

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

Binary Counter using verilog

A binary counter is a simple counter that has an initial value of 0 at the time of reset. Its value keeps incrementing by 1 at each clock cycle. And finally it reaches its maximum value, say 1111 in binary for a 4 bit counter, it again reaches 0000 in binary and keeps counting one.

Here is the verilog implementation of shift register.

```

1. // referencedesigner.com
2. // Example of binary_counter
3. module binary_counter
4.     #(parameter N=4)
5.     (
6.         input wire clk, reset,
7.         output wire [N-1:0] binary
8.     );
9.
10.    reg [N-1:0] r_reg;
11.    wire [N-1:0] r_next;
12.
13.    always @(posedge clk, negedge reset)
14.    begin
15.        if (~reset)
16.            r_reg <= 0;
17.        else
18.            r_reg <= r_next;
19.        end
20.        assign binary = r_reg;
21.        assign r_next = r_reg+1;
22.    endmodule

```

Explanation

We make use of the simple addition statement for incrementing value. We should note the property of addition that - it wraps when it reaches its max value.

```
assign r_next = r_reg+1;;
```

The testbench for the binary counter

```

1. `timescale 1ns / 1ps
2. module stimulus;
3.     // Inputs
4.     reg clk ;
5.     reg reset;
6.     // Output
7.     wire[3:0] binary;
8.     // Instantiate the Binary Counter
9.     binary_counter #(4) s1 (
10.        .clk(clk),
11.        .reset(reset),
12.        .binary(binary)
13.    );
14.
15.    integer i;
16.    initial
17.    begin
18.        clk = 0;
19.        for(i =0; i<=40; i=i+1)
20.        begin
21.            #10 clk = ~clk;
22.        end
23.    end
24.
25.

```

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

```

26. initial
27. begin
28.
29. $dumpfile("test.vcd");
30. $dumpvars(0,stimulus);
31.
32. reset =1;
33. #2 reset = 0;
34. #2 reset =1;
35. end
36.
37.
38.             initial begin
39.                 $monitor("clk=%d binary=%4b",clk,binary);
40.             end
41.
42. endmodule

```

Exercise

1. Change the binary counter so that it counts down in place of count up

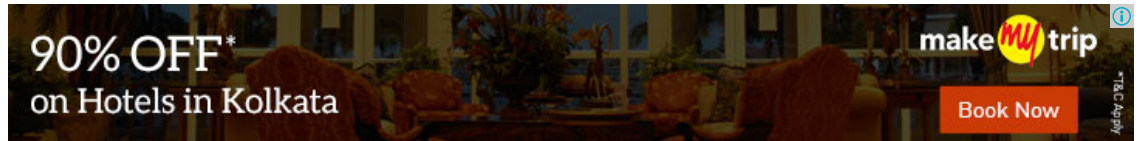
- Answer [here](#)

2. Modify the code so that, it gives output 1 (define another output wire), for one clock period every time it reaches max value.

Answer [here](#)

and test bench [here](#)

2. Modify the code so that, it has another input, call it load. When it is 1, at the next rising edge of clock, the counter value is uploaded with N bit wide input - call it input_load. It then keeps counting from there.



< Previous

Next >