ValidSIM UPDATE PACKAGE

Manual Number: 900-00223 Rev A

Release 1.0
18 October 1985

Valid Logic Systems, Incoprorated
2820 Orchard Parkway
San Jose, CA 95134
(408)945-9400 Telex 371-9004

LOGIC SIMULATOR SOFTWARE CHANGES INFORMATION

ValidSIM RELEASE 1.0

18 October 1985

1.0 GENERAL DESCRIPTION

Release 1.0 of ValidSIM is the start of a new generation for the Logic Simulator. It contains several significant new features and added capabilities beyond all previous releases. Highlights of Release 1.0 are listed below and are described in more detail both within this document and in the updated Simulator chapter of the reference manual.

- Invocation of the Compiler from within the Simulator to dramatically reduce compilation/linking time.
- Addition of graphics capabilities in the stand-alone Simulator with dramatic improvements to the speed of graphical output.
- Enhanced commands that eliminate the need for OPENing signals before performing other operations.
- An extended menu and greater puck capabilities to facilitate command entry.
- Fuller use of available space for the display of waveforms and BUS-mode values.
- Addition of two new primitives - a uni-directional MOS transistor and an identity buffer.

Existing circuits, Simulator models, command files, etc. do not need to be changed, although users may find it advantageous to do so to take advantage of new capabilities. Incompatibilities between ValidSIM 1.0 and recent releases of the Logic Simulator should be reported as bugs unless otherwise described in this document.

2.0 INVOKING COMPILER FROM SIMULATOR

In previous releases, the Simulator determined the circuit definition by reading in the expansion file and the synonyms file produced by the Compiler. In this release, the Simulator offers a new option that does not require the pre-existence of these two files. The Simulator obtains data about the circuit by invoking the Compiler directly - it is no longer necessary to explicitly invoke the Compiler on a design before the Simulator can be invoked.

It is simple to take advantage of this new feature. Invoking the Compiler from within the Simulator is accomplished either by specifying the root drawing name to the Simulator as a command line argument (as can be done to the Compiler) or by using the new Simulator directive, ROOT_DRAWING (see Directives section below). When the root drawing name is specified, the Compiler is automatically invoked on the indicated design. If a root drawing name is specified both on the command line and in the directives file, the command line argument will override the ROOT_DRAWING directive.

The data obtained by the Simulator is identical to the data that would be obtained if the Compiler were invoked explicitly. The primary difference to the user is the execution time. Since ValidPAGECOMP 1.0 is capable of processing the design page by page, compilation itself can proceed more rapidly. The time required for compilation when invoked from the Simulator is further reduced by avoiding the creation and reading of two potentially large files (the two data files produced by the Compiler for the Simulator). Accordingly, the time required between the creation of a drawing in GED and the start of actual simulation has been reduced significantly.

The Compiler generates error messages if there are any errors during the compilation. If the specified ROOT_DRAWING is not found or if compile errors are detected, the Simulator is exited. Since a design may be divided into separate pages, the error messages may be distributed among many files. These messages are collected and output on the screen when the Compiler has completed, except when the Simulator is invoked under GED. The program COMPERR can also be invoked explicitly to collect all the Compiler error messages. See the Changes document describing these and other features in ValidPAGECOMP 1.0.

3.0 GRAPHICS IN STAND-ALONE SIMULATOR

Graphics capabilities have been added to the stand-alone Simulator. These capabilities are accessed through the use of a new terminal type, GCLUSTER, that offers the full graphics functionality previously available only when running the Simulator under GED. 48 lines of graphical waveforms can now be viewed when running the Simulator in a full-screen window. In addition to the features under GED, faster graphics, an enhanced menu, and greater puck capabilities are also included.

The speed of graphical output has been improved dramatically over the speed of running the Simulator under GED. By moving the graphics capability into the Simulator itself, graphics typically can be output at a speed five times faster than before.

Other areas have benefited as a result of the separation of the Simulator from GED. These areas include a longer menu, an enhanced HARDCOPY command, and additional commands to take advantage of the puck. These features are described in greater detail below.

A window of at least 14 by 86 characters is required to run the new stand-alone Simulator; the user is prevented from running the graphics Simulator in a window that is smaller than this minimum.

The new stand-alone Simulator has been designed to provide all of the capabilities previously provided when running the Simulator under GED. However, the split-screen GED/Simulator mode is still available to the user. In addition, the character waveforms previously utilized in the stand-alone Simulator are still produced when using the terminal type, CLUSTER. Note that the ability to communicate with GED in another window has not yet been implemented in the graphics Simulator, so it is not yet possible to select signals from a GED window.

3.1 Command Menu

A command menu is included in the new graphics Simulator to facilitate command entry. As when running under GED, the Simulator menu appears along the right side of the screen. The number of menu items (boxes) is determined by the size of the current window, and extends from the top of the window to the top of the echo area.

The menu in the graphics Simulator never has less than nine entries (the same number of entries as when running the Simulator under GED). Depending on the size of the window, up to 17 commands can appear in the menu. As in the split-screen Simulator, the basic nine menu entries are OPEN, DEPOSIT, RADIX, RESUME, SCRIPT, HARDCOPY, REDISPLAY, semicolon, and the freebox (the last command not already in the menu). The new additions are SIMULATE, SIM C, WAVEFORMS, ROW, DELTA_TIME (a new command), TRACE, CURSOR, and BUS. These additional commands are listed in the order in which they are added to the menu as space permits.

4.0 GRAPHICAL CHARACTERS ON SS IV

The Simulator waveforms display on SCALD System IV's has also been improved. Although the full graphics capabilities available on the S-32 are not provided on the SS IV's, the traditional character waveforms have been replaced with waveforms utilizing graphical characters. The user does not need to issue any special commands to use these graphical characters, and the characters should result in a more pleasing display.

5.0 EXPANSION OF SIMULATOR DISPLAY AREA

ValidSIM 1.0 was designed to make fuller use of available space in the display area. In any window wider than the minimum width, the Simulator now takes advantage of the previously empty area on the right side of the screen by extending the space available for waveforms and BUS-mode values. This occurs using either the CLUSTER or GCLUSTER terminal type.

In a full-screen window, this expansion of the display area is substantial. When running in WAVEFORMS mode, the width of the waveforms area is 40% larger than when running in a minimum width window. This, of course, results in a corresponding increase in the resolution of the waveforms and/or an increase in the time span which can be displayed without loss of resolution. In combination with the new SIGNAME_CHARS directive (see below), the amount of space available for the display of waveforms has been increased by as much as 77% over previous Simulator releases.

6.0 NEW PRIMITIVES

6.1 UNI PASS TRANSISTOR

The performance of NMOS simulation has been enhanced through the addition of a uni-directional MOS transistor primitive, UNI PASS TRANSISTOR. This primitive not only increases the speed of simulation for MOS circuits, but also improves the readability of drawings where fully bi-directional gates are not required.

Pins and properties of the UNI PASS TRANSISTOR primitive are identical to those of the PASS TRANSISTOR primitive (i.e., the G pin determines if the A and B pins are connected). However, since the transistor described is now uni-directional, the A pin is an input pin rather than an output. Note that although this primitive was available in Release 7.5 of the Simulator, the library model for this device was not included until the latest library release (6.8); users should now be able to take full advantage of this device.

6.2 IDENTITY

The identity buffer primitive, IDENTITY, has been added to the Simulator. This primitive has one input, I, and one output, T, and behaves similarly to the existing primitive, BUF. The IDENTITY primitive propagates the exact signal on the input pin to the output pin, while the BUF primitive converts the Z state to U and soft values to hard values.

7.0 NEW/MODIFIED SIMULATOR DIRECTIVES

The following Simulator directives have been added or modified.

7.1 ROOT_DRAWING Directive

The ROOT_DRAWING directive specifies the name of the drawing to be simulated when the Compiler is invoked directly by the Simulator. The traditional expansion and synonyms files are not needed and are not created when the Compiler is invoked by the Simulator.

The syntax of this directive is identical to the corresponding directive used by the Compiler; the root_drawing name must match the drawing name specified in the Compiler directives file.

ROOT_DRAWING 'drawing name';

When the ROOT_DRAWING directive is used, the COMPILER_OUTPUT and SYNONYM_FILE directives should not be used. Even if specified, any existing expansion and synonyms files are ignored. If the ROOT_DRAWING directive is not included, the traditional expansion and synonyms files must exist (i.e., the drawing must already have been explicitly compiled), and Simulator operation is identical to previous releases.

Note that the Compiler directives file is still required when the Compiler is invoked from within the Simulator.

7.2 SIGNAME_CHARS Directive

The number of characters used for signal names on the left side of the screen in WAVEFORMS mode can now be controlled using the SIGNAME_CHARS directive. The parameter specified with this directive determines the number of character columns to be dedicated to the signal names.

The syntax for this new directive is as follows:

SIGNAME_CHARS { 9 - 24 };

The default value is 24, which was the number of characters used in previous releases of the Simulator. Values outside the legal range will be rounded to the closest legal value.

As the number of characters used for signal names is decreased, the space available for waveforms is correspondingly increased. Of course, with fewer characters available for signal names, a greater number of characters will be truncated when the length of the signal names exceeds the number of characters specified by this directive.

7.3 TERMINAL Directive

A new option, GCLUSTER (graphics cluster), has been added to the TERMINAL directive to specify the new graphics Simulator mode. Accordingly, when running the Simulator on a SCALD workstation, the user may either specify "TERMINAL CLUSTER;" for operation as before, or "TERMINAL GCLUSTER;" to take advantage of the new graphics capabilities in the stand-alone Simulator (see description above).

The new syntax for this directive is as follows:

TERMINAL { VT100 | CLUSTER | ANNARBOR | TTY | 3270 | GCLUSTER };

The operation of the other terminal types is unchanged by the addition of this new terminal type.

8.0 NEW/MODIFIED SIMULATOR COMMANDS

The following commands have been added or modified in this release.

8.1 ASSERTIONS Command

The ASSERTIONS command, first introduced in Release 7.5 of the Simulator, is now fully operational. This command allows timing assertions to be specified interactively while running the Simulator, rather than requiring assertions to be specified as part of the signal name when the drawing is created in GED. The addition of this commands provides the user with an extra degree of flexibility when performing simulations since signal timing assertions are no longer fixed with the signal name and need not be compiled with the drawing. Usage of this command is as follows:

ASSERTIONS <signal name>, <timing data>

The <timing data> parameter is specified using the standard SCALD syntax for timing assertion data (e.g., 0-4). The assertion type (!C or !P), is not specified - the Simulator automatically adds the "!C" property to the timing data.

This command can be invoked on existing clock signals as well as on any other signals in the drawing. Thus, any signal can be assigned timing assertions while in the Simulator, and assertions of existing clock signals can be re-defined. After assigning clock properties, the signal can be OPENed using either its previous or its new (with assertions) name.

8.2 DELTA_TIME Command

The DELTA_TIME command is a new command, used to indicate the time difference between two points on the current waveform display. The points are specified with the puck. Usage of this command is as follows:

DELTA_TIME <point1> <point2>

Note that since this command utilizes the puck for the specification of screen locations, the command only works on terminal types where puck usage is enabled (GED or CCLUSTER). The resultant value is returned in the echo area. The points can be specified anywhere in the waveforms display area within the time frame currently displayed (i.e., valid points are any place where waveforms can be drawn).

8.3 DEPOSIT Command

The DEPOSIT command has been enhanced to include an optional signal name parameter. With this addition, a signal no longer has to be OPENed before a value can be DEPOSITed. The new syntax for this command is as follows

DEPOSIT [<signal name>,] <value> [;]

where <signal name> is an optional parameter that may be specified using the puck. In the absence of the <signal name> parameter, the command works as in previous releases (i.e., the specified <value> is DEPOSITed on the currently OPEN signal). Note that this command does not OPEN the specified signal or change which signal is currently OPEN. If the specified <signal name> has not been OPENed, DEPOSIT causes the value to be placed on the signal, but does not OPEN the signal or cause the signal history to be started.

8.4 DISPLAY Command

The DISPLAY command has been added to allow the user to control the updating of the display area of the screen. Disabling screen updating increases the Simulator's speed when continuous updating of the display area is not required. The usage of this command is as follows:

DISPLAY [ON | OFF] [;]

When updating is disabled, a field on the status line indicates this to the user. The output to the echo area in response to the commands given proceeds as usual. When the display is reenabled, the screen is redrawn as if a REDISPLAY command was issued.

38.1MS

8.5 HARDCOPY Command

The HARDCOPY command is supported in the new graphics Simulator mode (GCLUSTER), and has been enhanced in this mode to provide greater functionality. The command syntax is as follows:

HARDCOPY [{ A - E }] [;]

The optional parameter (A - E) specifies the page size for the output. In the absence of this parameter, hardcopy is output on an 'A' size page.

Several plotter types are supported and are specified through the use of an enhanced SET command (see below). In addition, either local or spooled plots may be selected using the SET command. LOCAL_PLOT and 'versl1' are the defaults when doing HARDCOPY.

With the spooled_plot option, plots are written to a file and can be plotted later on either a remote or local printer. The name of the file is 'hardXX', where XX is the tty number of the current window. The UNIX utility HPR can be used to produce the plot.

Note that usage of the HARDCOPY command has not been enhanced when running under GED (i.e., no page-size parameter is available, and the Simulator SET options have no effect -- users must use the SET commands in GED).

8.6 HISTORY Command

The default value for signal history has been increased from 1000 ns to 10000 ns - the Simulator will now maintain the history of all signals OPENed in WAVEFORMS mode for 10000 ns. Note that the amount of memory required for signal history will increase correspondingly, and users should still use the HISTORY command to adjust the recording period for their own simulations. The syntax and function of the HISTORY command remains unchanged.

8.7 MOVE Command

A MOVE command has been added to allow the user to change the position of a previously OPENed signal on the screen. The syntax of this command is

MOVE <from_point> <to_point>

where <from_point> and <to_point> are specified using the puck in the Simulator window. The signal currently being displayed at <from_point> is removed and redisplayed at <to_point> and replaces any signal that may already be at that location. This command is only functional when the puck is enabled in the Simulator (GED or GCLUSTER).

8.8 PEEK Command

A PEEK command has been added to allow the user to observe the value of a specified signal without requiring that it first be OPENed in the display area. The signal value is simply output in the echo area. Usage of the command is as follows:

PEEK <signal name> [;]

The value of the specified signal is output in the current radix in the echo area. The <signal name> may be specified using the puck. If the CURSOR time differs from the current time, the value of the specified signal is output at both the CURSOR time and the current time. If the specified signal has no history, a message is displayed in the echo area, and the current value is output. Note that if the specified signal has insufficient history, the value output may not be correct.

8.9 REMOVE Command

A signal name parameter has been added to the REMOVE command to eliminate the need to OPEN a signal before it can be REMOVEd from the screen. The enhanced syntax of this command is as follows:

REMOVE [<signal name>] [;]

Note that <signal name> is an optional parameter that may be specified with the puck; if no <signal name> is specified, the currently OPEN signal is REMOVEd. If <signal name> is entered from the keyboard, only a single occurrence of the signal in the current radix is REMOVEd. If this command is selected from the menu (in the freebox), the user is prompted for a signal name instead of simply REMOVing the currently OPENed signal (previous releases simply removed the OPENed signal).

8.10 SET Command

Enhancements have been made to the SET command to facilitate the operation of the HARDCOPY command. The list of available options is as follows:

```
SET { Breakpoint <expression>              |
      Breakpoint # <number>                |
      Enable <signal> WHEN <expression>    |
      Patch <signal> WHEN <expression>     |
      Local_plot                           |
      Spooled_plot                         |
      W11versatec                          |
      W22versatec                          |
      W36versatec                          |
      W42versatec                          |
      Calcomp1043                          |
      Calcomp5744                          |
      B9424                                } [;]
```

Usage of the SET command for BREAKPOINTs, PATCHing, etc. remains unchanged. The new SET options are used in the production of HARDCOPY in the graphics Simulator and are specified identically to the SET command in GED. The same plotter types are supported - the various widths of Versatec plotters; calcomp1043, a pen plotter; calcomp5744, an electrostatic plotter; and the Benson 9424 plotter. In addition, a local/spooled plot option is also available - LOCAL_PLOT queues the output immediately, while SPOOLED_PLOT sends the output to a file for output using the HPR utility. LOCAL_PLOT and the 11" Versatec are the defaults. Note that these new options are only available with the new graphics Simulator - they have no affect on HARDCOPY in the split-screen GED/Simulator.

8.11 SIMULATE Command

The SIM C command is a new menu command in the graphics Simulator. This menu entry allows the user to advance simulated time by one clock period with a single puck point. The original syntax of the SIMULATE command remains unchanged.

8.12 TERMINAL Command

The terminal type GCLUSTER has been added as an option to the TERMINAL command to specify the new graphics Simulator mode (see previous description). The new syntax for this command is:

```
TERMINAL { VT100 | CLUSTER | ANNARBOR | TTY | 3270 | GCLUSTER } [;]
```

When running the Simulator on a SCALD CLUSTER terminal, the user may either specify "TERMINAL CLUSTER;" for operation as in previous Simulator releases or "TERMINAL GCLUSTER;" to take advantage of the new graphics capabilities.

8.13 WAVEFORM Command

The WAVEFORM command has been enhanced in the graphics Simulator to allow start and end times to be specified with the puck. With the WAVEFORM command, a user can zoom in on an interval of interest (the waveforms are extended horizontally on the screen) or shift the display forward in time. The enhanced syntax of this command is as follows:

WAVEFORMS <start time> { <end time> | ; }
or
WAVEFORMS <point1> { <point2> | <";"point}

Note that the second syntax is only available when the puck is enabled in the Simulator (GED or GCLUSTER). The first syntax of this command remains unchanged.

9.0 ANOMALIES CORRECTED

A number of anomalies have been corrected either in ValidSIM 1.0, or in previous releases and not reported until now.

1. Initialization time for the Simulator has been dramatically improved in the 8.0 release of the Simulator. (#61F-0463)

2. The Simulator no longer unnecessarily redraws the entire screen when only a portion of the screen needs to be updated. (#7F-0201, #7R-)

3. Graphical Simulator output is now available in a full-screen window. (#7F-0202)

4. OPENing a signal in WAVEFORMS mode no longer automatically causes the display to be redrawn at the current time.

5. The Simulator no longer requires a signal to be OPENed before it can be REMOVEd. (#7F-0456)

6. The ASSERTIONS command provides the capability to specify different clock assertions for a signal without requiring the drawing be re-compiled each time the assertion is changed. Various problems associated with this command have been corrected. (#7F-0460, #72F-0680, et al)

7. The cursor is now positioned correctly in the echo area, and the echo area is cleared properly after executing various commands that update the signal display area. (#7F-1474)

8. An IDENTITY primitive has been added to the Simulator. (#72F-0053)

9. A problem with the types INTEGER and LONGINT when compiling UCPs has been corrected. (#72F-157)

10. The script that invokes the Simulator has been changed to read a local copy of the file assignment file (simassign) as do the other SCALD programs. (#72F-0425)

11. The UNI PASS TRANSISTOR solves many of the problems users encountered in trying to use the PASS TRANSISTOR primitive. (#72F-0524)

12. A UCP with the name "IBUF" no longer causes Simulator errors to be generated. (#72F-525)

13. The Simulator no longer attempts to continue its initialization after it detects that the expansion file is of the wrong type (or after any other fatal-type errors). (#72F-0606)

14. The COVERAGE command can now be executed again after the WRITE_COVERAGE command is issued. (#27A-0037)

15. The SCRIPT command no longer generates an ASSERTION ERROR (error #12) when the filename given as its parameter does not exist.

ValidSIM 1.0.1 ANOMALIES

The following anomalies have been identified in Release 1.0.1 of ValidSIM:

- Problem: Non-existent signals in the Tabular input file may later cause the Simulator to crash.

  Workaround: The Simulator outputs a message when non-existent signals are discovered. Remove the non-existent signals and their values from the file before using it.

- Problem: Tabular input files cannot be read in the split-screen GED/Simulator.

  Workaround: Use the stand-alone Simulator, either with or without graphics, when using the tabular input feature.

- Problem: There are various problems associated with the PATCH command, particularly when the user tries to re-PATCH a PATCHed signal.

  Workaround: The cycle time to make a GED change and restart the Simulator has been dramatically reduced in ValidSIM 1.0. Keep the use of PATCHing to a minimum.

- Problem: The Simulator may go into an infinite loop while evaluating a circuit containing RES (resistor) or PASS TRANSISTOR primitives.

  Workaround: Insert a buffer or some other gate into the circuit at the input(s) to the RES or PASS TRANSISTOR primitive; values can then be deposited to the input of the buffer rather than directly to the RES or TRANSISTOR. Also, if fully bi-directional gates are not required in the circuit, use the UNI PASS TRANSISTOR primitive rather than the PASS TRANSISTOR.

- Problem: If a PASS TRANSISTOR is connected to a bubbled output, the simulated behavior may not be correct.

  Workaround: If fully bi-directional gates are not required in the circuit, use the UNI PASS TRANSISTOR primitive rather than the PASS TRANSISTOR. The uni-directional model is available in library release 6.8 and later releases.

- Problem: Simulating for excessive intervals (> 30,000,000 ns) with one SIM command can cause the Simulator to crash.

  Workaround: Simulate in more reasonable intervals - break huge intervals into smaller ones (e.g., instead of "SIM 50000000" use "SIM 25000000" twice).

- Problem: If the user types ^C (ctrl-C) to abort the new graphics Simulator, ^Z (ctrl-Z) is disabled when the user returns to UNIX and certain other functions may not work correctly (e.g., the editor 'vi' operates incorrectly).

  Workaround: The ^Z function can be restored by entering the UNIX command "stty susp ^Z".

- Problem: If the user types ^Z (ctrl-Z) or ^C (ctrl-C) to stop the Simulator, the screen will not scroll correctly if the terminal type was set to GCLUSTER and is now set to some other terminal type.

  Workaround: The scrolling region can be reset by entering the UNIX command "echo '<ESC>[r'" (the ESCAPE key, followed by "[r").

- Problem: When using Realchip with Realfast, if the user specifies "clock_edge=rise" or "clock_edge=fall" in the Realchip definition file, the simulated behavior may not be correct.

  Workaround: Change the specification to "clock_edge=both".

- Problem: In Realfast, if a constant signal is connected to a pin of the RESISTOR primitive, the constant value is not propagated to the other pin of the RESISTOR (e.g., a pull up resistor will not pull the pin up to 1).

  Workaround: Insert a buffer between the constant signal and the resistor, where the input is the constant signal and the output drives the resistor.

- Problem: The clock period is not appropriately scaled by the RESOLUTION factor if the RESOLUTION directive appears after the CLOCK_PERIOD directive in the Simulator directives file.

  Workaround: When using the RESOLUTION directive, ensure that it appears before the CLOCK_PERIOD directive in the directives file.

- Problem: Contrary to the description in the manual, simple breakpoints cannot be CLEARed by name.

  Workaround: Use the number of the breakpoint to CLEAR it.

- Problem: Typing ahead ruins the display, particularly when the terminal type is set to GCLUSTER.

  Workaround: Do not type ahead of the Simulator. Wait until the "*" prompt is displayed before entering commands.

- Problem: The Simulator does not generate a screen image properly in the list file when the SNAPSHOT command is executed under GED or when running as GCLUSTER.

  Workaround: Use the HARDCOPY command to generate a screen image. HARDCOPY will generate a graphical image which is not possible in the ASCII list file.