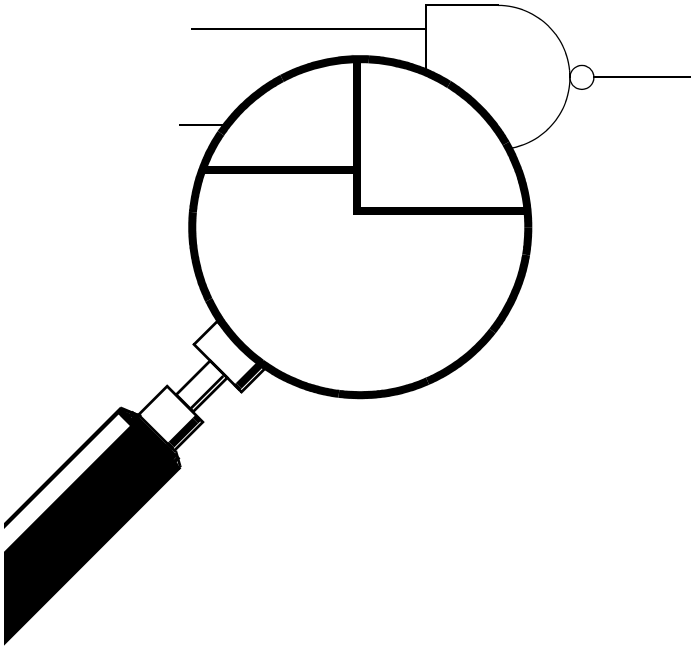
Sp2log User's Guide



Genashor Corp.
17 Dogwood Drive
Neshanic Station, New Jersey 08844-2517
(908) 369-8554
Fax: (908) 369-8544
Email: support@genashor.com

Genashor Corp
17 Dogwood Drive
Hillsborough, NJ 08844-2517
Telephone: (908) 369-8554

Table Of Contents

Documentation Conventions

`Courier` typeface is used to illustrate command syntax, command names, and signal names.

`Courier bold` typeface illustrates user input, the names of buttons in dialog boxes, and the names of bar menu buttons.

Italic type is used to indicated keyword-fields in commands and file names.

<key> (angle brackets surrounding this typeface) identifies a function key or a special keyboard key.

[] right braces identify optional items in commands and description statements.

Sp2log – Spice to Logic Compiler

1 Overview

Typically, simulation models are generated from abstract data sheet diagrams and specifications. While the resulting model captures the *intended* logical function of the design, it often fails to capture problems associated with the *actual* design (e.g., wiring errors, logic cell design errors, timing problems, etc., in the physical implementation). Also, the model may not provide an accurate basis for fault selection and test generation, since it is not based on the physical circuit. Thus, the actual physical design should be the starting point for generating *accurate* functional and fault models.

Sp2log translates a Spice level circuit description (usually obtained by extraction from the layout) into a gate and switch representation. Switches are only used in this representation when a gate (including tristate functions) cannot be generated.

Pattern recognition (transistor topology) is not used for gate reduction. Instead, boolean equations are generated representing the pullup and pulldown paths. Boolean algebra and conflict resolution routines determine where gates reside. This approach eliminates the complicated rules associated with topological matching and ultimately produces a better mapping of functionality.

Even in a "purely digital" design, there are subcircuits whose behavior is sufficiently non-digital to confuse a switch-level logic simulator. Typically, these subcircuits reside in the pad circuitry; TTL level shifters and Schmitt trigger inputs are examples. Analog functions such as memory sense amplifiers and voltage comparators are examples of other "troublesome" subcircuits. Sp2log recognizes analog transistor biasing, and transforms these subcircuits into a digital equivalent for the model.

Sp2log utilizes a flexible method for approximating gate delays. A postprocessor can "tweak" the delays to make them more accurate, if necessary. Sp2log provides a good estimation of the distribution of the delays throughout the circuit. This is very important for modeling single-input pulse handling characteristics and multi-input setup and hold characteristics. Distributed delays are necessary for properly modeling subcircuits such as inverter chains, one-shots, and blocks with many levels of logic between their input and output signals.

Sp2log has been tested on more than 150 digital cells in an ASIC library, from simple gates to the Advancell 2900 bit slice family. It successfully reduced all Spice descriptions in this library to near 100% gate level equivalents. Sp2log has also been used on a number of large custom designs with excellent results. The small amount of bilateral switches that remained were due to the bidirectional busses of RAM cells.

When Sp2log is used in conjunction with the delay characterization program, Gendel, the resulting models are extremely accurate in functionality and timing.

2 Description of Sp2log

Sp2log translates a circuit description in Spice format consisting of MOS transistors and capacitances to a SIMIC gate level description. Though manually-generated Spice descriptions can be used, files generated from the Cadence Dracula™ program are preferred. Gate delays are assigned by a first order delay approximation algorithm. In the event that the analysis cannot resolve a transistor grouping into a unilateral function, Sp2log generates a logical switch level representation of the function.

2.1 Invoking Sp2log

The command syntax for running Sp2log is:

`sp2log` *spicefile* [*outfile*] [*options*]

where:

spicefile is the name of the file containing the Spice description.

outfile is the name of the gate level description output file. If this file name is omitted, the description is directed to the terminal.

options are any of the following switches:

`-n` *delname* specifies the format for the symbolic delay name. It is used when using Sp2log with Gendel or manually generated delay tables. For further information, see the user file *symbol* option.

`-p` *pinfile* specifies that *pinfile* is the name of the pin description file generated by the Dracula™ program.

`-s` Specifies to sort the circuit's pin names alphanumerically, rather than in definition order.

`-t` *typename* assigns *typename* as the name of the circuit. If no name is assigned by this switch or by the Spice `.SUBCKT` statement, the circuit name defaults to `CIRCUIT`.

`-u` *userfile* specifies that the file named *userfile* contains technology information for computing delays and resolving conflicts. This optional file can be used to selectively override parameters in the user-created default file, *sp2log.dft*, which, if it exists, is always read by Sp2log.

`-v` turns on verbose mode. Sp2log will display the Version number and key processing points during the program's execution.

3 The Input Files

Note: All input files are case insensitive.

3.1 The Spice File

The Spice file can contain `M` and `C` statements for transistors and capacitors, respectively. Spice

in-line comments beginning with `$` can be used, but the Dracula™ extensions for X and Y positions are supported; these component locations are echoed in the digital description as comments. The `.SUBCKT` statement is processed, but all other statements beginning with a period are ignored. Comment statements (statements prefixed with an asterisk) can be used in the file, but the following "embedded comment" statements are recognized and processed:

- `.EQUI` statements generated by Dracula™ contain node aliases. Sp2log replaces the SPICE numeric node names with these equivalent names when generating the gate level description. These statements are also used by Sp2log to map the primary pins from the pin file to the proper nodes. For example:

  ```
  *.EQUI A=341 B=75 C=98 VDD=23 VSS=0
  ```

- `.GLOBAL` statement generated by DRACULA™ contains the global nodes for VDD and VSS respectively. For example:

  ```
  *.GLOBAL 23 0
  ```

- Double asterisk comments of the form:

  `**` *signal*`:`*type node*

  where:

  signal is the alias name for the node,

  type is either `i`, `o`, `b`, `v`, or `g` for primary input, primary output, primary bus, VDD, or VSS respectively. Any other character string is considered a designation for an internal node.

  node is the number of the node to be assign this alias.

  For example:

  ```
  ** A:I 341
  ```

3.2 The Pin File

The pin file contains primary signal information in the form:

signal`:`*type* [*comment*]

where:

signal is the alias name for the node,

type is `i`, `o`, `b` for primary input, primary output or primary bus, respectively. Any other characters are ignored.

comment is any text until the end of the line.

3.3 The User File

Sp2log has built-in default values for technology parameters, as described below. At start-up, it checks whether the file *sp2log.dft* exists and, if so, reads the default parameter values from this file. Additionally, the user file passed to Sp2log in the command line can override any built-in defaults or value contained in *sp2log.dft*. In both files, parameter values are specified, one per line, as follows:

parameter = *value*

where *value* is a value to assign to the parameter and *parameter* is one of the following:

pcap This is a floating point value that specifies the amount of capacitance per unit area for the gate of a P device. The default is value is 0.

ncap This is a floating point value that specifies the amount of capacitance per unit area for the gate of an N device. The default value is 0.

mratio this floating point value specifies the **n**/**p** mobility ratio used by the conflict resolution algorithms. The default value is 1.

wfactor this is a floating point multiplicative factor used to scale strength values for source follower and other topologies that create "weak" ON transistors and is used for conflict evaluations. The default value is 1000.

wdfactor this is a floating point multiplicative factor used to scale "W/L" for source follower and other topologies that create "weak" ON transistors and is used for delay calculations. The default value is 4.

sdcap This floating point parameter specifies the amount of capacitance to add for each source or drain connection. Its use is similar to the `gcap` parameter. The default value is 0.

symbol The value is a string in double quotes that specifies the format for creating the symbolic name for delays. This should be used in conjunction with Gendel or a manually created delay table for Simic. The string can contain a number of two character variables starting with a percent sign (%). These variables will be replaced by the value specified in the table below. In addition, an integer can be placed between the percent sign and the variable character. This value will be multiplied with the value for the final result. For example if %n has a value of 0.2, then %100n will have a value of 20. Only the rounded integer results will be used. So if the value of %n is 0.24, %n will be replaced by 0, and %10n will be replaced by 2.

Table 1: Variables for Delay Symbol

variable	Description
%%	Replaced with the character '%'
%d	Replaced with the average W/L value for the pulldown function.
%n	Replaced with the maximum number of transistors in series for the pulldown function.
%p	Replaced with the maximum number of transistors in series for the pullup function.
%s	Replaced with the node's (signal) name.
%t	Replaced with the type's name.
%u	Replaced with the average W/L value for the pulldown function.

If *symbol* is defined in the user file or the command line, then it will override any local delay

parameters defined below. The values are to compute the 7 parameters for output delay and ramp values for a Simic delay rise and fall curves in the format: (maxcap [a,b,c,d,e,f]). The parameters that end in 'r' are for the rise curve, and those that end in 'f' are for the fall curve. Each one is a quoted equation string, supporting the four math operations addition, subtraction, multiplication and division (+, -, *, /). The default values are 0. The following variables are supported in the equation:

Table 2: List of Variables for Delay coefficients

variable	Description
%d	Replaced with the average W/L ratio for the pulldown function.
%i	Replace with the maximum internal capacitance value.
%l	Replaced with the total capacitance on the node.
%n	Replaced with the maximum number of transistors in series for the pulldown function.
%p	Replaced with the maximum number of transistors in series for the pullup function.
%u	Replaced with the average W/L ratio for the pullup function.

Thee Simic delay equation for the *start* of the output ramp is:

```
Delay = a + b(cap) + c(ramp)
```

and the Simic equation for output ramp time is:

```
Ramp = d + e(cap) + f(ramp)
```

where "ramp" is the input ramp to the element, and "cap" is the capacitance on the output node. Specifically, the eight parameters are:

maxcapr The 'maxcap' coefficient for the rise equations.
ar The 'a' coefficient for the rise delay equation.
br The 'b' coefficient for the rise delay equation.
cr The 'c' coefficient for the rise delay equation.
dr The 'd' coefficient for the rise ramp equation.
er The 'e' coefficient for the rise ramp equation.
fr The 'f' coefficient for the rise ramp equation.
maxcapf The 'maxcap' coefficient for the fall equations
af The 'a' coefficient for the fall delay equation.
bf The 'b' coefficient for the fall delay equation.
cf The 'c' coefficient for the fall delay equation.
df The 'f' coefficient for the fall ramp equation.
ef The 'e' coefficient for the fall ramp equation.
ff The 'f' coefficient for the fall ramp equation.

For example, a simple RC equation might be described as:

```
ar = "1.45"
br = "8.75e12 * %n * %u"
af = "2.45"
bf = "5e12 * %p * %d"
```

4 Error Messages

4.1 FATAL Messages:

`Cannot open file.`
File not found or protections set incorrectly.

`Cannot find symbol.`
When reading the pin file, the specified pin was not in the symbol table.

`Invalid or missing port values.`
The ports for a capacitor or transistor were not integers, or not enough ports found.

`Duplicate part number.`
More than one transistor has the same "M" number specified.

`Invalid attribute syntax.`
The attributes for a transistor are not specified in the correct *attr* **=** *value* syntax.

`Missing 'W' parameter.`
Width of transistor is missing.

`Missing 'L' parameter.`
Length of transistor is missing.

`Invalid model type.`
Could not decipher the model name.

`Connection invalid.`
Capacitor was connected to non-existent node.

`Missing or invalid value.`
Capacitor's value was not a number or was missing.

`No transistors found.`
Circuit did not contain any transistors.

`Invalid keyword.`
: User parameter was not valid.

4.2 WARNING Messages:

`Conflicting drivers removed.`
: SP2LOG has found topologies in conflict with a non-tristating function. Because it was weaker, it was discarded. This is usually due to an error in the design.

`Cannot assign equivalence.`
: Symbol in .EQUI statement assigned to more than one node, or node is out of range of valid nodes, or node already has symbol assigned to it.

`Cannot assign alias.`
: Same as above, but from alias statement, not `.EQUI`.

4.3 INFORMATORY Messages:

`Permanently off.`
: Transistor is logically disabled.

`Source/Drain Short.`
: Source and Drain of transistor are connected together.

5 Examples

The following pages contain examples of Sp2log input and output files.

Figures 1 through 5 illustrate a two-input exclusive-nor circuit. Figure 1 illustrates the Spice description, contained in file *xnor.spice*, and Figure 2 illustrates its schematic diagram. Figure 3 illustrates a user file named *user.demo*, and Figure 4 illustrates the command line to invoke Sp2log, specify the two input files, and generate the gate level description in file *xnor.net*. Figure 5 shows the generated gate level description.

Figures 6 through 10 illustrate another example, a two-input multiplexer. Figure 6 illustrates the Spice description, Figure 7 illustrates its schematic diagram, Figure 8 illustrates a pin file for this circuit, Figure 9 illustrates the command line to invoke Sp2log, and Figure 10 illustrates the resulting gate level description.

Figures

```
** A:I 341
** B:I 75
** OUT:O 351
** VDD:V 23
** VSS:G 0
M1 23 98 344 23 PE L=4.00U W=13.00U $ X=492.50 Y=713.50
M2 23 75 98 23 PE L=4.00U W=13.00U $ X=507.50 Y=713.50
M3 23 341 345 23 PE L=4.00U W=13.00U $ X=541.50 Y=713.50
M4 98 345 351 23 PE L=4.00U W=23.00U $ X=521.50 Y=836.50
M5 345 98 351 23 PE L=4.00U W=23.00U $ X=541.00 Y=836.50
M6 0 98 344 0 NE L=4.00U W=13.00U $ X=492.50 Y=746.50
M7 0 75 98 0 NE L=4.00U W=13.00U $ X=507.50 Y=746.50
M8 0 341 345 0 NE L=4.00U W=13.00U $ X=541.50 Y=746.50
M9 344 345 351 0 NE L=4.00U W=23.00U $ X=521.50 Y=793.50
M10 345 344 351 0 NE L=4.00U W=23.00U $ X=541.00 Y=793.5
```

Figure 1 Spice File "xnor.spice" Exclusive-Nor Circuit

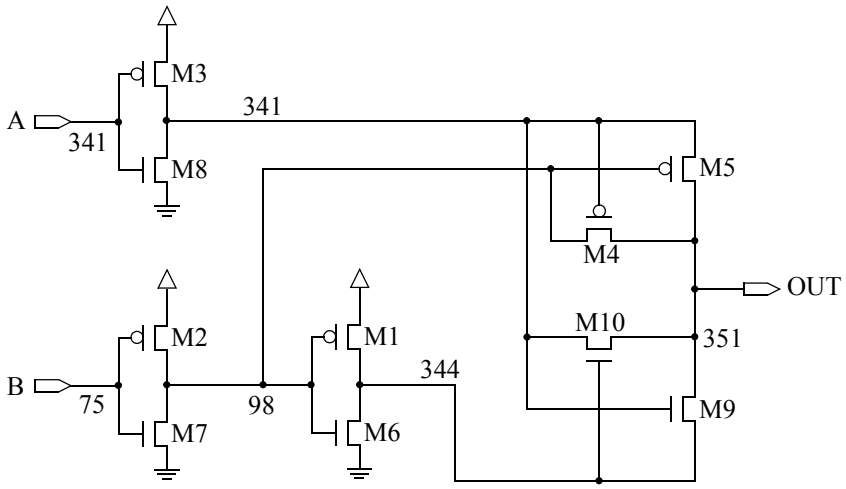


Figure 2 Circuit Diagram Of Exclusive-Nor Circuit

```
pcap = .5
ncap = .5
mratio = 2
br = "4e10 * %u * %p"
bf = "2e10 * %d * %n"
```

Figure 3 User File "demo.user" For Exclusive-Nor Circuit

```
sp2log xnor.spice xnor.net -udemo.user
```

Figure 4 Sp2log Command To Convert xnor Circuit

```
!Logical
Type= circuit $
  I= a,b $
  Ilod= 5.2e-11, 5.2e-11 $
  O= out $
  Transistors= 10
!Format Part= O= Type= I= Output-Rise= Output-Fall=
1 344 inv 98 [0,1.3e11] [0,6.5e10] Olod=4.6e-11 C=x=492.5 y=713.5
2 98 inv b [0,1.3e11] [0,6.5e10] Olod=9.8e-11 C= x=507.5 y=713.5
3 345 inv a [0.1.3e11] [0,6.5e10] Olod=9.2e-11 C= x=541.5 y=713.5
4 out exor 345,344 [0,2.3e11] [0,1.15e11] C= x=521.5 y=793.5

!Documentation
10 transistors removed from 10.
4 gates created.
```

Figure 5 Output File "xnor.net" From Above Command

```
*.EQUI A=341 B=75 C=98 OUT=351 VDD=23 VSS=0
*
*.GLOBAL 23 0
*
.SUBCKT MUX 341 75 98 351
*
M1 23 98 102 23 PE L=4.00U W=13.00U $ X=492.50 Y=713.50
M2 23 75 100 23 PE L=4.00U W=13.00U $ X=507.50 Y=713.50
M3 23 341 101 23 PE L=4.00U W=13.00U $ X=541.50 Y=713.50
M4 100 98 351 23 PE L=4.00U W=23.00U $ X=521.50 Y=836.50
M5 101 102 351 23 PE L=4.00U W=23.00U $ X=541.00 Y=836.50
M6 0 98 102 0 NE L=4.00U W=13.00U $ X=492.50 Y=746.50
M7 0 75 100 0 NE L=4.00U W=13.00U $ X=507.50 Y=746.50
M8 0 341 101 0 NE L=4.00U W=13.00U $ X=541.50 Y=746.50
M9 100 102 351 0 NE L=4.00U W=23.00U $ X=521.50 Y=793.50
M10 101 98 351 0 NE L=4.00U W=23.00U $ X=541.00 Y=793.50
```

Figure 6 Spice File "mux.spice" Two-Input Mux Circuit

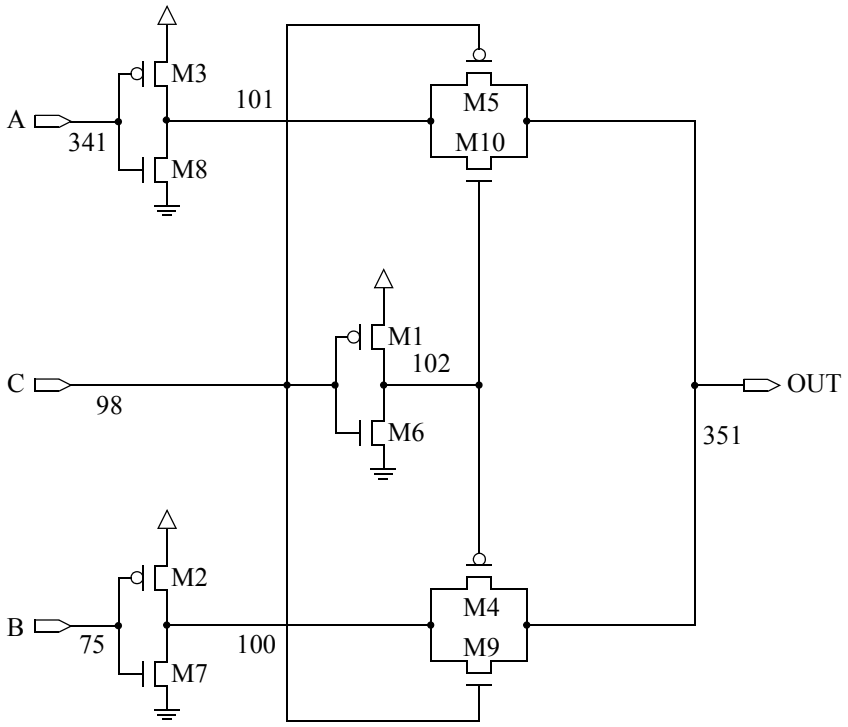


Figure 7 Circuit Diagram Of Two-Input Mux Circuit

```
A:I X = 51 Y = 5740.5 ATTACH METAL2
B:I X = 83 Y = 5740.5 ATTACH METAL2
C:I X = 115 Y = 5740.5 ATTACH METAL2
OUT:O X = 823 Y = 5740.5 ATTACH METAL2
VDD:P X = 0.0 Y = 15 ATTACH METAL1
VSS:P X = 0.0 Y = 267.5 ATTACH METAL1
```

Figure 8 Pin File "mux.pin" For Two-Input Mux

```
sp2log mux.spice mux.net -p mux.pin -u demo.user
```

Figure 9 Sp2log Command To Convert Two-Input Mux

```
!Logical
Type= mux $
  I= a,b,c $
  O= out $
  Transistors= 10
!Format Part= O= Type= I= Outout-Rise= Output-Fall=
1 102 inv c [0,1.3e11] [0,6.5e10] Olod=9.2e-11 C= x=492.5 y=713.5
2 100 inv b [0,1.3e11] [0,6.5e10] C= x=507.5 y=713.5
3 101 inv a [0,1.3e11] [0,6.5e10] C= x=541.5 y=713.5
4 out mux 100,101,102 [0,2.3e11] [0,1.1e11] C= x=521.5 y=793.5

!DOCUMENTATION
10 transistors removed from 10.
4 gates created.
```

Figure 10 Output File "mux.net" From Above Command