BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI (RAJ)

MEL G626 VLSI Architecture (Sem -2: AY 2024-25)

Lab-1

Objective:

(1) To learn the declarations of Verilog register and wire objects of different bit-widths and unsigned and signed types
(2) To learn of array declaration of objects and addressing of individual objects in the array; use of unsigned limited bit-width indices vs use of integer indices
(3) To learn about bit selection, part selection, concatenation and repetition
(4) To learn about numerical value specification using binary, octal, hex and decimal formats for unsigned and signed values
(5) To learn about the operators in Verilog and understand the operations they perform
(6) To learn about stimulus generation with desired timing for testing a Verilog model
(7) To learn about the Verilog commands for observing of values of objects during simulation
(8) To write a self-contained Verilog module that encompasses an experimental verification of all the learnings as above.

Examples (1) : Declaration of register and wire objects of different bit-widths and unsigned and signed types

reg a, b, c;   // one-bit registers a, b, c

reg [3 : 0] a1, b1;   // 4-bit registers a1, b1 : numerical value = 4-bit unsigned (magnitude)

reg [15 : 0] a2, b2, c2, d2;   // 16-bit registers a2, b2, c2: numerical value = 16-bit unsigned (magnitude)

reg signed [3 : 0] a1s, a2s;   //4-bit registers a1s, a2s: numerical value = 4-bit signed

reg signed [15 : 0] a2s, b2s, c2s;   //16-bit registers a2s, b2s, c2s: numerical value = 16-bit signed (2's complement representation)

wire x, y, z;   //one-bit wires x, y, z

wire [3 : 0] x4, y4, z4;   //4-bit wires, numerical value = 4-bit unsigned (magnitude)

wire signed [15 ; 0] x4s, y4s, z4s;   //16-bit wires, numerical value = 16-bit signed

Bit-select:

a2 [12] is one bit value e.g. bit number 12 of the 16-bit register a2

Part-select:

a2 [11 : 8] is a part e.g. four-bit value comprised of bits 11 down to 8 of 16-bit vector register a2

a1 = a2 [11 ; 8];   //assign bits 11,10 9,8 of sixteen-bit register a2 to bits 3,2,1,0 of
                                      //four bit register a1

b1 = a2 [7 : 4];   //assign bits 7,6,5,4 of sixteen-bit register a2 to bits 3,2,1,0 of

//four-bit register b1

Examples (2) : Declaration of arrays of register and wire objects and specifying an element of the array

reg [15 : 0] rf [0 : 15] ;   //rf is 1-d array with 16 elements; each element is a 16-bit

//unsigned register

reg signed [15 : 0] rfs [0 ; 7];   //rfs is 1-d array with 8 elements; each element is

// a 16-bit signed register

rf [3] is the element of array rf corresponding to index value 3. It is a 16-bit register (unsigned)

Concatenation:

d2 = { a2[7: 4], 4 ‘b0101, a, b1[3 : 1], 4 ‘h0};

/* 16- bits comprised of (from left to right) bits 7,6,5,4 of register a2, then bits 0101, register a, then bits 3,2,1 of register b1, then bits 0000 are assigned to sixteen bit register d2 */

// Note: The target of assignment can also be a concatenation as shown by the example below

{a1, b1[1:0]} = {a2 [7 : 4], x, y};

/*4-bit register a1 is assigned bits 7 down to 4 of sixteen bit register a2 and bit numbers 1 and 0 of 4-bit register b1 are respectively assigned the bit values of wires x and y */

Repitition:

{4{3 ‘b010}} is same as {3 ‘b010, 3’b010, 3’ b010, 3 ‘b010}

a2 = {4{2 ‘b10}, 4{2 ‘b1x}};

// This is same as:

a2 = {2 ‘b10, 2 ‘b10, 2 ’b10, 2 ‘b10, 2 ‘b1x, 2 ‘b1x, 2 ‘b1x, 2 ‘b1x};

//which is the same thing as:

a2 = 16 ‘b101010101x1x1x1x;

Number Specification in Verilog:

Numbers can be assigned to objects (reg or wire objects)

Numbers can also appear directly as operands in expressions

https://projectf.io/posts/numbers-in-verilog/

Numbers can be specified as sized or un-sized using binary, octal, decimal or hexadecimal base. Un-sized numbers use a default size of 32 bits whereas sized numbers use the specified size in number of bits.

Numbers can be unsigned or signed

Sized numbers start with a size specification (always a decimal number that specifies the number of bits to be used to store the number that follows). The number itself starts by specifying the base used for encoding its value. Any one of the four bases (‘b for binary, ‘o for octal, ‘d for decimal and ‘h for hexadecimal base) can be used. Following the base specification the numerical value is supplied using the digit literals applicable for the base specified.

Examples: Sized un-signed numbers:

```
12 ‘b111000110001      //12-bit unsigned binary number 111000110001

12 ‘he31          //12-bit unsigned hexadecimal number (value same as above)

12 ‘o7061         //12-bit unsigned octal number (value same as above)

12 ‘d3633         //12-bit unsigned decimal number (value same as above)
```

Examples: Un-sized unsigned numbers:

```
 ‘b111000110001        //32-bit unsigned binary number 111000110001

 ‘he31   //32-bit unsigned hexadecimal number (value same as above)

 ‘o7061 //32-bit unsigned octal number (value same as above)

 ‘d3633 //32-bit unsigned decimal number (value same as above)

3633     //32-bit unsigned decimal number (value same as above)

         //If no base is specified, decimal base is used as the default base
```

Examples: Sized signed numbers:

A negative sign can be placed before the size specification to indicate a negative number

This is simply the sign-magnitude representation of a negative number. The magnitude can be specified using hex, decimal, octal or binary base.

```
-12 ‘d463         //Signed decimal number (-463) is internally represented as
                  //12-bit 2’s complement number

-12 ‘sd463        //used when performing signed integer math
```

Note: Verilog performs unsigned arithmetic if one or both operands are unsigned. It performs signed arithmetic only if both operands are signed quantities.

Operators in Verilog:

At the end of this lab sheet, you should test the operation of every operator in Verilog by appropriately declaring registers and assigning test input values to them. Thereafter, assign the value of the expression formed by the operation of the operator on its operand registers to a result register and observe the values of the operand registers, name of the operator and value of the result register.

You should try out the same exercise by forming the expression where the operator has one register operand and one directly specified value as the operand

System Tasks:

Verilog provides standard system tasks for certain routine operations. Learn the functions and parameter specifications (from the Verilog reference book) for the following tasks:

$display

$strobe

$monitor

$scanf

$time

$signed

$stop

$finish

Stimulus generation:

Learn to generate multi-channel waveforms with timing specifications using delays (regular delay and intra-assignment delay):

In regular delay control a non-zero delay is specified to the left of a procedural assignment statement (blocking or non-blocking) which delays the execution of that procedural assignment statement

Intra-assignment delay assigns delay to the right of the assignment operator. it does not delay the execution of the assignment statement. Expression to the right of the assignment operator is evaluated at current time, but the evaluated value is written to the object of assignment on the left hand side of the assignment operator only after the intra-assignment delay specified.

Note: If the assignment operator is blocking assignment operator, then the following statement will be executed only after the target of assignment has been written to.

A non-blocking assignment, on the contrary, will proceed with the execution of the following statement without waiting for the writing of the evaluated expression value to the target of **assignment.**

Exercise:

Using your understanding so far, write an initial statement comprising of a sequence of Verilog sequential statements that tests the function of each of the Verilog operators and displays the value of your inputs to the operator and the output generated by the operator.

Each new test is performed after a delay of 10 time units after the previous test.