  - Most common way of Digital Design these days
- **Simulation**
  - Allows the behavior of the circuit to be verified without actually manufacturing the circuit
  - Simulators can work on behavioral or gate-level schematics

Behavioral HDL: Defining Functionality

```
module example (a, b, c, y);
       input a;
       input b;
       input c;
       output y;

// here comes the circuit description
assign y = ~a & ~b & ~c |
            a & ~b & ~c |
            a & ~b &  c;

endmodule
```