INTERNET ARCHIVE
WayBackMachine
http://wiki.opengraphics.org/tiki-index.php?page=Verilog+Lesson+1
Go
34 captures
24 Jul 07 - 2 Apr 12
JUL SEP
18
2008 2010

Verilog Lesson 1

Verilog Lesson 1 by T. Miller
Picking something out of a hat, I thought today, it would be good to start by discussing nothing but signals. Signals are binary digital numbers. They're wires that go between logic gates. They carry information without performing any computation. Signals are very important for transporting data, and they underlie everything else that we'll do.

The most basic kind of signal is the "wire". The wire is used for continuous assignment. That is, if you have an expression, you can associate the result of that expression with a named wire, and then you can use that wire name in another expression. It's a lot like a variable, but you cannot reassign it. The value of the wire is directly a function of what's assigned to it, and you can only assign one thing.

Here's a single-bit wire:

```
wire a;
```

You can assign some other signal, say 'b', to it, like this:

```
assign a = b;
```

Or you can define and assign it in the same statement:

```
wire a = b;
```

You can have multi-bit wires:

```
wire [3:0] c;
```

A multi-bit wire is also called a "bus". Assigning to them just the same:

```
wire [3:0] d;
assign c = d;
```

The range of bits in a wire can any two numbers. That is, these are also valid:

```
wire [11:4] e;
wire [0:255] f;
```

Given a wire, you can select individual bits, which is to say that you can pick out a bit from a vector of bits:

```
wire g;
assign g = f[2];  // assign bit 2 of f to g;
```

The bit-select can be variable:

```
wire [7:0] h;
wire i = f[h];
```

You can do bit slices, but the range must be constants:

```
wire [3:0] j = e[7:4];
```

You can also have arrays of wires (in most dialects of Verilog):

```
wire [7:0] k [0:20];  // An array of twenty-one 8-bit busses
```

There's another kind of signal element called a "reg". A reg is assigned in behavioral code, which we'll explain later. A reg will be much like a wire for behavioral code that is combinatorial, or it will correspond to a physical register (a flip-flop or group of flip-flops) in synchronous logic.

Regs are defined similarly to wires:

```
reg [3:0] m;
reg [0:100] n;
```

They can be treated just like wires on the right-hand-side of an expression:

```
wire [1:0] p = m[2:1];
```

You can also have arrays of regs, which we usually refer to as a RAM:

```
reg [7:0] q [0:15];  // 16-entry memory of 8-bit words
```

One other sometimes useful kind of signal is the integer. It's like a reg, but it's fixed at 32 bits and is the only signed datatype.

```
integer loop_count;
```

Verilog allows you to group logic into blocks. A block of logic is called a "module". Modules have inputs and outputs that behave very much like wires.

Modules first list their ports:

```
module my_module_name (port_a, port_b, w, y, z);
```

And then define the directions of their ports:

```
input port_a;
output [6:0] port_b;
input [0:4] w;
inout y;  // bidirectional signal, mostly used on external pins
```

Later, we'll see that you can override the wire-ness of an output for behavioral logic by also declaring it to be a reg:

```
output [3:0] z;
reg [3:0] z;
```

Now we can treat inputs like wires:

```
wire r = w[1];
```

And make continuous assignments to outputs:

```
assign port_b = h[6:0];
```

The last kind of 'signal' we'll talk about in this lesson is the constant sort of signal, which is to say a number:

```
wire [12:0] s = 12;                    // 32-bit decimal that gets truncated
wire [3:0] t = 4'b0101;                // explicitly sized 4-bit binary
wire [63:0] u = 64'hdeadbeefcafebabe;  // 64-bit hex
```

If you don't specify a size, it's 32 bits, which can cause problems for larger-sized bit vectors.

These numbers are just like any sort of number, which can be used in an expression. Skipping ahead to math, for instance, you can add 1 to a number:

```
wire [3:0] aa;
wire [3:0] bb;
assign bb = aa + 1;
```

Created by: hamish last modification: Saturday 06 of October, 2007 [13:34:13 UTC] by asdf

Traduisez cette page en français

Tradurre questa pagina in italiano

Traduza esta página em portuguêses

翻译这页成汉语 **(CN)**

日本語にこのページを翻訳しなさい **(Nihongo)**

한국인으로 이 페이지를 번역하십시요 **(Hangul)**

[Execution time: 0.70 secs] [Memory usage: 7.47MB] [157 database queries used] [GZIP Disabled] [Server load: 0.28]