

Tutorials Home

VERILOG BASIC TUTORIAL

Verilog Introduction

Installing Verilog and Hello World

Simple comparator Example

Code Verification

Simulating with verilog

Verilog Language and Syntax

Verilog Syntax Contd..

Verilog Syntax - Vector Data

Verilog \$monitor

Verilog Gate Level Modeling

Verilog UDP

Verilog Bitwise Operator

Viewing Waveforms

Full Adder Example

Multiplexer Example

Always Block for Combinational ckt

if statement for Combinational ckt

Case statement for Combinational ckt

Hex to 7 Segment Display

casez and casex

full case and parallel case

Verilog for loop

Verilog localparam and parameter

SEQUENTIAL CKT TUTORIAL

Sequential Circuits

Serial Shift Register

Binary Counters

Ring Counter

MISC VERILOG TOPICS

Setting Path Variable

Verilog Tutorial Videos

Verilog Interview questions #1

Verilog Interview questions #2

Verilog Interview questions #3

Verilog Books

Synchronous and Asynchronous Reset

Verilog Language Continued

Vector Data

In the single bit comparator example we had only two sets of 1 bit input. What if we need to design a comparator that has two sets of 2 bit input ? Verilog provides the concept of Vectors. Vectors are used to represent multi-bit busses.

A vector to represent a multi bit bus is declared as follows

```
reg [7:0] eightbitbus; // 8-bit reg vector with MSB=7 LSB=0
```

The reg [7:0] means you start with 0 at the rightmost bit to begin the vector, then move to the left. We could also declare the vector as

```
reg [0:7] eightbitbus; // 8-bit reg vector with MSB=0 LSB=7
```

In which case the LSB will be represented by leftmost bit.

Let us rewrite our comparator example, so that it now use two bit bus in place of one bit.

```

1. `timescale 1ns / 1ps
2. //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
3. // Company: referencedesigner.com
4. //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
5. module comparator2bit(
6.     input [1:0] x,
7.     input [1:0] y,
8.     output z
9. );
10.
11. assign z = (x[0] & y[0] & x[1] & y[1]) |
12.            (~x[0] & ~y[0] & x[1] & y[1]) |
13.            (~x[0] & ~y[0] & ~x[1] & ~y[1]) |
14.            (x[0] & y[0] & ~x[1] & ~y[1]);
15. endmodule
16.
17.
```

```

1. `timescale 1ns / 1ps
2. module stimulus;
3.     // Inputs
4.     reg[1:0] x;
5.     reg[1:0] y;
6.     // Outputs
7.     wire z;
8.     // Instantiate the Unit Under Test (UUT)
9.     comparator2bit uut (
10.         .x(x),
11.         .y(y),
12.         .z(z)
13.     );
14.
15.     initial begin
16.         $dumpfile("test.vcd");
17.         $dumpvars(0,stimulus);
18.         // Initialize Input
19.         x = 0;
20.         y = 0;
21.         #20 x = 1;
22.         #20 y = 1;
23.         #20 y = 3;
24.         #20 x = 3;
25.         #20 y = 1;
26.         #20 y = 0;
27.
28.         #40 ;
29.
30.     end
```

Left and Right shift << and >>

Negative Numbers

Blocking Vs Non Blocking

wand and wor

delay in verilog

\$dumpfile and \$dumpvars

Useful Resources

Verilog Examples

VERILOG QUIZS

Verilog Quiz # 1

Verilog Quiz # 2

Verilog Quiz # 3

Verilog Quiz # 4

Verilog Quiz # 5

Verilog Quiz # 6

Verilog Quiz # 7

Verilog Quiz # 8

Verilog Quiz # 9

OTHER TUTORIALS

Verilog Simulation with Xilinx ISE

VHDL Tutorial

```

31.
32.         initial begin
33.             $monitor("t=%3d x=%2b,y=%2b,z=%d \n",$time,x,y,z, );
34.             end
35.
36.     endmodule
37.

```

This example produces the following result in console

t= 0 x=00,y=00,z=1

t= 20 x=01,y=00,z=0

t= 40 x=01,y=01,z=1

t= 60 x=01,y=11,z=0

t= 80 x=11,y=11,z=1

t=100 x=11,y=01,z=0

t=120 x=11,y=00,z=0

Note that the assignment x =3 means 11 in binary.

The verification is only partial and we would let the reader write code that will verify it exhaustively. You may also write a self verifying code by checking the output z against expected value.

Thanks Rodney Schaerer for mentioning error in the stimulus code.



< Previous

Next >