Section 3

Simulator Demonstration

Lexical Conventions in Verilog

ModelSim Simulator Demonstration

- Start up ModelSim
- Click file -> New -> Project
- Enter the project name and path. Leave the default library name as “work.” Click OK.
- Click on “Add Existing File” in the “Add items to the Project” pop-up window. Browse for the file you want to add to the project. Repeat as necessary. Click close on the pop-up window when you’re finished.
- Click on Compile -> Compile All. Check for errors.
- Click on Simulate -> Simulate. In the “Simulate” pop-up window, click on the “+” next to work. Select the top level module, and click OK.
- Click on View -> Signals. In the “signals” pop-up window, select the signals you want to view. With the signals selected, right-click -> Add to Wave -> Selected Signals.
- In the “wave” window, click on the “run-all” (second icon from the right) icon to run the simulation.
- Use the zoom buttons to zoom in/out on the waveforms.
- Explore the tool to learn other features.

Lexical Conventions in Verilog

White Space and Comments

- White Space
  - White space may be used wherever desired.
    - Exceptions are in strings and escaped identifiers.
    - White space is:
      - Spaces
      - Tabs
      - Carriage returns
- Comments
    - ‘//’ This is a single line comment.
    - ‘/*’ This is for multiple
      line comments. ‘*/’

Operators

- There are three types of Operators:
  - Unary
  - Binary
  - Ternary

```
a = ~b;          // '~' Unary Operator

a = b && c;    // '&&' Binary Operator

a = b ? c : d;   // ' ? : ' Ternary Operator
```

These are examples of the usage of different operators. There are other operators (for example: a = b || c) that will be discussed later in the course.

Number Specifications

- Number Specifications can be sized or unsized, and have a radix of binary, octal, decimal or hexidecimal.
  - Syntax: <size>'<radix> <value>
  - Where:

    <size> (optional) is the number of bits

    <radix> (optional) can be b, B, o, O, d, D, h, H (default is decimal)

    <value> is 0-9, a-f, A-F, x, X, z, Z, _
- Decimal numbers may not use the X and Z logic values. The '_' is ignored and used to improve readability.

The underscore can be used to add readability to long binary integer declarations. For example:

1001100110011001 can be written as 1001_1001_1001_1001. Verilog will ignore the "_", and see it as a 16 bit integer value.

Number Specifications

- Verilog defaults <size> to 32 bits when left unspecified.
- Verilog defaults <radix> to **d** when left unspecified.
- Verilog expands the <value> to fill the specified <size> by working from right to left.
- When <size> is *smaller* than <value>, the left-most bits are truncated.
- When <size> is *larger* than <value>, the left-most bits are filled based on the following rules:
  - When left-most value is:
    - Known, (0 or 1 for binary), then Verilog fills with 0.
    - High impedance (z), then Verilog fills with z.
    - Unknown (x), then Verilog fills with x.

Examples of Number Specifications

```
module integertest;
  reg [31:0] A;   reg [5:0]  B;   reg [63:0] C;   reg [7:0]  D;
  initial begin
    A = 'hA;                  // A = 0000....1010 (32 bits -- hex value A)
    B = 4;                    // B = 000100 (6 bit binary 4, MSB zero filled)
    B = 6'hAA;                // B = 101010 (MSB truncated)
    B = 6'hA;                 // B = 001010 (MSB zero filled)
    C = 64'bz;                // C = zzzz....zzzz (64 bits)
    D = 8'oz5;                // D = zzzzz101 (MSB z filled, LSB octal 5)
    D = 8'bx;                 // D = xxxxxxxx (8 bits, x filled)
    D = 8'b111_1010;          // D = 01111010 (legal syntax - '_' ignored)
  end
endmodule
```

Number Specifications

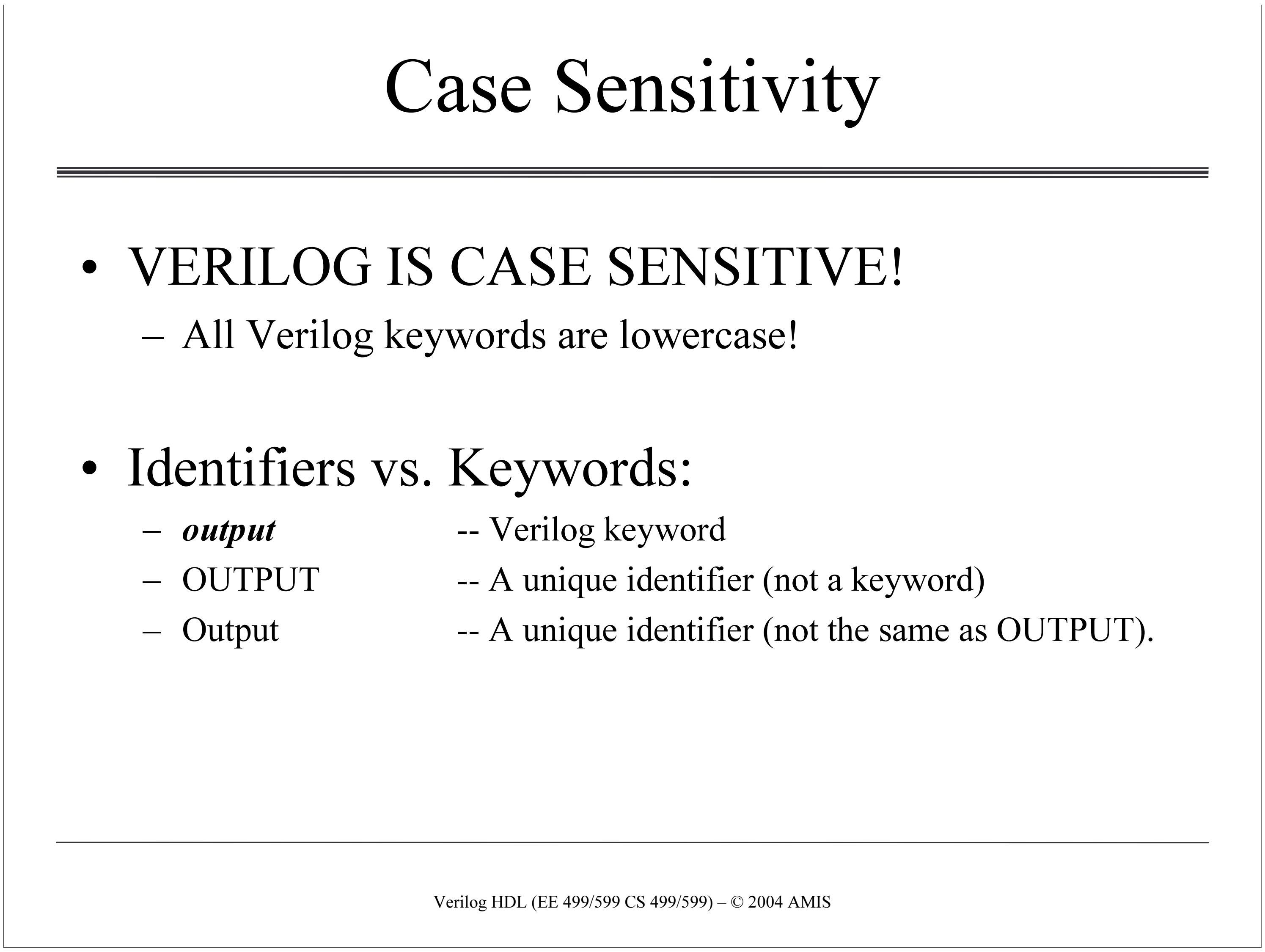
- Negative numbers are specified by putting a minus sign before the size.

```
-6’d3          // Negative Number Stored as two’s complement.

-6’sd3         // Used for signed integer math.

6’d-3          // Illegal  specification
```

Warning!
Negative Numbers and Signed Integer Math (as shown here) are part of the Verilog 2001 Standard and will not work in simulators that do not support Verilog 2001.

Strings

- Strings are enclosed in double quotes and must be specified on one line.
- Verilog recognizes some “C - type” language escape characters such as \n, \t, etc.
- Examples
  - $display (“This is a string with a tab \t and new line \n in it.”);
  - $monitor (“Time = %t input_a = %b input_b = %b”, $realtime, input_a, input_b);



Case Sensitivity

- VERILOG IS CASE SENSITIVE!
  - All Verilog keywords are lowercase!
- Identifiers vs. Keywords:

– ***output***	-- Verilog keyword
– OUTPUT	-- A unique identifier (not a keyword)
– Output	-- A unique identifier (not the same as OUTPUT).

These are all mutually exclusive Verilog identifiers:

OutPut

OutpuT

ouTPut

Identifiers

- Identifiers are names assigned to modules, nets, variables, parameters, specparams, etc.
  - Identifiers must begin with: *a-z, A-Z, _*
  - Identifiers may contain: *a-z, A-Z, 0-9, _ , $*
- Examples:

addbit	sum	Q
_74LS130	Input	my$money

Legal Identifiers?

- Which are legal identifiers??

34net
_bus3
a*b_net
varA_sig
shift_reg_a
{A&B<C}

~#@sel
n@238
module
$cat_food
bus_a[2]
!@#$%^&*[5]

Escaped Identifiers

If you need to use other printable ASCII characters in forming an identifier, you must **“escape”** it.

Escaped identifiers begin with a backslash (\ ) and end with a whitespace.

```
module \74LS74  ( q, \~q , d, clk, \reset* );
```

Notice the whitespace

Although somewhat useful. It is not recommended to use escaped identifiers in RTL code. It makes the code less readable and more cumbersome. Typically, you will only see escaped identifiers in tool-generated netlists.

Review

- Is white space always ignored in Verilog?
- What are the two ways to insert comments into your Verilog code?
- What is the default size of an integer? Default base?
- What are the rules for identifiers?
  - Begins with?
  - Contains?