10 HILO & SUSIE Simulator Interface....10-1
10.1 Introduction to HILO....10-1
10.2 CapFast HILO Tools....10-1
10.2.1 HILO Simulations....10-2
10.2.2 HILO Output....10-3
10.3 Preparing the Schematic....10-4
10.3.1 HILO Properties in the Schematic....10-4
10.3.1.1 primitive Property in Sch2hilo....10-4
10.3.1.2 HILO Primitive Symbol Name....10-5
10.3.1.3 HILO Device Name....10-5
10.3.1.4 Circuit Subfile Name....10-5
10.3.1.5 order Property in Sch2hilo....10-5
10.3.2 Required and Optional Properties for HILO....10-6
10.3.2.1 Required....10-6
10.3.2.2 Optional....10-7
10.3.3 porder: the property Defining Property....10-7
10.3.3.1 Porder and Library Symbols....10-8
10.4 Libraries and the cad.rc File....10-8
10.4.1 HILO Primitive Device Symbol Library....10-9
10.4.2 CapFast Standard Symbol Library....10-9
10.4.2.1 The cad.rc File....10-9
10.5 Using the HILO Tools....10-9
10.5.1 *Sch2hilo*....10-9
10.5.1.1 Sch2hilo Options....10-10
10.5.1.2 HILO Input Files....10-11
10.5.1.2.1 Nominal Delay Simulation....10-12
10.5.1.2.2 MIN/MAX Simulation....10-12
10.5.1.2.3 Fault Simulation....10-12
10.5.2 *Flt2sch*....10-12
10.5.2.1 Fault Information....10-13
10.5.2.2 Examples....10-13
10.5.2.3 Flt2sch Options....10-14
10.6 HILO Primitive Device Library....10-14
10.6.1 Circuit Elements....10-14
10.6.2 IEEE Standard Style and Traditional Style Symbols....10-15
10.6.3 Changing Delay Parameters....10-15
10.6.3.1 Group 1 Delay Properties....10-16
10.6.3.2 Group 2 Delay Properties....10-17
10.6.3.3 Group 3 Delay Properties....10-17
10.6.3.4 Group 4 Delay Properties....10-18
10.7 SUSIE....10-19
10.7.1 Preparing a Schematic for Netlisting in SUSIE Format....10-19
10.7.2 *Sch2sus* Options....10-19
10.7.3 Properties recognized by *Sch2sus*....10-20
10.7.4 Signal Referencing in SUSIE....10-20
10.7.5 SUSIE Netlist Format....10-21
10.7.5.1 Netlist syntax....10-21
10.7.5.1.1 Socket definition format....10-21
10.7.5.1.2 Node list format....10-21

Chapter

10 HILO & SUSIE Simulator Interface

This chapter shows you how to interface your CapFast schematics to the digital simulators HILO and SUSIE. The first section describes the CapFast HILO tools; the second section describes the CapFast SUSIE tools.

10.1 Introduction to HILO

CapFast programs operate on your schematic design to create a netlist and to let you perform fault diagnostics. The netlist you create is used as input to the HILO tools. Here's what you'll learn in the HILO section.

- How to prepare your design for HILO simulation
- How to use the CapFast library of HILO primitive symbols and the CapFast standard symbol library with the HILO interface.
- How to use the CapFast HILO tools and the main options available for each tool.

Also included is a reference section that contains a HILO primitive device library of delay parameters that you can use for various devices.

10.2 CapFast HILO Tools

The following is a summary of the CapFast HILO tools. These tools are programs that you run on your schematic design. Here's the function of each program.

- ***HILO Interface*** (*Sch2hilo*) Extracts information from schematics for use with HILO-3 and System HILO digital simulators. *Sch2hilo* accepts design files, (those with a `.dsn` extension), and schematic files (those with a `.sch` extension) as input. It then generates HILO circuit file (a file with a `.cct` extension) as output.
- ***Fault Translator*** (*Flt2sch*) Allows you to view HILO fault diagnostics in *Schedit*. *Flt2sch* reads HILO fault diagnostic files (files with a `.dia` extension) and HILO print files (files with a `.prt` extension). It then generates CapFast command files which have a `.flt` extension.

Also included in the software package are the following:

- **CapFast Library of HILO Primitive Symbols** The CapFast software includes a library of symbols representing HILO primitive devices.

- **CapFast Standard Symbol Library** The CapFast standard symbols contain properties referring to HILO device models in HILO libraries to help you with your HILO simulation.

Below is a diagram showing the flow among the CapFast HILO tools.

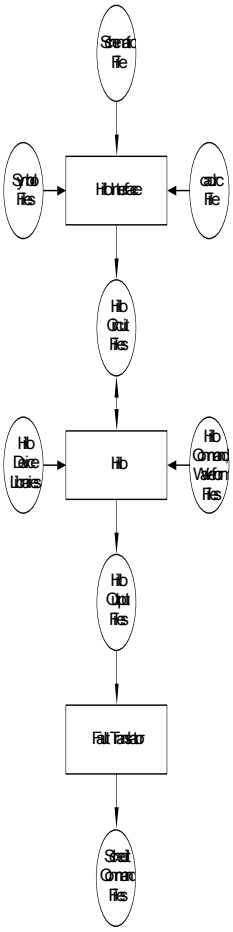

Figure 10-1. HILO Tools.

10.2.1 HILO Simulations

HILO is a gate-level logic simulator that lets you perform three kinds of simulation: fault, nominal delay, and dual delay. With HILO you can also generate automatic tests and interrogate fault lists. You'll find that most simulations will require you to:

- Prepare the schematics of the circuit to be simulated.

- Prepare an input waveform file.
- Prepare a simulation command file.

To simulate a schematic, you must first enter that schematic into HILO's internal database. You can do this in one of two ways. You can either use *Sch2hilo* to produce a `.cct` file by extracting the circuit information from the schematic, or use HILO's internal commands to enter the circuit descriptions textually.

The waveform file, which always has a `.wav` extension, contains a waveform description in the HILO waveform description language. It contains the circuit input stimuli for nominal delay, dual delay or fault simulations. It also contains the output "strobe" times for fault simulations and circuit assertions.

Strobe times specify the times when a hardware tester is to examine the outputs of the circuit to see if a fault is detectable. Circuit assertions cause the HILO program to print a warning message if the specified node does not have the specified value at the specified time.

If the circuit has no inputs, you do not need a waveform file. This is common if the inputs to the circuit consist only of clocks, in which case the inputs may be specified directly in the schematic using the `clock0` and `clock1` HILO primitive elements.

The simulation command file, which always has a `.cmd` extension, specifies the type of simulation: nominal delay, dual delay, or fault. For each type of simulation, it also specifies the following:

A list of nodes to be "captured" (watched) during the simulation

- *For fault simulation* An optional list of faults to be simulated or nodes to be displayed with the print and display commands.
- *For automatic test generation* A skeleton waveform and various other options.

The waveform and simulation command files are read into the HILO simulator as commands and may contain any arbitrary HILO commands except for the `READFILE` or `RF` command.

10.2.2 HILO Output

The two primary outputs from HILO are capture files and automatic test generation waveform files. Capture files (`.cap`) contain the information necessary to graph the nodes specified in the `CAPTURE` list of a `SIMULATE` command. Automatic Test Generation waveform files contain the test waveform generated by the Automatic Test Generator `TESTGEN` command (to create the waveform) followed by the `ATE -TIME` command (to output the waveform to the `.ate` file).

Other outputs available from HILO include fault diagnostics and state change tables. Fault diagnostic information includes lists of untestable faults, pins, etc. You can generate fault diagnostics by invoking the HILO `DIAG` command through the fault simulation command file. This command directs the fault simulator to save the diagnostic information. The fault diagnostic information may then be interrogated interactively or printed to a file using the HILO `DIAG` and/or `PRINT` commands. State change tables, listing the times and values of changes to specified nodes may be printed to the screen as the simulation runs or be stored in a file. You cannot obtain in a single simulation both a state change table and a capture file to be graphed.

In addition to other outputs, HILO creates print files which have a `.prt` extension and log (or display) files which have a `.log` extension.

The print file contains various additional information about a simulation that is too verbose to print to the screen. The log file contains a log of all input and output to the screen during the simulation.

To use HILO on schematics developed with CapFast tools, transfer the `.cct` files created with *Sch2hilo* to the HILO system. You may also have transferred (or created with a text editor) HILO command and waveform files which will now be on the HILO system. HILO-3 places the results in a simulation capture file for graphing. Results from automatic test generation are discussed later in this chapter.

10.3 Preparing the Schematic

This section shows you how to prepare a schematic for extraction by *Sch2hilo*. It also shows you how to use the library of primitive symbols, the standard symbol library, and the `cad.rc` file to obtain the correct operation.

Before reading this section, we suggest that you be familiar with *Schedit* and the ways in which you can create schematics described in the CapFast Tutorial chapter of this manual.

10.3.1 HILO Properties in the Schematic

This section describes the important HILO properties that you can use when you create symbols and schematics. These are the properties that *Sch2hilo* will recognize as it reads the schematic design during the extraction process.

10.3.1.1 `primitive` *Property in* Sch2hilo

The symbol at the bottom level of hierarchy must have the `primitive` property. The syntax is:

```
primitive:NAME
```

or

```
(HILO)primitive:NAME
```

The *NAME* can be a symbol name from the HILO primitive symbol library, the name of one of the devices in a HILO device library, or the name of a circuit subfile which is a circuit that has already been compiled into the HILO database. This name together with the `prefix` and `suffix` property values, if either is present, form the name for the symbol. However, if a `device` property exists, its value will be used for the name instead.

Note: The default qualifiers for *Sch2hilo* are `hilo`, `Hilo`, and `HILO`. See *Chapter 6: Properties*, for more information about properties and their qualifiers.

10.3.1.2 HILO Primitive Symbol Name

The schematic symbols representing primitive HILO circuit elements are in the directories `~p3/library/hilosf` and `~p3/library/hilohf`; details on using the HILO primitive symbol library are given later in this chapter.

10.3.1.3 HILO Device Name

A symbol can refer to an element in the HILO Device Library when that option is part of the HILO package.

10.3.1.4 Circuit Subfile Name

A functional or gate-level description of the circuit (using the HILO hardware description language) is written, then HILO is run to compile the circuit description into a circuit subfile in the HILO database. Each circuit subfile has a name (passed as part of the circuit description), and this name can be used directly as the name for the `primitive` property.

10.3.1.5 `order` Property in Sch2hilo

Every symbol that is a primitive symbol for HILO must have the `order` property. The syntax is:

```
order:PORTNAME,PORTNAME,...
```

The purpose of the `order` property is to match parts on symbols (`.sym` file) to the inputs and outputs of HILO simulation primitives or custom library parts. If you create a

simulation model using HILO's Hardware Description Language (HDL), and a symbol using *Symed*, you must have a way to match their inputs and outputs. Let's look at an example.

Make a symbol for your special part, one that has, say, two inputs, A and B, and one output, C. Name it box1.sym.

Using a text editor, describe what you want this box to do. The textfile might look like this:

```
CCT BOX 1 (out1,in1,in2)
    Register out1;
    Wire in1,in2;
    When in1(0 to 1) THEN
       out1=IN1 AND IN2;
    .
```

Properties must match the symbol ports to the text model inputs and outputs. In the example, C matches OUT1, A matches IN1, and B matches IN2. You would add the following properties to box1.sym :

```
primitive:box1
order:C,A,B
```

The order property (order:C,A,B) matches the top line of the textfile model (CCT BOX1 (out1,in1,in2)). Ports and inputs and outputs are matched by position.

10.3.2 Required and Optional Properties for HILO

Below is a list of those properties you can use for HILO simulation.

10.3.2.1 Required

primitive	See description above.
order	See description above.
porder	Use this property to define new properties for HILO symbols. Often, it's used with delay parameters. Although actually optional, we suggest you specify the porder property to obtain useful results from your HILO simulation.
framedelay(PORT)	This property is used by HILO to perform Automatic Test Generation. It's used for input ports or output ports connected to wires of type WAND or WOR.

10.3.2.2 Optional

`device`	If you give a value to this property, the value takes precedence over the `prefix`, `primitive`, and `suffix` properties as the symbol's name.
`hilotech`	If you give a value to this property, HILO reads the value as the technology of the symbol in the symbol's header. The possible values for this property are: `TTL`, `TTLOC`, `ECL`, `MOS`, `NMOS`, `PMOS`, `CMOS`.
`suffix` `prefix`	If available and you provide values for these properties, the *`PREFIX_VALUE`* + *`PRIMITIVE_VALUE`* + *`SUFFIX_VALUE`* become the symbol's name.
`wiredelay(PORT)`	Use the value of this optional property for wires of type `TRI`. The value is given as two comma-separated integers.
`wiretype`	If you give a value for the `wiretype` property, it specifies a node to be of that type. The possible values are `INPUT`, `INPUT0`, `INPUT1`, `SUPPLY0`, `SUPPLY1`, `TRI`, `TRI0`, `TRI1`, `TRIREG`, `WAND`, `WIRE`, `WOR`, `REGISTER`. In addition, other wiretypes are recognized by System HILO and include: `BIDR`, `INPUTX`, `OUTPUT`, `SUPPLY`, `UNID`, `UNID0`, `UNID1`. The default wiretype is `WIRE`.
`strength`	If you specify this property, then output logic 1 and logic 0 strengths are included for each instance of the symbol. The possible values are a comma-separated pair from the set `WEAK`, `STRONG`, `HIGHIMP` or `Z`. The first parameter is for `HI`; the second is for `LO`. Look in the HILO GHDL reference manual for more details.
`capacitance`	If you give the `capacitance` property a value for a node, such as a port or wire, a capacitor is written to the circuit file to indicate the terminal-to-ground capacitance.

10.3.3 `porder`: the property Defining Property

The `porder` property is a special property that defines other, new properties. You set values in these new properties that are then used after extraction by *Sch2hilo* in your HILO simulation. `porder` property identifies those properties which are a member of the delay parameter list.

10.3.3.1 Porder and Library Symbols

When you use a HILO primitive library symbol, its `porder` value names some of the properties you may set that govern the HILO simulation for the symbol. You do not have to set the `porder` property value for HILO primitive library symbols, it has already been set. For example, the NOT gate primitive symbol has the following `porder` value:

```
porder:drise,dfall,mrise,mfall,termcap
```

This means that the symbol has the five properties listed as the value of this `porder` property, (`drise`,`dfall`,`mrise`,`mfall`,`termcap`) and that you may set these property values for instances of the NOT symbol in your schematic. These properties are delays. See your HILO-3 User's Manual for their use.

Setting an instance value for one of these properties in your schematic is done in the standard manner.

For example, to set mrise for the NOT symbol:

```
mrise:10
```

The default values for the properties named in the `porder` properties are also set in the library symbol definition. For example, the default `drise` and `dfall` values:

```
porder:drise,dfall,mrise,mfall,termcap
drise:10:20
dfall:10:20
```

mean that the properties `drise` and `dfall` (named in the `porder` property) have the minimum default value of 10 and the maximum of 20. The properties `mrise`, `mfall`, and `termcap` are not given default values in this case. *Sch2hilo* will give these three properties the value 0 because they are not set anywhere, although they are named.

Instance (local) values for properties you set in your schematic have precedence over default (global) values.

The values from your properties are extracted by *Sch2hilo* and placed in `.cct` files. The `.cct` file is used as data by HILO for the simulation.

10.4 Libraries and the `cad.rc` File

Schematics to be extracted for HILO must use symbols with appropriate HILO properties. Symbol libraries are part of the CapFast software package but users may also create their own. This section gives the pathnames and brief descriptions of the following symbol libraries:

- HILO Primitive Device Symbol Library (Provided)
- CapFast Standard Symbol Library (Provided)

10.4.1 HILO Primitive Device Symbol Library

The CapFast symbol libraries are placed in sub-directories in `~p3/library`. The HILO primitive device symbols with the appropriate properties for extraction by *Sch2hilo* reside in `~p3/library/hilotf` and `~p3/library/hilosf`. The symbols come in two "shapes": IEEE standard symbol shapes and the traditional style logic shapes. IEEE standard symbols are in `~p3/library/hilosf`; traditional symbols are in `~p3/library/hilotf`.

Within each directory, the file names are the names of the HILO primitives, for example, `nand2`, with the `.sym` extension. *Schedit* options allow you to put either the IEEE style or traditional style libraries, on its data file search path and to save this choice for future editing sessions. The `Get` menu contains the `HILO primitive library` menu, which lists the elements of the HILO primitive device symbol library.

10.4.2 CapFast Standard Symbol Library

The CapFast Standard Symbol Library contains integrated circuit symbols for *Schedit*. Many of these symbols map to the gate-level HILO cell library. Check listings of HILO models from GENRAD to be sure you can simulate your design.

10.4.2.1 The `cad.rc` File

Sch2hilo reads options at invocation from the `cad.rc` file. The library directories you want to use must be included in the *Schedit* path, as well as the *Sch2hilo* path, using the `-p` option for each. This is most conveniently done in the `cad.rc` file. For more information, see *Appendix A: Customizing CapFast*.

10.5 Using the HILO Tools

This section is a tutorial, showing you how to use *Sch2hilo* and *Flt2sch*. It also describes the kinds of data files used, and the main options available for each tool.

10.5.1 *Sch2hilo*

Sch2hilo extracts schematics for logic simulation by HILO.

To run *Sch2hilo*, type at the system prompt:

sch2hilo [*OPTIONS***]** *FILENAME*

Sch2hilo extracts the specified design (`.dsn`) or individual schematic (`.sch`) files hierarchically or file by file. It then places the extracted information in `.cct` files. The extracted schematics are then available for simulation after you transfer them to a system that can run the HILO command. *Sch2hilo* performs schematic extraction by extracting each schematic file into a file with the same base name but with a `.cct` suffix. This ASCII file contains a description of the schematic in the HILO hardware description language.

If you choose, you can also use a text editor to write circuit descriptions directly into the HILO hardware description language. Put these circuit descriptions into files with a `.cct` suffix.

10.5.1.1 Sch2hilo *Options*

You can use the following options with *Sch2hilo*.

`-ds` *DELAYSCALE*	Sets time units for rise and fall times specified in the `drise` and `dfall` properties. Valid *DELAYSCALE*s are `1ps`, `10ps`, `100ps`, `1ns`, `10ns`, `100ns`. Default = `1ns`.
`-dn` *PREFIX*	Changes the prefix used when *Sch2hilo* makes a new name for a node. Default is `N`. *PREFIX* will be converted to uppercase.
`-dc` *PREFIX*	Changes the prefix used when *Sch2hilo* makes a new name for a symbol. Default is `C`. *PREFIX* will be converted to uppercase.
`-h`	Sets hierarchical extraction (default).
`-r`	Sets no hierarchical extraction, causing *Sch2hilo* to extract only from the top level.
`-flat`	Flatten the schematic.
`-NF`	Don't flatten the schematic. (default)
`-0` *NODE_LIST*	The nodes specified are added to the list of logical zero nodes. The *NODE-LIST* may be a comma-separated list of nodes. For example, `-0 GND1,GND2,CGND`

`-1 NODE_LIST`	The nodes specified are added to the list of logical one nodes. The *`NODE_LIST`* may be a comma-separated list of nodes. For example, `-1 VCC1,VCC2,PWR`
`-sh`	Takes into account the more flexible naming rules of System-HILO and extended wiretype set.

10.5.1.2 HILO Input Files

Most simulations will require preparation of three things: the schematics of the circuit to be simulated (`.dsn` or `.sch` file), an input waveform (`.wav`) file, and a simulation command (`.cmd`) file.

Schematics to be simulated must first be in the form of a `.cct` file. You may create this file directly with a text editor, or create it with *Sch2hilo* from your schematic file.

The waveform file (`.wav` suffix) contains a waveform description in the HILO waveform language. This file contains a header, the circuit input stimuli and response called "output declarations", and the waveform clauses. The waveform clauses specify changes in output values with respect to time (e.g `100 CLR=0`) as well as changes with respect to test programs and other special commands for example, `340 STROBE A`.

If the circuit has no inputs, the waveform file is not needed. This is common if the inputs to the circuit consist only of clocks, in which case the inputs may be specified directly in the schematic using the `clock0` and `clock1` HILO primitive symbols. Here is a sample waveform file, for a JKMS shift register circuit.

```
WAVEFORM JKMS
WAVSTIMULUS CLR=1 CLK=0;
RESPONSE C1,C2,C3,C4=X;
100 CLR=0;
200 CLR=1;
200 CLK= BY 300 TO 10K CHANGE0(150,300);
340 STROBE A;
640 STROBE B;
940 STROBE C;
1240 STROBE D;
10K FINISH.
```

Among the things specified in the simulation command file (`.cmd`) are:

- The type of simulation which can be nominal delay, dual delay, or fault.
- A list of nodes to be "captured" (watched) during the simulation, or a list of nodes to be displayed with the display or print commands such as `displaychange`, `displaystrobe`, `displaystable`.

- A list of nodes for fault simulation
- An optional list of faults to be simulated
- A skeleton waveform for automatic test generation, although this will normally not be in the `.cmd` file

Here are some examples of command files for the JKMS example.

10.5.1.2.1 Nominal Delay Simulation

```
SIMULATE JKMS JKMSWAV
CAPTURE(TIME,,CLR,CLK,,C1,C2,C3,C4,,FF1_NQ,FF2_NQ,FF3_NQ,FF4_
    NQ)
5K FINISH.
```

10.5.1.2.2 MIN/MAX Simulation

```
SIMULATE JKMS JKMSWAV
MINMAX
CAPTURE(TIME,,CLR,CLK,,C1,C2,C3,C4,,FF1_NQ,FF2_NQ,FF3_NQ,FF4_
    NQ)
5K FINISH.
```

10.5.1.2.3 Fault Simulation

```
SIMULATE JKMS JKMSEXWAV
Faultsim
DIAG PINS JKMSDIAG
CAPTURE(TIME,,CLR,CLK,,C1,C2,C3,C4,,FF1_NQ,FF2_NQ,FF3_NQ,FF4_
    NQ)
5K FINISH.
```

To display the diagnostic results, another file would be required:

```
DIAG JKMSDIAG.
```

10.5.2 *Flt2sch*

A utility program, *Flt2sch*, allows you to view HILO fault diagnostics in *Schedit*.

To run *Flt2sch*, type at the system prompt:

flt2sch [*OPTIONS***]** *FILENAME* **...**

10.5.2.1 Fault Information

Fault information is created by the HILO simulator using the `Faultsim` command to run the HILO fault simulator and the `DIAG` command to capture the fault information. The resultant fault diagnostic file contains a list of faults in the design that were not detected by the input waveform.

Flt2sch reads a fault diagnostic file (`.dia`) created by the HILO fault simulator and produces *Schedit* command files (with a `.flt` extension), one for each schematic file, that select all nodes with undetected faults and all symbols with undetected faults on any pin. The `.flt` files are created in the current directory.

Some designs contain undetectable faults that cannot be detected by any possible input waveform. HILO lists these faults in the HILO print (`.prt`) file. *Flt2sch* may be used to view these undetectable faults by first editing the `.prt` file to remove all lines but the list of undetectable faults, then invoking *Flt2sch* on the `.prt` file.

10.5.2.2 Examples

To run a HILO fault simulation of the circuit `jkms` with the waveform `jkmswav`, and print the diagnostic information to the file `jkms.dia`, the HILO command (`.cmd`) file might contain:

```
simulate jkms jkmswav
parallel
diag pins jkmsdiag.
print jkmsdiag to jkms.diag
```

The `.diag` file is created.

Note that this `.diag` file becomes a `.dia` file on PCs.

Run *Flt2sch* with the command:

```
flt2sch jkms
```

To examine the faults in `jkms.flt` using *Schedit*, start the *Schedit* and load `jkms`. Then type **`F`** (File) **`X`** (Execute...) **`jkms.flt`**.

To run *Flt2sch* on the undetectable faults in a design, first edit out all but the undetectable faults from the list in the `.prt` file with the text editor of your choice, and then run *Flt2sch* as in the following example:

```
flt2sch jkms.prt
```

10.5.2.3 Flt2sch *Options*

Flt2sch has the following option:

`-a`	Append undetected faults to existing `.flt` files, if any, instead of overwriting them.

10.6 HILO Primitive Device Library

CapFast symbol libraries are placed in subdirectories in `~p3/library`. The HILO primitive device symbols with the appropriate properties for extraction by *Sch2hilo* reside in `~p3/library/hilotf` and `~p3/library/hilosf`.

This section describes the symbols in the HILO Primitive Device library, including the properties that you can use to specify the delay parameters for these devices.

10.6.1 Circuit Elements

Primitive logic elements:

```
and2, and3, and4

or2, or3, or4

nand2, nand3, nand4

nor2, nor3, nor4

buf, not
```

Tristate drivers with inverting/non-inverting control inputs:

```
bufif0, bufif1

notif0, notif1
```

Transmission gates:

```
tranif0, tranif1

moveif0,moveif1
```

Balanced two-input line receiver:

```
balr
```

Multi-output elements for I^2L:

```
buf2, not2
```

Clock-generating oscillating elements:

```
clock0, clock1:
```

Capacitance element:

```
capacitor
```

10.6.2 IEEE Standard Style and Traditional Style Symbols

The symbols come in two shapes: IEEE standard symbol shapes and the traditional style logic shapes. IEEE standard symbols are in `~p3/library/hilosf`; traditional style symbols are in `~p3/library/hilotf`. Within each directory, the file names are the names of the HILO primitives with the `.sym` extension, e.g., `nand2.sym`.

Note: A default `cad.rc` file is supplied with the system. This default file contains a line referring to `hilotf`. Refer to *Appendix A: Customizing CapFast* for more information about `cad.rc`.

10.6.3 Changing Delay Parameters

The library symbols for the HILO primitive elements contain predefined property lists that include delay parameters as properties. The property names are similar to those used in the GENRAD HILO-3 User's Manual. The `default` property list for each symbol also contains a `porder` property that tells *Sch2hilo* the names of some additional properties you may set, such as the five delay properties. Because of the `porder` property, you can change the delay values of primitives by using the names listed in the `porder` property.

In general, the library symbols have property lines with values for the nominal delay parameters (`drise`, `dfall`). Within *Schedit*, use the `Properties→Edit...` command to add the new value for these parameters.

However, (`mrise`, `mfall`) and terminal capacitance (`termcap`) parameters are listed in the `porder` line, but are not given default values. For these parameters, you must use the `Properties→Edit...` command and add properties with the parameter name and desired delay value.

Note: Symbols you define may require the `porder` property, as described earlier in this chapter.

Here is a summary of the delay parameter names for the groups of elements, and the units used for the parameter values. Default values are given when they are present in the

symbol definition; use Properties→Edit... to change these. If no default value is given in the description, you must use Properties→Edit... and add the property name and value.

The GENRAD HILO Manual divides the elements into four groups, based on the kinds of delay parameters used. The discussion below retains the four groups, although the element names within each group differ from those shown in the GENRAD HILO Manual. Chapter 6 of the GENRAD HILO Manual also contains pertinent information.

10.6.3.1 Group 1 Delay Properties

Symbols corresponding to HILO Group 1 elements are:

and2	and3	and4
or2	or3	or4
nand2	nand3	nand4
nor2	nor3	nor4
not	buf	
bufif0	bufif1	
notif0	notif1	
tranif0	tranif1	
moveif0	moveif1	
balr		

Delay parameter properties for these symbols are:

`drise` — Delay for a 0-to-1 signal transition. May be three numbers separated by colons (*MIN*:*TYP*:*MAX*), two numbers separated by colons (*MIN*:*MAX*), or a single number (*MIN*, *TYP*, and *MAX* are the same). The time units for the delay are set by the `-ds` option to *Sch2hilo*. The default is 1 ns. The units should be the same for all symbols in a given circuit. If there are several units, HILO may use the smallest unit in any of the symbols for the entire circuit. Default value: 10:20.

`dfall` — Delay for a 1-to-0 signal transition. May be three numbers separated by colons (*MIN*:*TYP*:*MAX*), two numbers separated by colons (*MIN*:*MAX*), or a single number (*MIN*, *TYP*, and *MAX* are the same). The time units for the delay are set by the `-ds`

	option to *Sch2hilo*. The default is 1 ns. The units should be the same for all symbols in a given circuit. If there are several units, HILO may use the smallest unit in any of the symbols for the entire circuit. Default value: 10:20.
`mrise`	Marginal 0-to-1 transition delay in seconds per farad.
`mfall`	Marginal 1-to-0 transition delay in seconds per farad.
`termcap`	Terminal capacitance in femtofarads (10E-15 farads).

10.6.3.2 Group 2 Delay Properties

Symbols corresponding to HILO group 2 elements are:

```
not2        buf2
```

Delay parameter property names for these symbols are:

```
dr1         df1

dr2         df2

mr1         mf1

mr2         mf2

termcap
```

The properties `dr1` and `dr2` are delays for 0-to-1 transitions. See `drise` discussed earlier for details. Default value is 10:20. The properties `df1` and `df2` are delays for 1-to-0 transitions. See `dfall` discussed earlier for details. Default value is 10:20.

The properties `mr1` and `mr2` are marginal 0-to-1 transition delays in seconds per farad. The properties `mf1` and `mf2` are marginal 1-to-0 transition delays in seconds per farad. The property termcap is the terminal capacitance in femtofarads (10-15 farads).

10.6.3.3 Group 3 Delay Properties

Symbols corresponding to HILO group 3 devices are:

```
clock0      clock1
```

These two symbols have one delay property, `delay`. This property takes a list of three numbers as its value; for example:

```
delay:50,50,50
```

The first number is the absolute time of the first output change; the second number is the interval to the next output change; the third number is the interval from the second to the third output change. Default value for both symbols is `50,50,50`.

10.6.3.4 Group 4 Delay Properties

The symbol corresponding to HILO group 4 is:

```
capacitor
```

This symbol has a single delay property, `termcap` , the terminal capacitance in femtofarads (10 E-15 farads). The value for `termcap` can be one, two, or three numbers separated by colons, signifying the min, typ, and max values.

10.7 SUSIE

The CapFast Schematic package is designed to work with the digital simulator SUSIE. This section describes how to create a netlist for SUSIE after you've created a design with *Schedit*. The *Schedit* menus included in *Schedit* allow you to access SUSIE parts, using digital component templates. Selecting a part from the menu, for example, a 74ALS153 calls up a template and automatically adds the properties needed for netlisting.

Note: In order to get a part that is suitable for SUSIE simulation, it must be taken directly from the `SUSIE` menu.

10.7.1 Preparing a Schematic for Netlisting in SUSIE Format

To create a circuit for SUSIE, take the following steps:

1. Create the circuit design using *Schedit*. Use the `SUSIE Library` for your `Part` menu. During simulation, SUSIE refers to each part by its name. The names that SUSIE uses are passed to it through the netlist generated by *SUSIE Interface* (*Sch2sus*). By default, the name in a netlist for a part is that part's reference designator. If there is no reference designator on the part, then *Sch2sus* uses the part's instance name. Reference designators may be assigned by you, by the CapFast packaging utility *Pkgr*, or by *Sch2sus*.

   For example, if you have a 74LS00 NAND gate with reference designator of `U3`, and an instance name of `00#13`, then *Sch2sus* refers to that NAND gate as `U3`. But if that same NAND gate has no reference designator, then *Sch2sus* refers to the gate as `00#13`.

2. To start *Sch2sus*, type at the operating system prompt:

   sch2sus [*OPTIONS***]** *FILENAME*

   where *FILENAME* must either be a design (a file with a `.dsn` extension) or a schematic (a file with a `.sch` extension) file.

10.7.2 *Sch2sus* Options

`-Q` *QUALIFIER-LIST* Specifies one or more qualifiers which are in effect for the current run. The program matches qualifiers against the properties in the schematic files and only those properties which have matching qualifiers, or are not qualified, are recognized by the program. The default qualifier, `ASC`, is

always in effect. See *Chapter 6: Properties* for more information.

`-P` Does an elementary packaging of symbols by giving them reference designators. Each part is put in a separate package. A list of symbols and reference designators assigned them is written to an `sch2sus.log` logfile.

10.7.3 Properties recognized by *Sch2sus*

Below is a list of properties recognized by *Sch2sus*.

`primitive:`*`XXXX`* where *`XXXX`* is the SUSIE timing model name.

`ref:`*`REF_DES`* where *`REF_DES`* is the reference designator assigned by you or *Pkgr*.

`pin(`*`PORT_NAME`*`):`*`PIN_NUMBER`* where *`PIN_NUMBER`* is the physical pin number assigned to the symbol port *`PORT_NAME`*. The *`PIN_NUMBER`* is assigned by you or *Pkgr*.

`pinnumber (`*`PORT_NAME`*`):`*`X1,X2,X3`* where *`X1,X2,X3`* are the physical pin numbers that may be associated with the symbol *`PORT_NAME`* via the `pin` property.

10.7.4 Signal Referencing in SUSIE

Signals may be referenced in SUSIE by connector names or wire names.

If a connector is present at a node, the instance name for that particular connector will be used to reference the node in SUSIE. A node will always be referenced by the connector name if it is present.

If no connector is present, the node will be referenced by the wire name. Both default and user defined wire names will be used to reference a node. These wire names must not contain any of the following characters:

```
& / , ;  space  tab
```

Note: SUSIE is not case-sensitive and *Schedit* is. Therefore, use names which have unique spellings and do not rely on case differences to make distinctions. Example schematics are included in the `~p3/wcs/examples/susie` directory, which are the same ones used in the SUSIE tutorial. The schematics included are `test1.sch`, `bus.sch`, and `mpu68.sch`.

10.7.5 SUSIE Netlist Format

`/`	precedes chip pin and connector number, e.g. `a2/3,/bb`
`,`	mandatory separator of node elements, e.g. `a1/1,/bb9`
`&`	marks the continuation of a node on the next line
`space`	optional node element separator (ignored)
`tab`	optional node element separator (ignored)
`;`	comment mark, starting comment field.

Note: A comment mark, "`;`" cannot be used at the top of a file to start a netlist.

Caution: In SUSIE, only one I/O name may be used to reference a node. For example:

```
/in1
```

would be correct while

```
/in1,/node1
```

would be incorrect.

10.7.5.1 Netlist syntax

Running *Sch2sus* creates a file in the format :

FILENAME`.net`

where *FILENAME* is the schematic or design file argument to *Sch2sus*. The SUSIE netlist consists of two sections: the socket definition and the node lists.

10.7.5.1.1 Socket definition format

The format of the socket definition is as follows:

`sockets` *SYMBOL_ID* `=` *PART_TYPE*

where *SYMBOL_ID* is either the instance name of the symbol or the reference designator of the symbol. *PART_TYPE* is the name of the part used. The *PART_TYPE* is usually the primitive of a part.

10.7.5.1.2 Node list format

The node list format is as follows:

```
/NODENAME,SYMBOL_ID/PIN_NUMBER &
SYMBOL_ID/PIN_NUMBER
```

where *NODENAME* is the name of a wire or connector in the schematic format, *SYMBOL_ID* is the `ref` property or the instance name of a particular device, *PIN_NUMBER* is the physical pin associated with the *SYMBOL_ID* which is connected to a node.

The following procedure explains how to create a SUSIE netlist.

1. Netlist must start with socket definitions. No titles, headers, etc.

2. Each chip must be listed separately. For example:

```
sockets ff1 = 74LS74
sockets pld1 = 10H8
sockets xor1 = 74LS86
sockets nand1 = 74LS00
;the order is arbitrary
```

3. Each node must start on a new line and end with a newline. To continue a node, place an `&` character followed by newline and continue on the next line. There is no limit on the number of lines for a single node. For example:

```
/input,ff1/2,xor1/2
/n#24,ff1/3,xor1/3&
nand1/4,xor1/3
;this is a comment
/output,ff1/3,xor1/2,pld1/3 ;this is a comment also
```

4. Node connections can be listed in any order. However, nodes cannot be broken into pieces and listed separately.

5. The signal names **CANNOT** contain any spaces. Spaces are used as signal separators.