Reference Designer

Verilog Comparator example

In our first verilog code, we will start with the design of a simple comparator to start understanding the Verilog language. This will extend our "Hello World" example to create something useful. Let us take a look at the following table which describes the behaviour of a comparator circuit.

Table: A one bit comparator

Input x	Input y	Output z
0	0	1
0	1	0
1	0	0
1	1	1

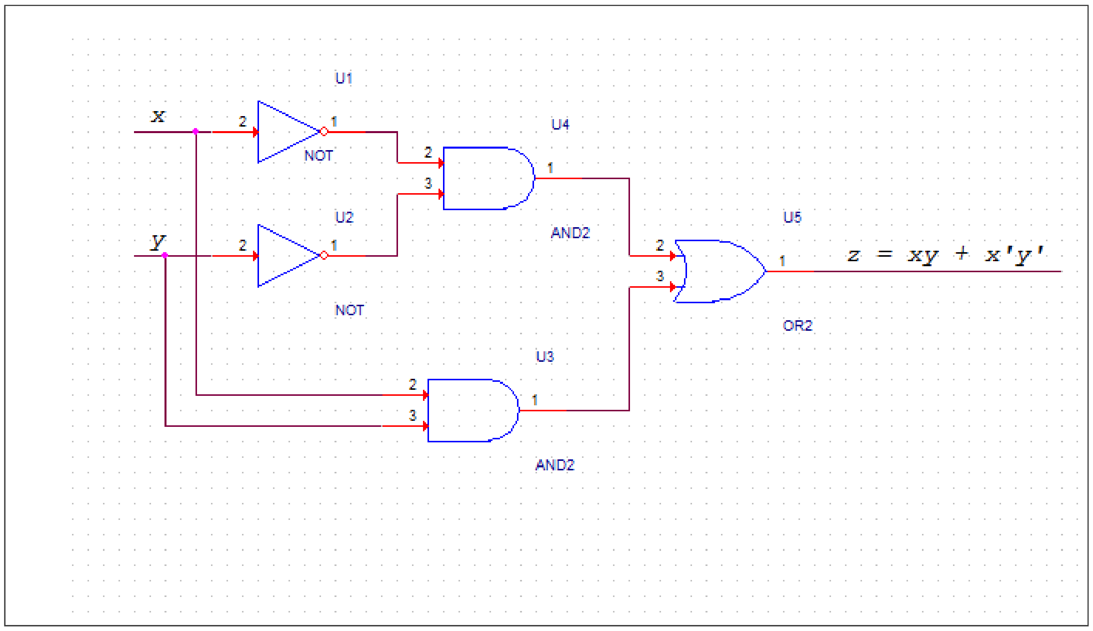
Basically when both the inputs x and y are same, the output z is 1. When the inputs are unequal, the output is 0.

We can describe the circuit using **AND, NOT** and **OR** gates using the following equation.

```
assign z = (~x & ~y) |(x & y);
```

where ~x and ~y represent the complements of x and y respectively.

The following shows a circuit that implements this logic.



And here is the verilog code that implements this logic

```
module comparator(
    input x,
    input y,
    output z
    );

assign z = (~x & ~y) |(x & y);

endmodule
```

We will try to make you understand what Verilog is - in a matter of one day. At least you should be able to compile and run verilog code. We hope that this is something you will be able to achieve within next few pages. Throughout this tutorial we will present you enough examples and exercises so that you have a good grip over the language as well as the verilog concepts.

While Verilog has concurrent blocks executing in parallel, it is still similar to software programming language like C.

If you have closely watched the schematics above and the verilog code below it, you must have appreciated how verilog simplifies the process of hardware design. Before the advent of Verilog, everything was done using schematics. The Schematics were error-prone, difficult to verify and had long process of design, verification, fix, redesign and re verify.

With Verilog the whole dimension and process of hardware circuit design changed. This provided a new kind of abstraction where the details of implemmentation are separated. Verilog design is more like a software programming, but, you must also have a strong understanding of the circuit that works behind the code.

Let us now understand the code. Take a look at

```
module comparator(
    input x,
    input y,
    output z
    );
```

Verilog consists of modules. Inside the modules we have a list of ports (or pins). A port can be an input or an output depending upon its direction. A pin can also be defined as bidirectional using inout.

The direction can also be specified out of the module. This code is equivalent to the previous code.

```
module comparator(
     x,
     y,
     z
    );
        input x;
        input y;
        output z;
```

Let us now take a look at the assign statement

```
assign z = (~x & ~y) |(x & y);
```

This implements a combinational logic. An assign statement is used for modeling only combinational logic. The statement in the assign statement is executed continuously (as against those that trigger on a clock). An assign statement is also called 'continuous assignment statement'.

This statement implements the comparator logic that we had shown earlier in example.

Change Code and Exercize

Now the fun part of the tutorial. The code is reproduced in below. It does not give any output, but it also does not has any syntax error. If you make any error and click on "Edit Verilog Code and Click Here", it will give an error. Now the exercize.

1. Change the code such that it give assigns 1 when the two inputs are unequal.
2. Change the variable names from x, y z to something more meaningful like input1, input2 and output1.
3. Change module name from comparator to something like unequal.

Make sure you do not get any compile errors. We will not see anything in the output window, but that is fine - we will have something more exciting in next page. Save the code in a text file in your computer. We will use it later.

Edit Verilog Code and Click Here

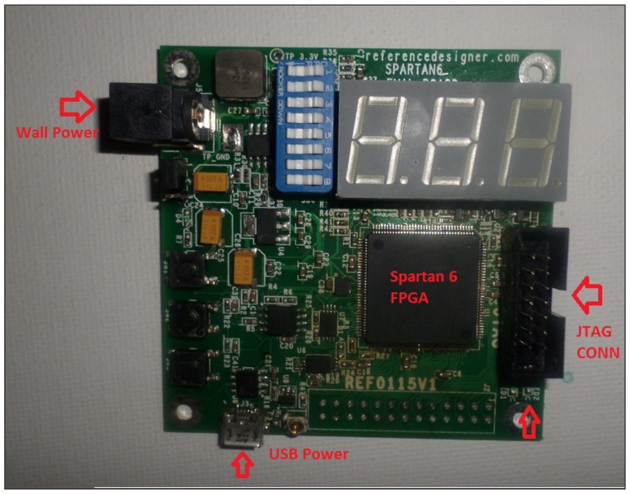
```
module comparator(
    input x,
    input y,
    output z
    );

assign z = (~x & ~y) |(x
& y);

endmodule
```

In the next page we will see how to test this code using the simulation.

NEW The Spartixed evaluation board from Reference Designer has been released for sale and can be purchased here . It is a very low cost $37 board and will allow you to download the synthesized code on this board and you can see the verilog in action. For $55 you can purchase ann Spartan 6 Evaluation board plus a programmer.



We have put together the learning material for using the Spartixed here

< Previous   Next >

Tutorials | Products | Services | Contact Us