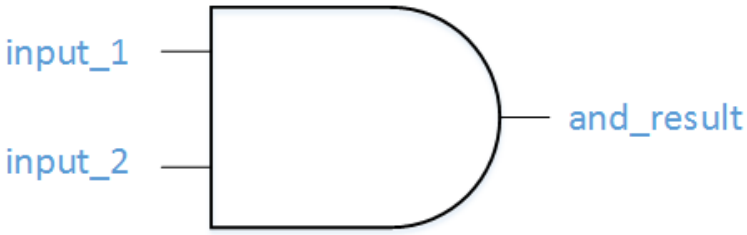
Home
The Go Board
FPGA 101
VHDL
Verilog
YouTube Channel
GitHub
Patreon *NEW*

Introduction to Verilog

Verilog is a type of Hardware Description Language (HDL). Verilog is one of the two languages used by education and business to design FPGAs and ASICs. If you are unfamilliar with how FPGAs and ASICs work you should read this page for an introduction to FPGAs and ASICs. Verilog and VHDL are the two most popular HDLs used. Compared to traditional software languages such as Java or C, Verilog works very differently. Let's get started by looking at a simple example.

First we will create a Verilog file that *describes* an And Gate. As a refresher, a simple And Gate has two inputs and one output. The output is equal to 1 only when both of the inputs are equal to 1. Below is a picture of the And Gate that we will be describing with Verilog.



An And Gate

Let's get to it! One fundamental unit of Verilog is called a wire. For now let's assume that a wire can only be a 0 or a 1. Here is some basic wire logic:

```
1  wire   and_temp;
2  assign and_temp = input_1 & input_2;
```

We are creating a wire called and_temp on the first line of code. On the second line of the code, we are taking the wire that we created and we are *assigning* the wire. To assign it, we are using the Boolean AND function which in Verilog is the Ampersand (&). If you were to describe the code shown above, you might say, "The signal and_temp *gets* input_1 AND-ed with input_2."

Input_1 and Input_2 are inputs to this piece of Verilog Code. Let's show the complete list of inputs and outputs. This is done in the *module* definition. Module is a reserved keyword in Verilog which shows the creation of a block of code with defined inputs and outputs.

```
1  module example_and_gate
2    (
3      input_1,
4      input_2,
5      and_result);
6
7    input  input_1;
8    input  input_2;
9    output and_result;
```

This is your basic module. It defines our module called example_and_gate and 3 signals, 2 inputs and 1 output. Let's put everything together to finish the file. The only thing we are missing is the assignment of the output *and_result*. One other note, // in Verilog is used for a comment.

```
1  //////////////////////////////////////////////////////////////////////////////
2  // File Downloaded from http://www.nandland.com
3  //////////////////////////////////////////////////////////////////////////////
4  module example_and_gate
5    (
6      input_1,
7      input_2,
8      and_result);
9
10   input  input_1;
11   input  input_2;
12   output and_result;
13
14   wire   and_temp;
15
16   assign and_temp = input_1 & input_2;
17
```

```
18    assign and_result = and_temp;
19
20  endmodule // example_and_gate
```

Congratulations! You have created your first Verilog file.

Does it seem like you had to write a lot of code just to create a stupid and gate? First of all, and gates aren't stupid. Secondly, you are correct, HDLs take a lot of code to do relatively simple tasks. You can take some comfort in the fact that Verilog is at least less verbose than VHDL. Get used to the fact that doing something that was very easy in software will take you significantly longer in an HDL such as Verilog or VHDL. But just ask some software guy to try to generate an image to a VGA monitor that displays Conway's Game of Life and watch their head spin in amazement! By the way, that video is created with an FPGA. You will be able to do that soon enough!

Next we will discuss another fundamental Verilog keyword: ALWAYS

Help Me Make Great Content! **Support me on Patreon!** **Buy a Go Board!**