

LDVSIM man pages

NAME

ldvsim - An mixed-level simulator with timing for digital/MOS circuits and static timing analysis

SYNOPSIS

```
ldvsim [list_of_data_files] [-s comand_file] [-s -]
```

DESCRIPTION

LDVSIM is an event-driven, mixed, logic-level simulator for digital/MOS transistors or gate level circuits. It also has the capability of doing static timing analysis for circuits which use a strict two-phase, non-overlapping clock methodology.

SIMULATION DESCRIPTION

Two simulation models are available:

switch

Each transistor is modeled as a voltage-controlled switch. Useful for determining the functionality of the network.

linear

Each transistor is modeled as a resistor in series with a voltage-controlled switch; each node has a capacitance. Node values and transition times are computed from the resulting RC network.

LDVSIM first processes the files named on the command line. Three types of files may be processed: network files that have been preprocessed by **LPRESIM**, alias files and high level files. The simulator looks at the first few bytes of the file to determine the type.

Note that there is only a single name space for nodes, so references to node "A" in different network files all refer to the same node. While this feature allows one to modularize a large circuit into several network files, care must be taken to ensure that no unwanted node merges happen due to an unfortunate clash in names.

Command files allow the simulation to be controlled. These files are specified with the **-s** option ("**-s** file_name"). To read from standard input, a file name of "-" should be used. These files are processed line by line; when an end-of-file is encountered, processing continues with the next file. After all the command files have been processed, and if an "exit" command has not terminated the simulation run, **LDVSIM** will accept further commands from the user, prompting for each one like so:

```
ldvsim>
```

LDVSIM COMMAND DESCRIPTIONS

Commands have the following simple syntax:

```
cmd arg1 arg2 ... argn <newline>
```

where '**cmd**' specifies the command to be performed and the remaining are arguments to that command. The arguments are separated by spaces (or tabs) and the command is terminated by a <newline>. Iterators may be used to expand a command argument. A '{begin:end}' used in an argument expands the argument so that the argument is repeated with the numbers inclusive of begin and end. The iterator may have a step specified by '{begin:end:step}'. If '**cmd**' is not one of the built-in commands documented below, **LDVSIM** appends ".cmd" to the command name and tries to open that file as a command file (see "@" command). Thus the command "foo" has the same effect as "@ foo.cmd".

Notation:

... indicates zero or more repetitions.

[] enclosed arguments are optional.

node name of node or vector in network.

wnode

name of node or vector in network, can include '*' wildcard which matches any sequence of zero or more characters

| comment...

Lines beginning with vertical bar are treated as comments and ignored -- useful for comments or temporarily disabling certain commands in a command file. All arguments after a '|' are also ignored. Most commands take one or more node names as arguments. Whenever a node name is acceptable in a command line, one can also use the name of a bit vector. In this case, the command will be applied to each node of the vector (the "t" and "d" treat vectors specially, see below).

vector label node...

Define a bit vector named "label" which includes the specified nodes. If you redefine a bit vector, any special attributes of the old vector (e.g., being on the display or trace list) are lost. Note that wild cards are not accepted in the list of node names since you would have no control over the order in which matching nodes would appear in the vector.

set vector val...

Set the values of the nodes in the indicated vector to the given values. "val..." is a string of "1"s, "0"s, "x"s, and "U"s (upper or lower case). It must have length identical to the number of nodes in the vector.

The simulator performs most commands silently. To find out what's happened you can use one of the following commands to examine the state of the network and/or the simulator.

d [wnode]...

Display. Without arguments displays the values all nodes and bit vectors currently on the display list (see "w" command). With arguments, only displays the nodes or bit vectors specified. See also the "display" command if you wish to have the display list printed out automatically at the end of certain simulation commands.

w [-]wnode...

Watch/unwatch one or more nodes. Whenever a "d" command is given, each watched node will displayed like so:

```
node1=0 node2=X ...
```

To remove a node from the watched list, preface its name with a '-'. If "wnode" is the name of a bit vector, the values of the nodes which make up the vector will be displayed as follows:

```
label=010100
```

where the first 0 is the value of first node in the list, the first 1 the value of the second node, etc. The output format may be changed with xprv, dprv, oprv and bprv commands which change a vectors print format to hex, decimal, octal or binary respectively.

"?" and "!" allow the user to go both backwards and forwards through the network in search of that piece causing all the problems.

? wnode...

Prints a synopsis of the named nodes including their current values and the state of all transistors that affect the value of these nodes. This is the most common way of wandering through the network in search of what went wrong... The output from a "? out.1" command looks like

```
out.1=0 (vl=0.3 vh=0.8) (0.100 pf) is computed from:
n-channel phi2=0 out.1=0 in.1=0 [1.0e+04, 1.3e+04, 8.7e+03]
pulled down by (5=1 6=1) [1.0e+04, 1.3e+04, 8.8e+03]
pulled up [4.0e+04, 7.4e+04, 4.0e+04]
```

The first line gives the node's name and current value, its low and high logic thresholds, user-specified low-to-high and high-to-low propagation delays if present, and its capacitance if nonzero. Succeeding lines list the transistor whose sources or drains connect to this node: the transistor type ("pulled down" is an n-channel trans to gnd, "pulled up" is a depletion pullup to vdd), the values of the gate, source, and drain nodes, and the modeling resistances. Simple chains of transistors with the same implant type are collapsed by LPRESIM into a single transistor with a "compound" gate; compound gates appear as a parenthesized list of nodes

(e.g., the pulldown shown above). The three resistance values -- static, dynamic high, dynamic low -- are given in ohms.

Finally, if there are any events pending for a node, they, are listed after the electrical information.

! wnode...

For each node in the argument list, prints a list of transistors controlled by that node. Any node can be made an input -- the simulator will not change an input node's value until it is released. Usually on specific nodes -- inputs to the circuit -- are manipulated using the commands below, but you can fool with a subcircuit by forcing values on internal nodes just as easily.

h wnode...

Force each node on the argument list to be a high input. Overrides previous input commands if necessary.

l wnode...

Like "h" except forces nodes to be a low (0) input.

u wnode...

Like "h" except forces nodes to be a undefined (X) input.

x wnode...

Removes nodes from whatever input list they happen to be on. The next simulation step will determine their correct value of the node from the surrounding circuit. This is the default state of most nodes. Note that this does not force nodes to have an "X" value -- it simply removes them from the input lists.

inputs

prints the high, low, and undefined input lists.

It is possible to define a sequence of values for a node, and then cycle the circuit as many times as necessary to input each value and simulate the network. A similar mechanism is used to define the sequence of values each clock node goes through during a single cycle.

Each value is a list of characters (with no intervening blanks) chosen from the following:

1, h, H logic high (1)

0, l, L logic low (0)

u, U undefined (X)

x, X remove node from input lists

Presumably the length of the character list is the same as the size of the node/vector to which it will be assigned. Blanks (spaces and tabs) are used to separate values in a sequence. The sequence is used one value at a

time, left-to-right. If more values are needed than supplied by the sequence, LDVSIM just restarts the sequence again.

v [node [value...]]

Define a vector of inputs for a node. After each cycle of an "R" command, the node is set to the next value specified in the sequence. With no arguments, clears all input sequences (does not affect clock sequences however). With one argument, "node", clears any input sequences for that node/vector.

clock [node [value...]]

Define a phase of the clock. Each cycle, each node specified by a clock command must run through its respective values. For example,

```
clock phi1 1 0 0 0
clock phi2 0 0 1 0
```

defines a simple 4-phase clock using nodes "phi1" and "phi2". Alternatively one could have issued the following commands:

```
vector clk phi1 phi2
clock clk 10 00 01 00
```

With no arguments, **clock** clears all clock sequences. With one argument, "clock node", clears any clock sequences for that node/vector.

After input values have been established, their effect can be propagated through the network with the following commands. The basic simulated time unit is 0.1ns; all event times are quantized into basic time units. A simulation step continues until stepsize time units have elapsed, and any events scheduled for that interval are processed. It is possible to build circuits which oscillate -- if the period of oscillation is zero time units, the simulation command will never return (*caveat emptor!*).

When using the linear model (see the "model" command) transition times are estimated using an RC time constant calculated from the surrounding circuit. When using the switch model, transitions are scheduled with zero delay. These calculations can be overridden for a node by setting its **tplh** and **tphl** parameters which will then be used to determine the time for a transition.

s [n]

Simulation step. Propogates new values for the inputs through the network, returns when n (default: simstep) time units have passed. If n is specified, it is remembered as the default value for the simstep

parameter. If the display mode is "automatic", the current display list is printed out on the completion of this command (see "display" command).

C [n]

Cycle n times (default: 1) through the clock, as defined by the "clock" command. Each phase of the clock lasts simstep time units. If the display mode is "automatic", the current display list is printed out on the completion of each cycle (see "display" command).

c [n]

Cycle n times (default: 1) through the clock, as defined by the "clock" command. Each phase of the clock lasts simstep time units. If the display mode is "automatic", the current display list is printed out on the completion of this command (see "display" command). p Step the clock through one phase (or simulation step). For example, if the clock is defined as above

```
clock phi1    1 0 0 0
clock phi2    0 0 1 0
```

then "p" will set phi1 to 1 and phi2 to 0, and then propagate the effects for one simulation step. The next time "p" is issued, phi1 and phi2 will both be set to 0, and the effects propagated, and so on. If the "c" command is issued after "p" has been used, the effect will be to step through the next 4 phases from where the "p" command left off.

R [n]

Run the simulator through n cycles (see the "c" command). If n is not present make the run as long as the longest sequence. If display mode is automatic (see "display" command) the display is printed at the end of each cycle. Each "R" command starts over at the beginning of the sequence defined for each node.

create_clock node period [duty time] [offset]

provides an alternative way to drive the simulation. This allows the simulation to run for several cycles and allows easier control over irregular clocks. The command creates a clock starting at the current time with a specified period. The duty time represents the time during the period in which the signal is high (by default, half of the period). The offset represents the time from the current time that the signal goes initially high (by default, 0). The clock can be stopped when a particular signal goes high. The simulation does not have to stop right after the signal goes high, but rather may wait a specified number of extra clock cycles (by default, 0). Clocks specified by this command are not run with 'C', 'c' or 'R'. Instead, the 's' command is used to determine the length of the simulation. This number should be very large. None of the reporting commands are supported with this clocking mechanism.

path wnode...

display critical path(s) for last transition of specified node(s).

printp

print list of all pending events sorted in time. The node associated with each event and the scheduled time is printed.

printx

print a list of all nodes with undefined (X) values. Using the trace command, it is possible to get more detail about what's happening to a particular node. When a node is traced, the simulator reports each change in the node's value:

```
[event #100] node out.1: 0 -> 1 @ 407.6ns
```

The event index is incremented for each event that is processed. The transition is reported as "<old value> -> <new value> @ <report time>". Note that since the time the event is processed may differ from the event's report time, the report time for successive events may not be strictly increasing.

If the debug level is nonzero (see the "debug" command) each calculation of a traced node's value is reported:

```
[event #99, 6->0] 0 -> 1 transition for node
out.1(tau=1.3ns delta=1.4ns)
Definite:   r_up=[2.4e+03,2.9e+03]
r_down=[2.2e+04,1.0e+15]
low_VR=[0.88,2.5e+03] high_VR=[1.00,2.4e+03]
r_min,r_cs,r_max=[2.4e+03,2.6e+03,2.9e+03],rin=2.9e+03
tau_ae=1.3e+03 tau_de=1.1e+03 tau_cs=1.1e+03
tau_x=9.8e-12
enqueueing final value transition
```

In this example, a calculation for node out.1 is reported. The calculation was caused by event 99 in which node 6 went to 0. When using the linear model (as in this example) the report shows

```
<current value> -> <final value>
```

"Definite" indicates that there is at least one path between node out.1 and an input which is free of X-transistors. In contrast, if there is no such path, it will be labeled as "Indefinite." If a circuit does not have any input at all, it will appear as "Pure CS" (pure charge sharing.) In case of pure charge sharing, the <final value> of a node is determined by charges in capacitors. Consequently, an additional event may be scheduled for this node to decay from its <final value> to X in a later time. If the <final

value> differs from the current value, an appropriate event is scheduled -- the last lines of the report indicate which events were added to the event list. "tau" is the intrinsic RC delay, "delta" is when any consequences of the event will be computed; the difference in the two times is how LDVSIM accounts for the shape of the transition waveform on subsequent stages. The middle lines of the report indicate the pull-up, pull-down resistive ranges, left and right bounds of the solution space, minimum, charge sharing, and maximum resistance to the dominant driver, effective resistance controlled by the input, and various kinds of intrinsic delays. These parameters are mainly used to debug the simulator.

debug [level]

set debugging level. Useful for debugging simulator and/or circuit. If debug switch is on, then during simulation step each time a watched node is encountered in some event, that fact is indicated to the user along with some event info. If a node keeps appearing in this printout, chances are that its value is oscillating. Vice versa, if your circuit never settles (ie., it oscillates), you can use the "debug" and "t" commands to find the node(s) that are causing the problem.

t [-]wnode...

set trace flag for node. Enables the various printouts described above. Prefacing the node name with '-' clear its trace flag. If "wnode" is the name of a vector, whenever any node of that vector changes value, the current time and the values of all traced vectors is printed. This feature is useful for watching the relative arrival times of values at nodes in an output vector.

System interface commands:**@ filename**

Take the text inside the specified file as command input. This may be nested.

cmdpath dir...

Set the search path for command files. "dir..." is a list of directories separated by spaces. The default (used for "-filename" arguments) is the current directory only. "cmdpath" with no arguments echos the current search path.

print text...

Simple prints the text on the user's console. Useful for keeping user posted of progress through a long command file.

logfile [filename]

Create a logfile with the specified name, closing current log file if any; if no argument, just close current logfile. All output which appears on

user's console will also be placed in the logfile. Output to the logfile is cleverly formatted so that logfiles themselves can serve as command files.

q or quit

Terminate current input stream. If this is typed at top level, the simulator will exit back to the system; otherwise, input reverts to the previous input stream. (Notice the difference between q and exit).

exit [n]

exit to system, n is the reported status (default: 0).

Simulator parameters are set with the following commands. With no arguments, each of the commands simply prints the current value of the parameter.

decay [n]

set decay parameter, n should be an integer in basic time units (default: 0). If non-zero, tells the number of time units it takes for charge on a node to decay to X. A value of 0 implies no decay at all. You can-not specify this parameters separately for each node, but this turns out not to be a problem. See "report" command.

display [arg]...

set various display modes; possible "arg"s are:

cmdfile commands executed from command file are displayed to user before executing

automatic print out current display list (see "d" command) after completion of "s", "c" or "C" command. Prefacing an "arg" with a "-" turns off that display option.

model [name]

Set simulation model to one of "linear" or "switch" (default: linear). You can change the simulation model at any time -- even with events pending -- as only new calculations are affected.

report [n]

When nonzero, report nodes which are set to X because of charge decay. Setting the parameter to zero disables reporting, but not the decay itself (see "decay" command).

stepsize [n]

Specify duration of simulation step or clock phase. N is in basic time units, e.g., N = 1000 specifies a simulation step of 100.0 ns.

unitdelay [n]

When nonzero, force all transitions to take *n* time units. Setting the parameter to zero disables this feature.

SIGVIEW INTERFACE

sigview [-]wnode [[-]wnode ...]

The sigview command add/delete nodes to/from sigview list. The prefix '-' means delete wnode from the list. Note: The sigview list may be changed during a run. However, a new file must be opened.

sigoption [trigger] [-t start_time] [-l length]

The sigoption command with 'trigger' will set the trigger mode, otherwise, operation will be in normal mode. The '-t' start time option in normal mode is the time to start printing. In trigger mode, this time is the time to begin to try match the pattern. If it is omitted, current time is assumed. The '-l' option specifies the length of time in nanoseconds after data collection has begun. If omitted, then there is no limit. Note: User can switch between the two modes as desired.

sigfile [filename]

This command closes any open signal files and then opens a new one. If filename is present, this is the name of file; otherwise, the file will be called sigview.out and will be sequentially numbered from 0 (ie. "sigview0.out", "sigview1.out", ...). Note: Only one file may be open at a time.

trigger history count signal [signal ...]

Trigger specifies a pattern to begin capturing signal information. The history argument specifies the number of lines the sigview file is to print before the pattern is matched. Count specifies the number of times the pattern must be matched before the trigger is activated. A list of nodes (not vectors) which create a list to be checked. The final argument is a pattern to be matched against this list. The pattern is a regular expression (see **ed(1)**). For example, `^hl.*x.*HL$|^1001.*` will match the where the first two are HIGH, LOW, then zero or more don't care, zero or more X, and last two are HIGH, LOW. The string would also match if first four are HIGH, LOW, LOW, HIGH and the rest are don't care.

MEGAONE INTERFACE

setvector -C node

Set the node for clocking the simulation. This is usually the input clock. This node must be set.

setvector -R node

Set the node for resetting the chip. The outputs of the simulation are set to unknown when this node is HIGH.

setvector -N name

Set the name of the simulation to some name. No name is required, but this may help keep the simulations straight for later reference.

setvector -S sample time

Set the time after the rise of the specified clock that the inputs and outputs should be sampled. Multiple sample times may be set.

setvector -I wnode

Set the vector or nodes for input. Multiple nodes may be specified.

setvector -O wnode

Set the vector or nodes for output. Multiple nodes may be specified.

setvector -T node node

Set a node which may be tristated. The node which causes the tristate is the second argument. Multiple tristate nodes may be specified.

setvector -V filename

Specify the output file.

setvector -Z

Start the collection of vectors.

STATIC TIMING ANALYSIS DESCRIPTION:

If one or more critical paths are specified, the program reads a VPNR network and determines the critical path for worst case timing. The program does not support transistor level analysis. The sequential circuits are removed from the network before processing. Processing will fail if the circuit is not testable (see `audit`). The number of paths to print is determined by the `-C` option which has one argument, representing the number of critical paths to print. The timing information is obtained from the database `db` in the current working directory or the system directory. An alternative database may be defined with the `-D` option.

The gate load is obtained from the database. However, no wiring capacitance is assumed. To provide estimates of wiring capacitance early in the design, the wiring capacitance can be estimated by the fanout of the node using the `-W` option which contains a factor (in picofarads) for multiplying the fanout of each node to determine the approximate wiring capacitance. One to three times the gate capacitance of a nominal (i1s or i1m) inverter or a 2 or 3 input inverting gate (ai2s or ai2m) should be a reasonable value. This value is given by the `loads` attribute in the database. If a design has been extracted and wiring capacitances are known, the `-S` option can be used to provide a better estimate of the wiring capacitance. The `-W` and `-S` options should not be used together.

The information output by the program appears in 7 fields as shown below.

worst rise signal is I151 with delay 9.86

aoi22s	I151	;	0.088	pF	fanout = 1	1.407R	1.262F
oi2s	I142	;	0.164	pF	fanout = 2	1.354R	1.549F
ils	I148	;	0.088	pF	fanout = 1	0.818R	0.650F
ai2s	I149	;	0.048	pF	fanout = 1	0.698R	0.512F
ils	I129	;	0.062	pF	fanout = 1	0.683R	0.549F
oi2s	Long	;	0.133	pF	fanout = 2	1.225R	1.425F
ai2s	I44	;	0.176	pF	fanout = 2	1.279R	0.916F
ils	I41	;	0.232	pF	fanout = 3	1.569R	1.206F
ai2s	I42	;	0.048	pF	fanout = 1	0.979R	0.622F
dsr2s	I3	;	0.146	pF	fanout = 2	0.000R	0.000F

The first field is the gate type. The second field contains the output signal. The fourth and fifth fields contain the amount of driving capacitance of the output signal and the number of gates. The sixth and seventh fields contain the amount the gate affects the critical path for both falling and rising signals in nanoseconds.

More detailed information about the standard cells on the critical path may be obtained with the verbose **-V** option.

LDVSIM SIMULATION COMMAND SUMMARY

```

@ filename
    take commands from command file
? wnode...
    print info about node's source/drain connections
! wnode...
    print info about node's gate connections
| comment...
    comment line
bprv vector...
    set vector print style to binary
c [n]
    simulate for n (default: 1) clock cycle(s) print vectors at end of all
    cycles.
C [n]
    simulate for n (default: 1) clock cycle(s) print vectors at end of each
    cycle
changes begin [end]
    print list of nodes that changed in time interval
clock [node [value...]]
    define sequence of values for clock node
cmdpath dir...
    set the searchpath for command files
create_clock node period [duty time] [offset]
    Create a clock starting at the current time with a specified period. The
    duty time represents the time during the period in which the signal is
    high. The offset from the
d [wnode]...
    print display list or specified node values
debug [level]
    set debug level (default: 0)
decay [n]
    set charge decay time (0 means no charge decay)
display [arg]...
    control what gets displayed when
dset vector value...
    set values of nodes in specified vector using decimal number
dprv vector...
    set vector print style to decimal
exit [status]
    return to system
h wnode...
    make node logic high (1) input
inputs
    print current list of input nodes
l wnode...

```

make node logic low (0) input

logfile [filename]
start/stop log file

model [name]
set simulation model (one of: linear, switch)

opr vector...
set vector print style to octal

oset vector value...
set values of nodes in specified vector using

octal number

p
step the clock through one simulation step (phase)

path wnode...
display critical path for last transition of a node

print comment...
print specified text

printp
print a list of all pending events

printx
print a list of all undefined (X) nodes

q
terminate input from current stream quit terminate input from current stream

R [n]
simulate for n cyc (default:size of longest sequence)

s [n]
simulate for n (default: stepsize) time units

set vector value...
set values of nodes in specified vector in bit format [01xu]

sigfile [filename]
signal file open

sigoption [trigger] [-t start_time] [-l length]
signal option command with trigger argument.

sigview [-]vn [[-]vn ...]
add/delete nodes to/from sigview list.

trigger history count signal [signal ...]
defines trigger for sigview xset vector value...

xset vector value
set values of nodes in specified vector using hex number

xpr vector...
set vector print style to hex

setvector -C clock
Set the clock node of the circuit.

setvector -R reset
Set the reset node of the circuit.

setvector -I input

Set the inputs to the chip.
setvector -O output
Set the outputs to the chip.
setvector -T enable tristate
Set the tristates to the chip, and what enables the tristate.
setvector -S sample time
Set the time (or times) the input, output and tristate vectors are to be sampled.
setvector -N name
Set the name of the vector set.
setvector -Z
Start the capturing of vectors.
stepsize [n]
set simulation step size, in time units
t [-]wnode...
start/stop tracing of specified nodes
u wnode...
make node undefined (X) input
unitdelay [n]
force all transitions to take n time units(0 disables)
V [node [value...]]
define sequence of inputs for a node
vector label node...
define bit vector
w [-]wnode...
add/delete nodes from display list
x wnode...
remove node from input lists

AUTHORS

Jack V. Briner, Jr. (jvb@cs.duke.edu), high level modelling and timing analysis

John L. Ellis, (jle@cs.duke.edu), timing analysis

Mark Horowitz, switch level modelling

C. Y. Chu, switch level modelling

C. J. Terman, framework and unit delay modelling