11

Digital Vector File Format

This chapter describes how to perform vector checks and apply stimuli according to digital vectors using the Virtuoso® UltraSim™ simulator. To process digital vector file formats, the following statement needs to be specified in the netlist:

```
.vec 'vector_filename' [HLCheck = 0|1] [autostop=true | false]
```

Description

`HLCheck` is a special flag that you need to set to generate the vector output check for H and L states of input signals. Bidirectional and output signals always check H and L states and are unaffected by the `HLCheck` flag. Normally, you do not need to use the `HLCheck` flag unless it is necessary to check if input signals are shorted in the netlist. The output resistance of H and L states for input signals can be specified by the `hlz` statement.

Each `.vec` card can specify only one vector file. If a netlist needs to include multiple vector files, multiple `.vec` cards can be used. For example, if a netlist needs to include three vector files, then it needs to use three `.vec` cards:

```
Card 1: .vec 'file1.vec'
Card 2: .vec 'file2.vec'
Card 3: .vec 'file3.vec'
```

The Virtuoso UltraSim simulator handles the vector file content as case insensitive, except when called in Virtuoso Spectre® mode. For Spectre mode, use the `-spectre` option or input file name extension `*.scs`.

Arguments

vector_filename	The filename of the digital vector file
HLCheck = 0 | 1	Special flag which turns on checking for the H and L states for input signals (default = 0)

`autostop=true|false`

- **false** tells the Virtuoso UltraSim simulator to use the end time from the `.tran` or `tran` statement (default).
- **true** tells the simulator to use the last specified time point in the vector file as the end time. If multiple `.vec` files are specified, and `autostop=true` is in one or all `.vec` statements, the simulator takes the longest time point available in the `.vec` files and uses it as the end time.

  Note: `autostop` can also be used when loading `.vcd` and `.evcd` files. For more information about these files, refer to "Netlist Formats Support" on page 37.

Example

```
.vec "vec1.vec" autostop=true
```

Tells the Virtuoso UltraSim simulator to replace the end time with the time from the `vec1.vec` file (that is, the time from the `vec1.vec` file is used as the transient simulation end time).

The digital vector file is described in detail in the following sections:

General Definition

Comment Line

A comment line begins with a semicolon '`;`'.

Note: A semicolon is only used in digital vector file format comment lines.

Continuous Line

A continuous line is indicated by a plus sign '+'.

The maximum length of a line is 1024 characters. If a card is longer than 1024 characters, you need to use the continuous line for the card.

Tip

For a long identifier (for example, a 1280-bit vector bus) that cannot fit on a single line, use the \ sign after the last bit. Do not use a space between the last bit and the \ sign. Put a space in front of the continuous vector or use a + sign. If you use a + sign, the continuous vector is treated as another vector bus.

Signal Mask

A signal mask can be used to specify the effective range of the current statement in a vector file (statement applies to specific signals). The Virtuoso UltraSim simulator matches the signals according to the signal definition order in the `radix`, `vname`, and `io` statements. For the corresponding signal, a value of 1 indicates the statement is valid and a value of 0 indicates the statement is ignored. Based on the size of the vector specified in the `radix` statement, the signal mask value can range from 0 to 1 for 1bit, 0 to 3 for 2bit, 0 to 7 for 3bit, and 0 to 9 or A to F for 4bit.

Example

```
radix 2 2 4
io i i o
vname A[1:0] B[1:0] P[3:0]
vih 2.5 3 0 0
vih 3.3 0 3 0
trise 0.5 1 2 0
chk_window -1 5 1 0 0 F
```

For more information about the statements used in this example, refer to "Vector Patterns" on page 424 and "Signal Characteristics" on page 438.

Vector Patterns

In this section, vector patterns (such as signal sizes, directions, names, and check windows) are defined. The Virtuoso UltraSim simulator supports the following digital vector pattern statements:

radix

`radix` *`vector1_size1 vector2_size2 ...vector_sizeN`*

Description

Specifies the size (in bits) of the vector. This statement must be located before any other statements, and can only be specified once. Valid vector sizes include 1 (binary), 2, 3 (octal), or 4 (hexadecimal).

Tip

If the `radix` of the vector is larger than 1, the name of this vector specified in vname must be indexed as [msb:lsb] or [lsb:msb]. If the `radix` is 4, the `vname` can use names such as name[3:0] and name[0:3].

Examples

The following example

```
radix 2 2 4
```

contains three vectors: Two 2-bit vectors and one 4-bit vector.

Note: The examples presented in the rest of this chapter follow this format.

In the next example

```
radix 2 11 1111
```

also contains three vectors, two 2-bit vectors and one 4-bit vector, but in a different format.

io

`io` *`type1 type2 ...typeN`*

Description

`io` defines the type of vector. It can be the `i` (input), `o` (output), or `b` (bidirectional) type. If this statement is specified more than once, the last value is used.

Notes

- Use the `enable` statement to specify the control signal for the bidirectional vector (`b`). If this specified control signal is not found, the Virtuoso UltraSim simulator issues an error.
- If the control signal of the bidirectional vector is not specified by the `enable` statement, the Virtuoso UltraSim simulator treats it as an input signal.

Example

```
radix 2 2 4
io i i o
```

The first and second vectors are input vectors, and the third vector is an output vector.

vname

vname name1 name2 ... nameN

Description

vname assigns a name to each vector. For a single bit vector, it can have the following naming format: Va, Va[0:0], or Va[[0:0]]. For multiple bit vectors, the naming formats include: Va[2:0], Va[[2:0]], Va[0:2], or Va[[0:2]]. Each naming format is given a different resulting name. If this statement is specified more than once, the last value is used.

Hierarchical signal names are also supported by vname. That is, you can apply vector stimuli or perform a vector check on the internal signals of instances. When mapping hierarchical signal names, the default delimiter is a period (.). You can change the value of the delimiter using the hier_delimiter option in the analog netlist. The hier statement can be used to enable or disable this option.

Table 11-1 vname Vector Names

Naming Format	Resulting Names
Va[2:0]	Va2, Va1, Va0
Va[[2:0]]	Va[2], Va[1], Va[0]
Va[0:2]	Va0, Va1, Va2
Va[[0:2]]	Va[0], Va[1], Va[2]
X1.Va[0:2]	Internal signals Va0, Va1, and Va2 of instance X1
TOP.X1.Va[[0:2]]	Internal signals Va[0], Va[1], and Va[2] of instance TOP.X1

Tip

If the radix of the vector is larger than 1, the name of the vector specified in vname must be indexed as [msb:lsb] or [lsb:msb]. If radix is 4, vname can use names such as name[3:0] and name[0:3].

Examples

In the following example

```
radix 2 2 4
io i i i
```

```
vname va[1:0] vb[[1:0]] vc[[0:3]]
```

tells the Virtuoso UltraSim simulator that the voltage sources in the first vector are named `va1` and `va0`. Voltage sources in the second vector are connected to `vb[1]` and `vb[0]`. The third vector has voltage sources with the names `vc[0]`, `vc[1]`, `vc[2]`, and `vc[3]`.

In the next example

```
radix 2 2 4
io i i o
vname X1.va[1:0] X2.vb[[1:0]] X1.X3.vc<[0:3]>
hier 1
```

tells the simulator the voltage sources in the first vector are mapped to internal signals `va1` and `va0` of instance `X1`. Voltage sources in the second vector are connected to `v[1]` and `vb[0]` of instance `X2`. The third vector defines the output vector check for signals `vc<0>`, `vc<1>` , `vc<2>` , and `vc<3>` of instance `X1.X3`.

hier

```
hier 0|1
```

Description

This option is used to specify whether or not the hierarchical signal name mapping feature is enabled. If `hier` is set to `0`, the hierarchical delimiter (for example, signal period or `.`) is considered to be part of the signal name. The default value is `1` (hierarchical signal name mapping enabled). If this statement is specified more than once, the last value is used.

Example

```
radix 2
io i
hier 0
vname X1.va[1:0]
```

tells the Virtuoso UltraSim simulator to connect the voltage sources with the `X1.va1` and `X1.va0` signals located in the top level of the analog netlist.

tunit

`tunit` *`time_unit`*

Description

Sets the time unit for all time related variables. The time unit can be one of the following: `fs` (femto-second), `ps` (pico-second), `ns` (nano-second), `us` (micro-second), and `ms` (milli-second). The default time unit is 1 ns. If this statement is specified more than once, the last value is used.

Example

```
tunit 1.5ns
```

chk_ignore

```
chk_ignore start_time end_time [mask1 mask2 ... maskN]
```

Description

`chk_ignore` specifies a window for ignoring output vector checks. A mask can be provided to specify which vector and bit to apply. If the mask is not specified, the setting applies to all output vectors. The `start_time` and `end_time` arguments must be specified. To define multiple time windows for ignoring output vector checks, use multiple `chk_ignore` statements.

Arguments

`start_time`	Defines the start time for the window used to ignore the output vector checks (use tunit to define the `start_time` units).
`end_time`	Defines the end time for the window used to ignore the output vector checks (use tunit to define the `end_time` units). You can use `end_time=-1` to ignore the entire transient time.

Example

```
tunit 1n
chk_ignore 0 100 0F30          ; 0F30 is a signal mask
chk_ignore 3e+2 500 0F30
chk_ignore 0 -1 F000           ; F000 is a signal mask
```

tells the Virtuoso UltraSim simulator to ignore the output vector check for signals specified by the mask `0F30` in the time windows 0 ns to 100 ns and 300 ns to 500 ns, and to ignore the entire transient time for the signals specified by the mask `F000`.

chk_window

```
chk_window start_time end_time steady [period=const [first=const] ] [mask1 mask2
     ... maskN]
```

Description

`chk_window` specifies a window for vector checking. The Virtuoso UltraSim simulator only checks the signal states within this window. The signal states outside the window are ignored. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all output vectors. The checks occur at every time point specified in the vector file or as defined by the `period` and `first` arguments.

Setting the `period` argument activates periodic window checking. If `period` is not defined, the `first` argument is ignored by the simulator.

Note: To activate periodic window checking, you need to include the `"period="` and `"first="` keywords.

Arguments

`start_time`	Defines the window start time at which the window starts at time `vec_time-start_time`. If the `period` argument is defined, `vec_time` is the first time point defined by the `first` argument, and the vector checks are repeated according to the value of `period`. If the `period` argument is not defined, `vec_time` is the time point defined in the vector file.
`end_time`	Defines the window end time at which the window ends at time `vec_time+end_time`.
`steady = 0 | 1`	Can be set to `0` or `1`. If set to `0`, then the vector check passes as long as the signal has reached the desired state once. If set to `1`, then the signal remains in the desired state for the entire window period to pass the vector check.
`period`	Activates periodic window checking and defines its time period.
`first`	Defines the first check point for periodic window checking (only valid when the `period` argument is also defined).

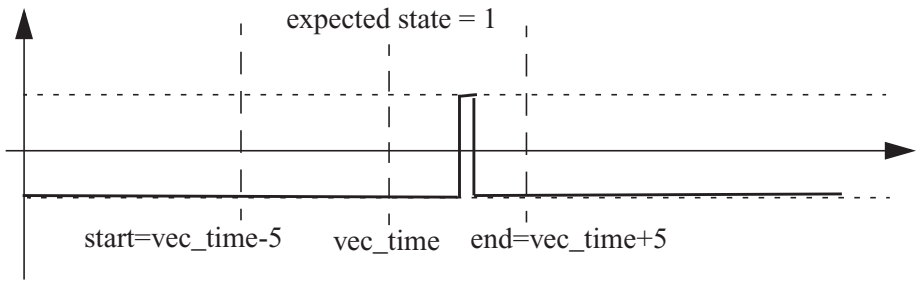
Examples

The following example

```
chk_window 5 5 0
```

tells the Virtuoso UltraSim simulator to set the steady state to `0`, so the waveform passes the vector check (see Figure 11-1 on page 433).

Figure 11-1 Vector Check with chk_window Steady State Set to 0

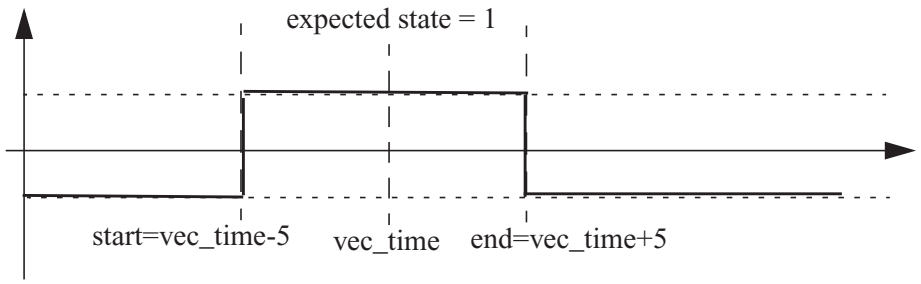


In the next example

```
chk_window 5 5 1
```

tells the simulator to set the steady state to `1`, which means the signal needs to stay at state `1` for the whole window period to pass the vector check, as shown in Figure 11-2 on page 433. If the signal is as shown in Figure 11-1 on page 433, the vector check fails.

Figure 11-2 Vector Check with chk_window Steady State Set to 1



In the next example

```
radix 1 1 1 1
vname ph1 d q qb
io i i o o
tunit 10ns
chk_window -10 30 1 period=100 first=5 0 0 1 0
```

tells the simulator to activate periodic window check for signal `q`. The vector check points start at 50 ns and repeat every 1 us.

In the next example

```
chk_window -10 30 1 first=5 0 0 1 0
```

tells the simulator to ignore the `first` argument because a valid `period` argument has not been specified.

enable

```
enable 'enable_signal_expr' [mask1 mask2 ... maskN]
```

Description

The `enable` statement connects the enable signal, or enable signal expression, to the bidirectional vector. The resulting value 1 (H) enables the output signal. The controlled bidirectional signal is regarded as an input for other values.

You can provide a mask to specify to which vector and bit the enable signal expression applies. If the mask is not specified, the setting applies to all bidirectional vectors. Also, if this statement is specified more than once, the last value is used.

The enable signal can be used in a vector file or an analog netlist. When an enable signal is used in an analog netlist, it can also be defined as an output signal for a vector check or only used as an enable signal. The avoh and avol statements can be used to define the logic high and low voltage thresholds for the analog signal.

Note: The enable signal cannot be defined as a bidirectional signal.

Bit-wise logic operators are supported in an enable signal expression: `&` (AND), `|` (OR), `^` (XOR), and `~` (NOT). Additional operators can be created using a combination of the supported operators. The order of processing for the logic operators is NOT > AND > OR, XOR (OR and XOR are processed at the same time). You can use parentheses `()` around the operators to change the processing order.

Note: You need to use single quotation marks `''` for enable signal expressions.

Examples

The following example

```
radix 1 1 1 1
io i i b o
vname en in bi out
enable en 0 0 1 0
```

tells the Virtuoso UltraSim simulator to set `en` as the enable signal for `bi`, and when `en` is in 1 (or H) state, `bi` becomes the output signal. When `en` is in 0 (or L, X, U) state, `bi` changes to the input signal. When `en` is in Z state, the `bi` (input and output) signal also changes to Z state.

In the next example

```
radix 1 1 1
io i b o
vname en bi out
enable ~en 0 1 0
```

tells the simulator to set `en` as the enable signal for `bi`. Unlike the first example, this enable signal name contains a `~` sign, which reverses the state to control the bidirectional signal. Now when the enable signal is in 1 (or H) state, the `bi` becomes an input signal.

In the next example

```
radix 1 1 1
io b b o
vname bi_1 bi_2 out
enable ana_en1 0 1 0
enable '(ana_en1 | X1.ana_en2) & out' 1 0 0
```

tells the Virtuoso UltraSim simulator that the `ana_en1` and `X1.ana_en2` enable signals originate in the analog netlist, and `X1.ana_en2` is a hierarchical signal. Although the `out` signal is used as an enable signal, the simulator still performs a vector check.

period

`period` *time*

Description

`period` is used to specify the time interval for tabular data, so that the absolute time is not needed.

If `period` is not specified, then the absolute time must be specified in the tabular data. If it is specified more than once, the last value is used.

Example

```
period 10.0
```

tells the Virtuoso UltraSim simulator that the signal period is 10 ns and the absolute time points are unnecessary.

Signal Characteristics

In this section, signal characteristics containing various attributes for input or output signals (such as delay, rise or fall time, voltage thresholds for logic low and high, and driving ability) are defined. For most of these statements, the mask can be used to apply the specified characteristics to the corresponding signals. The statements are organized into three groups:

- Timing on page 439
- Voltage Threshold on page 446
- Driving Ability on page 455

Note: In the following examples for time-related statements, the time unit is 1 ns if the statement is not specified with tunit.

Timing

Timing characteristics of input or output signals (such as delay, rise time, and fall time) can be specified using the following statements. The values of these statements can be positive or negative. For the delay timing characteristics, the negative value is used to advance the signals by a specified time. For the rise and fall timing characteristics, the negative value is the same as the positive one.

- idelay on page 440
- odelay on page 441
- tdelay on page 442
- slope on page 443
- tfall on page 444
- trise on page 445

Note: The Virtuoso UltraSim simulator checks whether the values of the `trise`, `tfall`. and `slope` statements are reasonable (warning message is issued when the defined value is too small or large).

idelay

`idelay` *`time_value [mask1 ... maskN]`*

Description

`idelay` specifies the delay time for the corresponding input signal. If a bidirectional signal is specified, this applies only to the input stage of the bidirectional signal. The default value is 0.0, if `idelay` or `tdelay` is not set.

Example

```
idelay 5.0
```

tells the Virtuoso UltraSim simulator to delay all input signals by 5 ns, whereas

```
idelay -5.0
```

tells the simulator to advance all input signals by 5 ns.

odelay

`odelay` *`time_value [mask1 ... maskN]`*

Description

`odelay` specifies the time delay for the corresponding output signal. If a bidirectional signal is specified, this applies only to the output stage of the bidirectional signal. The default value is 0.0, if `odelay` or `tdelay` is not set.

Example

```
odelay 5
```

tells the Virtuoso UltraSim simulator to delay all output signals by 5 ns, whereas

```
odelay -5.0
```

tells the simulator to advance all output signals by 5 ns.

tdelay

`tdelay` *`time [mask1 mask2 ... maskN]`*

Description

`tdelay` specifies the delay time for corresponding vectors. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all vectors (input, output, and bidirectional).

If `tdelay` is not specified, the default value is 0.0. If this statement is specified more than once, the last value is used for the active mask. This statement can also overrule the value previously set by the `idelay` or `odelay` statements.

Examples

```
tdelay 5.0
```

tells the Virtuoso UltraSim simulator to advance all signals by 5 ns.

```
tdelay -5.5  3 0 F
```

tells the simulator to advance all signals, specified with a mask, by 5.5 ns.

slope

`slope` *`time [mask1 mask2 ... maskN]`*

Description

`slope` sets the input vectors rise and fall time. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If this statement is not specified, then the default value of 0.1 ns is used. If this statement is specified more than once, the last value is used for the active mask. This statement can also overrule the value previously set by the `trise` or `tfall` statements.

Examples

```
slope 0.05
```

or

```
vname va[1:0] vb[[1:0]] vc[[0:3]]
io i i o
slope .025 1 3 5
```

The least significant bit, `va0`, of the first input vector and the two bits, `vb1` and `vb0`, of the second input vector have a `trise` and `tfall` of 0.025 ns. The third vector is an output vector (specified in the `io` statement), so it is not affected by the `slope` statement.

tfall

`tfall` *`time [mask1 mask2 ...maskN]`*

Description

`tfall` specifies the falling time of the input vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

The value from the `slope` statement is used, if `tfall` is not specified. If this statement is specified more than once, the last value is used for the active mask. This statement can also overrule the value previously set by the `slope` statement.

Examples

The following example

```
tfall 0.05
```

tells the Virtuoso UltraSim simulator that all input vectors have a fall time of 0.05 ns.

In the next example

```
vname va[1:0] vb[1:0] vc[0:3]
tfall 0.1  0 2 0
```

the most significant bit, `vb[1]`, of the second input vector has a fall time of 0.1 ns. The fall time of `vb[0]` and other input vectors remains the same.

trise

trise time [mask1 mask2 ...maskN]

Description

`trise` specifies the rise time of the input vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If `trise` is not specified, the value from the `slope` statement is used. If this statement is specified more than once, the last value is used for the active mask. This statement can also overrule the value previously set by the `slope` statement.

Examples

The following example

```
trise 0.1
```

or

```
trise -0.1
```

tells the Virtuoso UltraSim simulator that all input vectors have a rise time of 0.1 ns.

In the next example

```
vname va[1:0] vb[1:0] vc[0:3]
trise 0.1  0 3 0
```

the two bits of the second input vector has a rise time of 0.1 ns. The `trise` of the other input vector remains the same.

Voltage Threshold

When converting input vectors to stimuli or performing an output vector check, the voltage threshold for logic low and high can be specified using the following statements:

- vih on page 447
- vil on page 448
- voh on page 449
- vol on page 450
- avoh on page 451
- avol on page 452
- vref on page 453
- vth on page 454

vih

`vih` *`voltage [mask1 mask2 ...maskN]`*

Description

`vih` specifies the logic high voltage of the input vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If `vih` is not specified, the default voltage is 3.3. If this statement is specified more than once, the last value is used for the active mask.

Examples

```
vih 5.0
```

or

```
vih 5.5  3 1 0
```

vil

`vil` *`voltage [mask1 mask2 ... maskN]`*

Description

`vil` specifies the logic low voltage of the input vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If `vil` is not specified, the default voltage is 0.0. If this statement is specified more than once, the last value is used for the active mask.

Examples

```
vil 0.25
```

or

```
vil 0.5  3 0 0
```

voh

`voh` *`voltage [mask1 mask2 ... maskN]`*

Description

`voh` specifies the logic high voltage of the output vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all output vectors.

If `voh` is not specified, the default voltage is 3.3. If this statement is specified more than once, the last value is used for the active mask.

Examples

```
voh 5.0
```

or

```
voh 5.5  0 0 F
```

vol

`vol` *`voltage [mask1 mask2 ... maskN]`*

Description

`vol` specifies the logic low voltage of the output vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all output vectors.

If `vol` is not specified, the default voltage is 0.0. If this statement is specified more than once, the last value is used for the active mask.

Example

```
vol = 0.05
voh = 1
```

tells the Virtuoso UltraSim simulator to interpret all output signals with values below 0.05 V as 0, print all signals above 1 V as 1, and all signals between 0.05 V and 1 V are U.

avoh

```
avoh voltage [ signal_name1 signal_name2 ... signal_nameN ]
```

Description

`avoh` specifies the logic high voltage of the signal from the analog netlist, which is not defined in the radix, vname or io statements. You can provide signal names to specify the valid scope for `avoh` (wildcards are supported). A period (`.`) can be used as the hierarchical delimiter to specify the hierarchical signal. If a signal name is not used, the setting applies to all analog signals used in the vector file.

For more information about wildcards, see "Wildcard Rules" on page 41.

Note: A mask cannot be used to specify which vector and bit to apply to the signal (different behavior from other vector format statements).

Example

```
avoh = 1 ana_en* X1.Enanble
```

tells the Virtuoso UltraSim simulator that analog signals `ana_en*` and `X1.Enanble` have a logic high voltage of 1.0.

avol

```
avol voltage [ signal_name1 signal_name2 ... signal_nameN ]
```

Description

`avol` specifies the logic low voltage of the signal from the analog netlist, which is not defined in the radix, vname or io statements. You can provide signal names to specify the valid scope for `avol` (wildcards are supported). A period (`.`) can be used as the hierarchical delimiter to specify the hierarchical signal. If a signal name is not used, the setting applies to all analog signals used in the vector file.

For more information about wildcards, see "Wildcard Rules" on page 41.

Note: A mask cannot be used to specify which vector and bit to apply to the signal (different behavior from other vector format statements).

Example

```
avol = 0.5 ana_en* X1.Enanble
```

tells the Virtuoso UltraSim simulator that analog signals `ana_en*` and `X1.Enanble` have a logic low voltage of 0.5.

vref

`vref` *`node_name [mask1 mask2 ... maskN]`*

Description

`vref` sets the reference node of the input vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If `vref` is not specified, the default value is 0 (that is, the ground). If this statement is specified more than once, the last value is used for the active mask.

Examples

The following example

```
vref 0
```

tells the Virtuoso UltraSim simulator to set the negative node of the vector source to ground.

In the next example

```
vref vss
```

tells the simulator to set the negative node of the vector source to `vss`.

Note: The Virtuoso UltraSim simulator only supports reference node to ground. References to other nodes causes the simulator to issue error messages.

vth

`vth` *`voltage [mask1 mask2 ... maskN]`*

Description

`vth` sets the threshold voltage of the output vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all output vectors.

If `vth` is not specified, the default value is 1.65. If this statement is specified more than once, the last value is used for the active mask.

Examples

```
vth 2.5
```

or

```
vth 2.7 0 0 8
```

Driving Ability

For input stimuli, the output resistance of vector sources can affect Virtuoso UltraSim simulation results. To specify the driving ability of vector sources, use the following statements:

- hlz on page 456
- outz on page 457
- triz on page 458

hlz

`hlz` *`resistance [mask1 mask2 ... maskN]`*

Description

`hlz` specifies the output resistance for the corresponding input vector, but unlike `outz`, this output resistance only applies to the H and L states of the vector. This resistance overwrites the resistance for the H and L states set by `outz`.

You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If `hlz` is not specified, the default value follows `outz`. If `hlz` is set to 0, the Virtuoso UltraSim simulator uses 0.01 instead. If this statement is specified more than once, the last value is used for the active mask.

Examples

```
hlz 1meg
```

or

```
hlz 4.7k 2 2 0
```

outz

`outz` *resistance* [*mask1 mask2 ... maskN*]

Description

`outz` specifies the output resistance for the corresponding input vector. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If `outz` is not specified, the default value is 0.01. If `outz` is set to 0, the default value is used. If this statement is specified more than once, the last value is used for the active mask.

Examples

```
outz 1meg
```

or

```
outz 5.5meg 2 2 0
```

triz

`triz` *`resistance [mask1 mask2 ... maskN]`*

Description

`triz` specifies the output impedance when the corresponding input vectors are in tri-state. You can provide a mask to specify which vector and bit to apply. If the mask is not specified, the setting applies to all input vectors.

If `triz` is not specified, the default value is 1,000 Meg. If `triz` is set to 0, the Virtuoso UltraSim simulator uses 0.01 instead. Also, if this statement is specified more than once, the last value is used for the active mask.

Examples

```
triz 2000meg
```

or

```
triz 550meg 2 2 0
```

Tabular Data

This section describes the values of signals at specified times (absolute or period time modes). For periodic signals, it is unnecessary to specify the absolute time at each time point. The `period` statement can be used to specify the signal period.

Absolute Time Mode

The period is not specified.

```
Time1 vector1_value1 vector2_value1 vector3_value1
Time2 vector1_value2 vector2_value2 vector3_value2
...
TimeN vector1_valueN vector2_valueN vector3_valueN
```

Period Time Mode

The period is specified.

```
vector1_value1 vector2_value1 vector3_value1
vector1_value2 vector2_value2 vector3_value2
...
vector1_valueN vector2_valueN vector3_valueN
```

vector_value can be 0-9, A-F, Z, X, L, H, or U, and is dependent on how `radix` is set.

Description

Tabular data is used to describe the waveform of voltage sources.

Examples

```
; format: time vector
0 000101010
10 011010101
20 000101010
```

or

```
; format: vector
00101010
11010101
00101010
```

Note: This example assumes the period has been set by `period` 10.0.

or

```
; format: time vector
10 02A
20 315
30 02A
```

Valid Values

The valid values in tabular data depend on the `radix` statement setting.

Table 11-2 Tabular Data Valid Values

Value Specified in `radix` Statement	Valid Value
`1`	0, 1
`2`	0-3
`3`	0-7
`4`	0-9, A-F

The values specified in the table above are converted into 0 and 1 states by the Virtuoso UltraSim simulator. The simulator also accepts `L`, `H`, `Z`, `X`, and `U` values.

Vector Signal States

Input

The Virtuoso UltraSim simulator accepts the following signal states for input vector signals.

Table 11-3 Input Vector Signal States

Signal State	Description
`0`	Drive to ZERO (GND)
`1`	Drive to ONE (VDD)
`Z`, `z`	Floating to high-impedance
`X`, `x`	Drive to ZERO (GND)

Table 11-3 Input Vector Signal States, *continued*

Signal State	Description
`L`, `l`	Resistively drive to ZERO (GND)
`H`, `h`	Resistively drive to ONE (GND)
`U`, `u`	Drive to ZERO (GND)

The resistance values of `L` and `H` are set by the `hlz` statement, and the impedance value of `Z` is set by the `triz` statement.

Output

The Virtuoso UltraSim simulator accepts the following signal states for output vector signals.

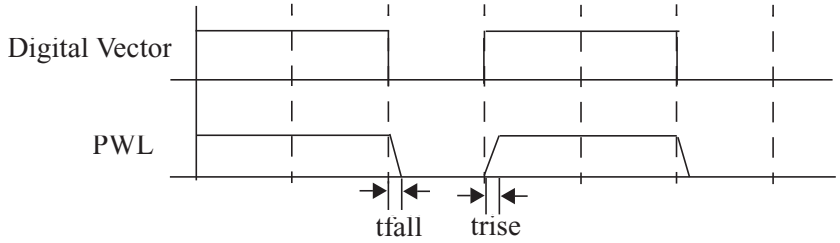
Table 11-4 Output Vector Signal States

Signal State	Description
`0`	Expects ZERO
`1`	Expects ONE
`Z`, `z`	Accepts any signal state
`X`, `x`	Accepts any signal state
`U`, `u`	Accepts any signal state

Digital Vector Waveform to Analog Waveform Conversion

The Virtuoso UltraSim simulator converts the digital vector waveform into a PWL waveform. The rising/falling edge occurs at the switching state point of the digital waveform, as shown in Figure 11-3 on page 462.

Figure 11-3 Conversion of Digital Waveform to PWL Waveform



Expected Output and Comparison Result Waveforms for Digital Vector Files

If a digital vector file contains output or bi-directional vectors, the Virtuoso UltraSim simulator generates two waveform files: One contains all the expected output vector waveforms as specified in the digital vector file and the other contains the waveforms from the comparison results.

You can use the following statement in the digital vector file to enable or disable the simulator from generating these waveforms (default is `1` or enabled).

```
.output_wf 0|1
```

The waveform format is defined by the `wf_format` option in the analog netlist. A maximum of two waveform files are generated for one or more digital vector files. The expected waveform filename is `netlist.vecexp.trn` (PSF, FSDB, etc.) and the output vector is `signal_name_exp`. The comparison waveform filename is `netlist.vecerr.trn` (PSF, FSDB, etc.) and each comparison waveform is `signal_name_err`.

The comparison result values include,

0 – matched

1 – mismatched

X – ignored (output vector = X or bi-directional vector at input stage are possible causes)

In addition to the individual comparison result waveforms, the simulator generates a single `vec_error` waveform to indicate the overall comparison results. Waveform `vec_error` equals `1` when any of the individual comparison result waveforms also have a value of `1` (`X` is treated as `0`).

Example of a Digital Vector File

This is a basic digital vector file that shows how each Virtuoso UltraSim simulator statement is used.

```
; enable generation of expected output vectors and comparison result waveforms.
output_wf 1
; radix specifies the number of bit of the vector.
radix 2 2 4
; io defines the vector as an input or output vector.
io    i i o
; vname assigns the name to the vector.
vname A[1:0] B[1:0] P[3:0]
; tunit sets the time unit.
tunit ns
; trise specifies the rise time of each input vector.
trise 1
; tfall specifies the fall time of each input vector.
tfall 1
; vih specifies the logic high voltage of each input vector.
vih 2.5
; vil specifies the logic low voltage of each input vector
vil 0.0
; voh specifies the logic high voltage of each output vector
voh 2.0
; vol specifies the logic low voltage of each output vector
vol 0.5
0 0 0 x
200 3 3 x
400 1 2 0
600 2 1 9
800 3 1 2
1000  1  3   2
1200  2  2   3
1400  3  2   3
1600  2  3   4
1800  0  0   6
2000  0  0   7
```

Frequently Asked Questions

Can I replace the bidirectional signal with an input and output vector?

Bidirectional signals can be divided into two columns, one for an input vector and the other for an output vector (the enable signal is no longer needed). The same `vname` and signal name is used for the input and output vectors.

For the input stage, the value of the output vector must be `X` or `x` (output vector check is not performed). For the output stage, the value of the input vectors must be `Z` or `z` (no stimulus for this signal). For example:

```
radix 1 1 1 1
io i o i o
vname DI DO DQ DQ
tunit ns
0 0 1 0 x
100 1 0 1 x
200 0 1 0 x
300 0 0 z 1
400 1 1 z 0
500 0 0 z 1
```

How do I verify the input stimuli?

Use `.probe tran v(*) depth=1` to probe the top-level signals and then check the waveform outputs with WaveScan or SimVision.

Note: The signal names are case sensitive.

Review the log file to check if the signals defined in the digital vector file match those defined in the analog netlist.

- When the signal is defined in the vector file, but not in the analog netlist, the following type of warning message appears:

  ```
  ****WARNING:UFE-5164****: The signal name ddrc_b12 defined in the
  Vcd or Vec file is not defined in the netlist.
  ```

- When the input signal is used in the analog netlist, but does not match the one located in the vector file, check the list of dangling nodes or no DC path to ground in the log file.

How do I verify the vector check?

A `netlist.veclog` file is generated at the location specified by the Virtuoso UltraSim simulator `option-raw` statement if there are any vector checks. A `netlist.vecerr` file is also generated when errors occur during the vector check. Refer to these two files for detailed information about the vector check.

When the signal is defined in the vector file, but not in the analog netlist, the simulator issues the following warning message in the log file:

```
****WARNING:UFE-1253****: Node, p1<31>, not in the netlist.
```

In addition, the simulator generates two waveform files: One contains all the expected output vector waveforms as specified in the digital vector file and the other contains the waveforms from the comparison results.