UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

**CS152** **D. Patterson**
**Spring 95** **S. Kong**

**Helpful Tips: Powerview**

## A. VHDL Tips

1. Two of the Powerview VHDL text available
   a. The VHDL Cookbook by Peter Ashenden
   b. VHDL: Hardware Description and Design by Roger Lipsett, Carl Schaefer, and Cary Ussery

2. Syntax details
   a. VHDL is case insensitive.
   b. The only symbol which can be used in a name is underscore. However, a name must not start or end with an underscore.

3. Tips on variable assignment in Powerview VHDL
   a. The type TIME can only take on whole number values. If you want to have a 2.5 ns delay, use 2500 ps.
   b. The assumed base for integers is decimal. Here are some examples:

```
variable foo: integer := 2#0010110#;    -- base 2 (binary)
variable bar: integer := 45;            -- base 10 (decimal)
variable baz: integer := 16#FEED#;      -- base 16 (hexidecimal)
```

   c. To assign a constant value to a vlbit vector, use double quotes preceded by an X for a hex value and B for a binary value, or an O for an octal value.

```
iable zoom:  vlbit_1d(31 downto 0) := X"0BAD_1DEA";
gnal  mooz:  vlbit_1d(6 downto 0) := B"1_00_11_11";
```

   **Note the values may contain underscores, which don't affect the values, but can be helpful for dividing the field in a meaningful way.*

   d. The assignment

```
variable data:  vlbit_1d(3 downto 0) := X"A";
```

   is actually a short cut in Powerview VHDL for typing:

```
variable data:  vlbit_1d(3 downto 0) := ('1','0','1','0');
```

   You can build a vlbit_1d from multiple vlbits by using &, the concatenation operator.

```
variable feeb:  vlbit_1d(3 downto 0) := '1' & '0' & '0' & '1';
```

   This is most useful when constructing a vlbit vector out of individual vlbit variables or vlbit vectors.

4. Relational operator

/= is the "not equals" relational operator in VHDL. The others are =, <, <=, >, >=. The logical operators AND, OR, NAND, NOR, XOR, and NOT can be used on the resulting BOOLEAN values when the relational operators are used.

5. Process statements

If a process has a sensitivity list (i.e. has some signals as arguments after the process statement), it has an implicit WAIT statement as its last statement which waits for any of the signals in the list to change. Such a process must not contain an explicit WAIT statement or call a procedure that contains a WAIT statement. Each process in a VHDL program is invoked once at the beginning of simulation and continues until it reaches an implicit or explicit WAIT statement. Every process loops perpetually stopping only at implicit or explicit WAIT statements. After the wait conditions are satisfied, execution continues at the next statement after the WAIT or at the top of the procedure if the WAIT is at the end.

## B. VIEWSIM

1. Details
   a. Viewsim rounds delays to the nearest .1ns.

2. Helpful keyboard commands
   a. Open a Viewsim window for file <filename>.
      **simu <filename>**
   b. Rereads the .vsm file without having to close and open the viewsim file every time.
      **network <filename>**
   c. Save the output of a simulation into a file, type:
      **logfile <filename.log>**
      perform the simulation, and then type:
      **logfile**
   d. Execute a command file you can just type the name of the command file:
      **<filename.cmd>**

**Note: These commands can be included in a command file for convenience.*

3. Some helpful tips for assigning values in a .cmd file:
   a. You can store all of the test patterns in a (or several) separate file and do assignment as follows:

      **|example use of input files**
      **(assign x < input_file.pat; assign z < input_file.pat; sim) * 2**

   b. x is assigned the first value in the file input_file.pat, then z is assigned the next, then the values are simulated. This loop is repeated twice. An example input_file.pat is:

      **|input_file.pat : an input pattern file example**
      **|x input value**
      **|z input value**
      **0**
      **1**
      **1**
      **1**

      **Note two files could have been used, one for x and one for z.*

c. You can increment or decrement through the values of a bit vector. This is an easy way to exhaustively test all input patterns. See the command file in the Powerview VHDL Tutorial for a good example.

d. Above example could use this feature if we define a vector in the .cmd file as:

**v input_vector x z**

Then one can use the dec or inc commands as shown in the Powerview VHDL Tutorial. (This tutorial is highly recommended)

## C. VIEWDRAW

1. Keyboard commands for all window operations.

For example instead of performing**:**

**Window->Open->Viewdraw->Schematic**

and then specifying the file name, you can just type in:

**sc <filename>.**

Where **<filename>** is the name of the schematic you wish to load. In fact, there are some keyboard commands for which there are no window operations. To get the keyboard command for a window operation, select **Help** and then select the window operation that you want to know the short cut for.

2. Symbols size
While in the schematic you can push the left mouse button and use the X, Y, DX, DY information on the bottom of the screen to determine a good size. Then when in the symbol use the size keyboard command to size the symbol area. zsi can also be used to change the schematic to some arbitrary size (instead of using the A sheet, B sheet, etc..).

**Note: Do not make any schematic you ever turn in unreadable. If you make a schematic too large, labels will not be legible on a printout.*

3. Some helpful keyboard commands
   a. Create the .vsm file for the current schematic on screen.
      **mex vsm**
   b Opens up schematic <filename>
      **sc <filename>**
   c. Opens up symbol <filename>
      **sym <filename>**

4. Pin numbers

When you start making symbols with a huge number of pins (which you will eventually) you will notice that the pinorder attribute has a length limit. To solve this problem, you can type the keyboard command save pin. This will create a .pin file in the sym directory which will contain the pin order information so the pinorder attribute will not be necessary. This file can then be edited with a text editor, so you can have as many pins as you want with long names.

**D. Good Schematics**

A good clean schematic should have the following characteristics:

1. Name of the label should be meaningful. For examples, the name of the bus I[31:0] does not tell the reader anything. On the other hand, the name, RegToAlu[31:0] says a lot about the purpose of the bus.

2. Although it is impossible to avoid nets and buses from crossing, a little patience and thinking should be paid to prevent the schematics from looking like noodles.

3. All connection should be made with T-intersection. By this I mean that

the left example clearly indicates the nets are connected. The example on the right could mean either the nets are crossing or they are connected. Therefore, one should follow the convention only nets with T-intersection are connected.

4. Information should flow from one direction to the other. For example, datapath flow could flow from left to righ or up to down. It should not, however, flow from left to right, then to up, down, and to left again.

5. Connectivity. All datapath signal should be connected and all control signals should not be connected. Although nets with the same name will be connected by Powerview and it will make the schematics extremely clean, it is also very hard to read. Therefore, all datapath signals should be connected. On the other hand, control signals do not have to be connected because that will make the schematics messier. In fact, we will insist that they aren't connected.

6. Schematics should have reasonable spacing between components. Of course there is no absolute rule about this; use your own judgement.

7. Labels should be large enough to be readable on printouts. To change the size of labels, select the label, and type "si" and Powerview will prompt you for a number. Default is size 10.

8. In general, make your homework as easy to read and grade as possible.