

MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE  
6.111 - Spring 2003

**Introductory Digital Systems Laboratory**

Negative True and VHDL

Donald E. Troxel  
January 10, 1999  
Revised 01/20/02

Signals in VHDL are inherently positive true. Normally this is not a problem as most signals in digital systems are positive true. That is, a signal is high when it "happens". As such, it is good practice to have the signal without a bubble at the input or output of a device or block. It is also good practice to have the signal name without a beginning slash or equivalent beginning to reinforce the fact that the signal is positive true.

This is standard with VHDL. Indeed, a slash is not a legal character for use as a signal name. However, sometimes a signal is desired to be negative true. That is, a signal is low when it "happens". As such, it is good practice to have the signal with a bubble at the input or output of a device or block. It is also good practice to have the signal name with a beginning distinctive part such as a slash, or, perhaps, a letter n, or even a syllable such as n\_, not\_, or neg\_true\_. For example, the following are good names to use for the signal foo:

```
/foo  
nfoo  
n_foo  
not_foo  
neg_true_foo
```

The first is impossible since a slash is not a legal character in a VHDL identifier. The second could be troublesome for signals whose names naturally begin with the letter n. Either of the remaining three are ok, except that the last one is, perhaps, excessively verbose. We will use (and recommend for consistency) the scheme used for the middle name, i.e., using n\_ as a prefix to the signal name.

If we make up a block from an entity "by hand", then we are free to choose a bubble or not, depending on whether the signal is negative or positive true. Most tools which create block symbols from an entity definition automatically do not produce bubbles on the inputs or outputs of the block. So, if we use these tools (which is convenient when they are available), then we have to forgo the use of bubbles and the readability which they provide.

Please bear in mind that the equation as listed in the report file has little, if anything, to do with the negative true or positive true property of a signal. The polarity of the signal is, in fact, determined by your interpretation of the signal, not by the equation or the presence or absence of an inverting architecture of the device.

We start with the format of the equations as used in the report file rather than the VHDL format as it is easier to understand (and type). Perhaps it would be nice to have a tool (program) to automatically convert this format into that required (accepted) by VHDL.

Consider the equation,  $x = a1 * b1$ . Is this positive true or negative true? Our only clue is the choice of signal name! Since the signal name does not begin with `n_` we will call it positive true. The signal, `x`, is true (high) when both `a` and `b` are true (high).

Alternatively, consider the equation,  $n\_y = a2 * b2$ . This is negative true as we have followed the above recommended convention in naming the signal. Were we to use a slash, we would have written  $/y = a2 * b2$ . This would mean that the signal, `/y`, is false (low) when `a2` and `b2` are true (high). This is precisely what we mean by the above equation,  $n\_y = a2 * b2$ .

To express a positive true signal in VHDL (using VHDL syntax from now on), we would (naturally) write

```
x <= a1 and b1;
```

We have two ways (perhaps there are more) of expressing a negative true signal in VHDL. One way is to declare a signal, `y`, in the architecture and to write (in VHDL syntax, again)

```
y <= a2 and b2;  
n_y <= not y;
```

Another way to express a negative true signal is to do this in one fell swoop and write

```
n_z <= not (a3 and b3);
```

A VHDL file to illustrate these methods is

```
library ieee;  
use ieee.std_logic_1164.all;  
entity neg is  
  port (a1, b1, a2, b2, a3, b3 :in std_logic;  
        x, n_y, n_z: out std_logic);  
end neg;  
architecture equations of neg is  
  signal y: std_logic;  
begin  
  x <= a1 or b1;  
  y <= a2 or b2;  
  n_y <= not y;  
  n_z <= not (a3 or b3);  
end equations;
```

If we run galaxy and choose C16L8 as the device, we get the following equations from the report file:

$$\text{/x} = \text{/a1} * \text{/b1}$$

$$\text{/n\_y} = \text{b2} + \text{a2}$$

$$\text{/n\_z} = \text{b3} + \text{a3}$$

This looks suspiciously as if all three were negative true. However, these equations were chosen as the C16L8 only has an inverting architecture. If we choose the device, 16V8, then the equations from the report file are

$$\text{n\_y} = \text{/a2} * \text{/b2}$$

$$\text{n\_z} = \text{/a3} * \text{/b3}$$

$$\text{/x} = \text{/a1} * \text{/b1}$$

Well, these are different equations, not what one would expect, as the first two look like positive true and the last looks like negative true, which are just the opposite of what we know to be the case. However, the two chips work exactly the same! The difference is that the signals n\_y and n\_z use noninverting architectures while the signal x uses an inverting architecture. So, the equations do not provide a clue as to positive or negative true. Neither does the architecture. The only clue is provided by the signal names.

Be careful. Different examples could make it appear as if the equations in the report file correspond to negative true or positive true.