ECE 351

Professor Greenwood

Day 2

* Verilog Code Example: **Number representation**
  + Syntax: <size> ‘ <base> <value>
  + “size” in bits
  + “base” as ‘b’, ‘B’, ‘h’, ‘H’, ‘d’, or ‘D’
  + Example:
    - 4’b1111
    - 16’h1111
  + For binary strings, there are 4 possible choices:

1. “0”
2. “1”
3. “x”
4. “z”
   * Example:
     + 4’b11x0
   * This is ok:
     + 20’b00xz\_1111\_1....\_0
     + Underscore is ignored, and numbers are concatenated.

* Verilog Code Example: **Strings**
  + “hello world” is supported.
* Verilog Code Example: **Identifiers/Variables**
  + Naming convention:

1. Identifiers can only start with “\_” or a letter.
2. “$” is a reserved special character.
   * Designates a variable as a system task.
   * Don’t use this in naming variables.
3. Identifiers are case sensitive.

* Verilog Code Example:  **Keywords**
  + Variables are names that are reserved.
  + Examples:
    - module
    - begin
    - end
    - parameter
    - include
    - …
  + Keywords are **ALWAYS** lowercase.
* Verilog Code Example: **Data Types**
  + **Net Type Variable:** An interconnection between hardwire elements (doesn’t hold value)
    - Values:
      * “0”
      * “1”
      * “x”
      * “z”
    - Keywords to declare net variables
      * wire //Internal to the module
      * wand
      * wor
      * input //Wire type external to the module (input port)
      * output //Wire type external to the module (output port)
    - Syntax example:
      * wire a,y; // ‘a’ & ‘y’ declared as net type variables.
  + **Register Variable**: A variable that can hold a value.
    - Keyword to declare Register Variables:
      * reg
    - Can be multiple bit values (so can net variables).
    - **NOT** a D-FF!!!
    - “reg” means it’s an identifier that can be **updated** **with assignment statements**.
    - Example:

*reg reset;*

*initial*

*begin*

*reset = 1’b1;*

*#100 reset = 1’b0; //”#100” means wait 100.*

*End*

* + - Example: **Declaring a reg with multiple bits**
      * We want an 8-bit bus.

*Wire [7:0] my\_bus;*

* + - * The “7” is the MSbit and “0” is the LSbit.
      * my\_bus[1] gives you bit one of “my\_bus”.
      * The size of an identifier can be another identifier (i.e. a size variable).
    - Example:

*reg A;*

*reg [7:0]B;*

*reg [2:0]C;*

*B = 8’b11011100;*

*C = B[4:2]; //C = 3’b111*

* + - Sometimes identifiers can be declared as:

*reg [0:7]A;*

* + - * This declares “0” as a the MSbit.
    - Floating point values have some applications in FPGAs, but in most cases they’re not.
    - Signed numbers in a digital design may not be necessary either.
    - Example: **Declaring a reg with multiple bits**

*Reg [7:0]m[4095:0]; //Provides 4096 “m’s” each being 8 bits wide.*

* + - * The prefix [7:0] is the bits per location.
      * The suffix [4095:0] is the number of locations.
  + **Parameter**: Allows you to define constants at compile time.
    - Code cannot change them.
    - Example:

*parameter byte=8;*

*parameter K4=4095;*

*reg [byte-1:0]sally[K4:0];*

* + - Parameters support module re-use.
  + **Compiler Directive**: Instructions to the compiler which do not generate exec. code.
    - Example:
      * (in C) *#include <file.h>*
      * (in Verilog) *`include<file.h>*
      * (in C) *#define A 6; //Simply replaced “A” during compile.*
      * (in Verilog) *`define A 6;* Analogous to: *parameter A=6;*

Then when used further in the code…

*wire[`A:0]w;* Analogous to: *wire[A:0]w;*

* + **Instantiation**: The process of creating an object from a module template.
  + **Instance**: The objects created by instantiation.
    - Example:
      * (in C)  *float foo(int m)*

*{ //This is the template*

*…*

*}*

*main(void){*

*…*

*y = foo(2); //Where the template is exec.*

*…*

*}*

* + - * (in Verilog) *module foo( ..args..) //module template*

*…*

*endmodule*

*module main(..args..) //Instantiation.*

*foo(…); //This is an instance of template*

*endmodule*

* + **port:** The interface between a module and its environment.
    - Example of **net variable** declarations:
      * Input // Input port
      * Output // Output port
      * Inout //Bidirectional port
* Modules declared without any port list are the “Top” level modules and are the testbench with defined test vectors for any modules instantiated.
* **Execution in Verilog is “Massively Parallel”.** 
  + Variables in C are required to be defined prior to program execution (at the head of the program).
  + (Variables in Verilog may be declared anywhere, including at the end of the text (no constraints).
* **ALWAYS REMEMBER**:
  + The purpose of writing a program is **not** to get something you can simulate, but rather **something you can SYTHNESIZE**.