IBIS Interconnect SPICE Subcircuits Specification (IBIS-ISS)

Draft .2
August 11, 2009

Overview

The IBIS Open Forum, in order to enable easier data exchange between users of signal/power integrity simulation and physical layout/routing software tools, is issuing a generic netlist format, to be called “IBIS Interconnect SPICE Subcircuits” (IBIS-ISS).

This format would be similar in structure and major functions to the SPICE (Simulation Program with Integrated Circuit Emphasis) nodal syntax developed at the University of California at Berkeley and since implemented in various forms by individual software tool vendors. If approved, IBIS-ISS would be the first industry-standard version of SPICE Subcircuitss.

This version of IBIS-ISS is based on a subset of HSPICE ®, used with permission from Synopsys, Inc. HSPICE is a registered trademark of Synopsys, Inc.

Goals and Scope

The syntax of IBIS-ISS is intended for use to:

- describe interconnect structures (such as PCB traces, connectors, cables, etc.) electrically, for analysis in a signal integrity and/or power integrity context
- describe the arrangement or topology of interconnect structures, as they relate to each other and to active devices in a system

To these ends, IBIS-ISS will include support for:

- elementary circuit elements (resistors, capacitors, inductors)
- abstraction through modular, user-defined subcircuit definitions
- transmission line elements (lossless and lossy)
- frequency-domain network parameters (e.g., S-parameters)
- parameter/variable passing to elements and subcircuits
- dependent sources
- string-based node naming
- user-defined comments

IBIS-ISS will NOT include or cover:

- model formats or “process cards” for active devices (e.g., diodes, transistors)
- independent sources
- controls or options for any simulation engine (e.g., precision, algorithm selection)
- simulation or analysis types (e.g., DC, transient)
- sweep or run control (e.g., Monte Carlo)
- geometrical descriptions for field solvers
- support for other kinds of data extraction/export (e.g., S-parameter generation)
- measurement, printing or probing
- encryption support

Best Practices

Scaling

Scaling of interconnect subcircuits may give different results between different simulators and should be avoided.

Global Parameters

Global parameters may give different results between different simulators and should be avoided.

Exponent Range

Exponent range should be limited to between e-60 and e+60.

Numeric Scale Factors

Berkeley Spice does not support the “X” (Meg) scale factor and should be avoided.

Name Fields

A name field should begin with [a-z] or [A-Z], the remaining characters should be limited to [a-z], [A-Z], [0-9], ~!@#%&_<>?[]|:;

Parameter Names

Parameter Names should begin with [a-z] or [A-Z], and the remaining characters should be limited to [a-z] or [A-Z], **[0-9],** ! # $ % [ ] _

A Parameter should be defined in only one .param statement within a subckt.

Node Names

Node names should either be all numeric [0-9], or be a Name Field.

Conventions

The following typographical conventions are used in IBIS-ISS documentation.

Convention	Description
Courier	Indicates command syntax.
Italic	Indicates a user-defined value, such as object_name.
Bold	Indicates user input—text you type verbatim—in syntax and examples.
[ ]	Denotes optional parameters, such as: write_file [-f filename]
...	Indicates that parameters can be repeated as many times as necessary: pin1 pin2 ... pinN
|	Indicates a choice among alternatives, such as low | medium | high
+	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.
Control-c	Indicates a keyboard combination, such as holding down the Control key and pressing c.

4

Input Netlist and Data Entry

Describes the input netlist file and methods of entering data.

Input Netlist File Guidelines

An input filecontains the following:

- Design netlist (subcircuits, and so on).

An input filename can be up to 1024 characters long. The input netlist file cannot be in a packed or compressed format.

Statements in the input netlist file can be in any order.

N

Netlist input processing is case insensitive, except for file names and their paths.

Input Line Format

- The input reader can accept an input token, such as:
  - a statement name.
  - a node name.
  - a parameter name or value.

  Any valid string of characters between two token delimiters is a token.
- An input statement, or equation can be up to 1024 characters long.
- IBIS-ISS ignores differences between upper and lower case in input lines, except in quoted filenames.
- To continue a statement on the next line, enter a plus (+) sign as the first non-numeric, non-blank character in the next line.
- To indicate “to the power of” in your netlist, use two asterisks (**). For example, 2**5 represents two to the fifth power (2^5)
- To continue all IBIS-ISS statements, including quoted strings (such as paths and algebraics), use a single whitespace followed by a backslash (\) or a double backslash (\\) at the end of the line that you want to continue.

- A single backslash preserves white space.

■ Parameter names must begin with an alphabetic character, but thereafter can contain numbers and some special characters.

- Curly braces ({ }), are interpreted as square brackets ([ ]).
- Names are input tokens. Token delimiters must precede and follow names.
- Names can be up to 1024 characters long and are not case-sensitive.
- Do not use any of the time keywords as a parameter name or node name in your netlist.
- The following symbols are reserved operator keywords:

( ) = " ‘

Do not use these symbols as part of any parameter or node name that you define.

Special Characters

The following table lists the special characters that can be used as part of node names, element parameter names, and element instance names. For detailed discussion, see the appropriate sections in this chapter.

N

To avoid unexpected results or error messages, do not use the following mathematical characters in a parameter name in IBIS-ISS: *** - + ^** and **/**.

T *IBIS-ISS / Netlist Special Characters*

Special Character “Legal anywhere”=first character or any position in name “Included only”=any position except first character		**Node Name**	**Instance Name (cannot be the first character; element key letter only)**	**Parameter Name (cannot be the first character, element key letter only)**	**Delimiters**
~	tilde	Legal anywhere	Included only	Included only	n/a
!	exclamation point	Legal anywhere	Included only	Included only	n/a

@	at sign	Legal anywhere	included only	Included only	n/a
#	pound sign	Legal anywhere	Included only	Included only	n/a
$	dollar sign	Included only (avoid if after a number in node name)	Included only	Included only	In-line comment character
%	percent	Legal anywhere	Included only	included only,	n/a
^	caret	Legal anywhere	Included only	included only (avoid usage),	“To the power of”, i.e., 2^5, two raised to the fifth power
&	ampersand	Legal anywhere	Included only	Included only	n/a
*****	asterisk	included only (avoid using * in node names),	Included only	included only (avoid using in parameter names),	Comment in both IBIS-ISS Wildcard character. Double asterisk (**) is “To the power of”.
( )	parentheses	Illegal	Illegal	Illegal	Token delimiter
-	minus	included only	Included only	Illegal	n/a
_	underscore	Legal anywhere	Included only	Included only	n/a

+	plus sign	included only	Included only	included only (avoid usage); Illegal	Continues previous line, except for quoted strings (expressions, paths, algebraics)
=	equals	Illegal	Illegal	optional in .PARAM statements	Token delimiter
< >	less/more than	Legal anywhere	Included only	Included only	n/a
?	question mark	Legal anywhere	Included only	Included only	Wildcard in character in both IBIS-ISS
/	forward slash	Legal anywhere	Included only	Illegal	n/a
{ }	curly braces	included only, converts { } to [ ]	Included only	Included only	Auto-converts to square brackets ([ ])
[ ]	square brackets	Included only	Included only	Included only	n/a
****	backslash (requires a whitespace before to use as a continuation)	included only	Included only	Illegal	Continuation character for quoted strings (preserves whitespace)

****	double backslash (requires a whitespace before to use as a continuation)	included only	Illegal	Illegal	Continuation character for quoted strings (preserves whitespace)
|	pipe	Legal anywhere	Included only	Included only	n/a
,	comma	Illegal	Illegal	Illegal	Token delimiter
.	period	Illegal	Included only	Included only	Netlist keyword, (i.e., .PARAMETER, etc.).
:	colon	Included only	Included only	Included only	Delimiter for element attributes
;	semi-colon	Included only	Included only	Included only	n/a
" "	double-quotes	Illegal	Illegal	Illegal	Expression and filename delimiter
‘ ’	single quotes	Illegal	Illegal	Illegal	Expression and filename delimiter
	Blank (whitespace)	Use before \ or \\ line continuations			Token delimiter

First Character

The first character in every line specifies how IBIS-ISS interprets the remainder of the line.

T *First Character DescriptionsIBIS_ISS_DRAFTo2.doc*

Line	If the First Character is...	Indicates
Subsequent lines of netlist, and all lines of included files	. (period)	Netlist keyword. For example, .PARAM
	c, C, e, E, f, F, g, G, h, H, , k, K, l, L, r, R, s, S, v, V,w,W	Element instantiation
	* (asterisk)	Comment line
	+ (plus)	Continues previous line

Delimiters

- An input token is any item in the input file that IBIS-ISS recognizes. Input token delimiters are: tab, blank, comma (,), equal sign (=), and parentheses ( ).
- Single (‘) or double quotes (“) delimit expressions and filenames.
-
-

Instance Names

The names of element instances begin with the element key letter, except for subcircuit instances, whose instance names begin with X. (Subcircuits are sometimes called macros or modules.) Instance names can be up to 1024 characters long.

T *Element IdentifiersIBIS_ISS_DRAFTo2.doc*

Letter (First Char)	Element	Example Line
C	Capacitor	Cbypass 1 0 10pf
E	Voltage-controlled voltage source	Ea 1 2 3 4 K
F	Current-controlled current source	Fsub n1 n2 vin 2.0
G	Voltage-controlled current source	G12 4 0 3 0 10
H	Current-controlled voltage source	H3 4 5 Vout 2.0
K	Linear mutual inductor (general form)	K1 L1 L2 1
L	Linear inductor	LX a b 1e-9
R	Resistor	R10 21 10 1000
S	S-parameter element	S1 nd1 nd2 s_model2
V	Voltage source	V1 8 0 DC=0
W	Transmission Line	W1 in1 0 out1 0 N=1 L=1
T	Transmission Line	
X	Subcircuit call	X1 2 4 17 31 MULTI WN=100 LN=5

-

Numbers

You can enter numbers as integer, floating point, floating point with an integer exponent, or integer or floating point with one of the scale factors listed below.

T *Scale FactorsIBIS_ISS_DRAFTo2.doc*

Scale Factor	Prefix	Symbol	Multiplying Factor
T	tera	T	1e+12
G	giga	G	1e+9
MEG or X	mega	M	1e+6
K	kilo	k	1e+3
MIL	n/a	none	25.4e-6
M	milli	m	1e-3
U	micro	μ	1e-6
N	nano	n	1e-9
P	pico	p	1e-12
F	femto	f	1e-15
A	atto	a	1e-18

N

Scale factor A is not a scale factor in a character string that contains amps. For example, IBIS-ISS interprets the 20amps string as 20e-18mps (20^{-18}amps), but it correctly interprets 20amps as 20 amperes of current, not as 20e-18mps (20^{-18}amps).

- Numbers can use exponential format or engineering key letter format, but not both (1e-12 or 1p, but not 1e-6u).
- To designate exponents, use D or E.
- Trailing alphabetic characters are interpreted as units comments.
- Units comments are not checked.

Parameters and Expressions

- Parameter names use IBIS-ISS name syntax rules, except that names must begin with an alphabetic character. The other characters must be either a number, or one of these characters:

  ```
  ! # $ % [ ] _
  ```

- If you create multiple definitions for the same parameter, IBIS-ISS uses the last parameter definition even if that definition occurs later in the input than a reference to the parameter.
- You must define a parameter before you use that parameter to define another parameter.
- When you select design parameter names, be careful to avoid conflicts with parameterized libraries.
- To delimit expressions, use single quotes.
- Expressions cannot exceed 1024 characters.
- For improved readability, use a double slash (\\) at end of a line, to continue the line.

Using Subcircuits

Reusable cells are the key to saving labor in any CAD system. This also applies to circuit simulation, in IBIS-ISS

- To create and simulate a reusable circuit, construct it as a subcircuit.
- Use parameters to expand the utility of a subcircuit.

5

Parameters

Describes how to use parameters within IBIS-ISS *subckts.*

Parameters are similar to the variables used in most programming languages. Parameters hold a value that you assign when you create your circuit design or that the simulation calculates based on circuit solution values. Parameters can store static values for a variety of quantities (resistance, source voltage, rise time, and so on).

Using Parameters in Simulation (.PARAM)

Defining Parameters

Parameters in IBIS-ISS are names that you associate with numeric values. You can use any of the methods described below to define parameters.

T *.PARAM Statement SyntaxIBIS_ISS_DRAFTo2.doc*

Parameter	Description
Simple assignment	.PARAM <SimpleParam>=1e-12
Algebraic definition	.PARAM <AlgebraicParam>='SimpleParam*8.2' SimpleParam excludes the output variable.
Subcircuit default	.SUBCKT <SubName> <ParamDefName>=<Value> .

A parameter definition in IBIS-ISS always uses the last value found in the input netlist (subject to local versus global parameter rules). The definitions below assign a value of 3 to the *DupParam* parameter.

```
.PARAM DupParam=1
...
.PARAM DupParam=3
```

IBIS-ISS assigns 3 as the value for all instances of `DupParam`, including instances that are earlier in the input than the `.PARAM DupParam=3` statement.

All parameter values in IBIS-ISS are IEEE double floating point numbers. The parameter resolution order is:

1. Resolve all literal assignments.
2. Resolve all expressions.
3. Resolve all function calls.

Error: Reference source not found shows the parameter passing order.

T *Parameter Passing Order**IBIS_ISS_DRAFTo2.doc*

.PARAM statement ()	.SUBCKT call (instance)
.SUBCKT call (instance)	.SUBCKT definition (symbol)
.SUBCKT definition (symbol)	.PARAM statement (

Assigning Parameters

You can assign the following types of values to parameters:

- Constant real number
- Algebraic expression of real values
- Predefined function
- Circuit value
- Model value

To invoke the algebraic processor, enclose a complex expression in single quotes. A simple expression consists of one parameter name.

The parameter keeps the assigned value, unless a later definition changes its value.

Inline Parameter Assignments

To define circuit values, using a direct algebraic evaluation:

```
r1 n1 0 R='1k/sqrt(HERTZ)' $ Resistance for frequency
```

Using Algebraic Expressions

In IBIS-ISS, an algebraic expression, with quoted strings, can replace any parameter in the netlist.

Some uses of algebraic expressions are:

- Parameters:

```
.PARAM x='y+3'
```

- Algebra in elements:

```
R1 1 0 r='ABS(v(1)/i(m1))+10'
```

In addition to using quotations, you must define the expression inside the `PAR( )` statement for output.The continuation character for quoted parameter strings, in IBIS-ISS, is a double backslash (`\\`). (Outside of quoted strings, the single backslash (`\`) is the continuation character.)

Built-In Functions and Variables

In addition to simple arithmetic operations (`+`, `-`, `*`, `/`), you can use the built-in functions listed below and the variables listed below in IBIS-ISS expressions.

IBIS-ISS Form	Function	Class	Description
sin(x)	sine	trig	Returns the sine of x (radians)
cos(x)	cosine	trig	Returns the cosine of x (radians)
tan(x)	tangent	trig	Returns the tangent of x (radians)
asin(x)	arc sine	trig	Returns the inverse sine of x (radians)
acos(x)	arc cosine	trig	Returns the inverse cosine of x (radians)
atan(x)	arc tangent	trig	Returns the inverse tangent of x (radians)
sinh(x)	hyperbolic sine	trig	Returns the hyperbolic sine of x (radians)
cosh(x)	hyperbolic cosine	trig	Returns the hyperbolic cosine of x (radians)
tanh(x)	hyperbolic tangent	trig	Returns the hyperbolic tangent of x (radians)
abs(x)	absolute value	math	Returns the absolute value of x: |x|
sqrt(x)	square root	math	Returns the square root of the absolute value of x: sqrt(-x)=-sqrt(|x|)
pow(x,y)	absolute power	math	Returns the value of x raised to the integer part of y: $x^{(\text{integer part of y})}$
pwr(x,y)	signed power	math	Returns the absolute value of x, raised to the y power, with the sign of x: (sign of x)$\|x\|^y$
x**y	power		If x<0, returns the value of x raised to the integer part of y. If x=0, returns 0. If x>0, returns the value of x raised to the y power.

log(x)	natural logarithm	math	Returns the natural logarithm of the absolute value of x, with the sign of x: (sign of x)log(|x|)
log10(x)	base 10 logarithm	math	Returns the base 10 logarithm of the absolute value of x, with the sign of x: (sign of x)$\log_{10}(\|x\|)$
exp(x)	exponential	math	Returns e, raised to the power x: e^x
db(x)	decibels	math	Returns the base 10 logarithm of the absolute value of x, multiplied by 20, with the sign of x: (sign of x)$20\log_{10}(\|x\|)$
int(x)	integer	math	Returns the integer portion of x. The fractional portion of the number is lost.
nint(x)	integer	math	Rounds x up or down, to the nearest integer.
sgn(x)	return sign	math	Returns -1 if x is less than 0. Returns 0 if x is equal to 0. Returns 1 if x is greater than 0
sign(x,y)	transfer sign	math	Returns the absolute value of x, with the sign of y: (sign of y)|x|
def(x)	parameter defined	control	Returns 1 if parameter x is defined. Returns 0 if parameter x is not defined.
min(x,y)	smaller of two args	control	Returns the numeric minimum of x and y
max(x,y)	larger of two args	control	Returns the numeric maximum of x and y
[cond] ?x : y	ternary operator		Returns *x* if *cond* is not zero. Otherwise, returns *y*. .param z='condition ? x:y'
<	relational operator (less than)		Returns 1 if the left operand is less than the right operand. Otherwise, returns 0. .para x=y<z (y less than z)

<=	relational operator (less than or equal)	Returns 1 if the left operand is less than or equal to the right operand. Otherwise, returns 0. .para x=y<=z (y less than or equal to z)
>	relational operator (greater than)	Returns 1 if the left operand is greater than the right operand. Otherwise, returns 0. .para x=y>z (y greater than z)
>=	relational operator (greater than or equal)	Returns 1 if the left operand is greater than or equal to the right operand. Otherwise, returns 0. .para x=y>=z (y greater than or equal to z)
==	equality	Returns 1 if the operands are equal. Otherwise, returns 0. .para x=y==z (y equal to z)
!=	inequality	Returns 1 if the operands are not equal. Otherwise, returns 0. .para x=y!=z (y not equal to z)
&&	Logical AND	Returns 1 if neither operand is zero. Otherwise, returns 0. .para x=y&&z (y AND z)
||	Logical OR	Returns 1 if either or both operands are not zero. Returns 0 only if both operands are zero. .para x=y||z (y OR z)

IBIS-ISS Form	Function	Class	Description
time	current simulation time	control	Uses parameters to define the current simulation time, during transient analysis.

temper	current circuit temperature	control	Uses parameters to define the current simulation temperature, during transient/temperature analysis.
hertz	current simulation frequency	control	Uses parameters to define the frequency, during AC analysis.

Parameter Scoping and Passing

If you use parameters to define values in sub-circuits, you need to create fewer similar cells, to provide enough functionality in your library. You can pass circuit parameters into hierarchical designs, and assign different values to the same parameter within individual cells, when you run simulation.

A parameter is defined either by a .parameter statement (local to that subcircuit), or can be passed into a subcircuit, or can be defined on a subckt definition line.

(Some details need to be clarified on this)

```
.param x=0
.subckt def
.param x=1
x1 1 2 abc x=2
.subckt abc 1 2 x=3
.param x=3
r1 1 2 R=x
.ends abc
.ends def
.end
```

How you handle hierarchical parameters depends on how you construct and analyze your cells. You can construct a design in which information flows from the top of the design, down into the lowest hierarchical levels.

- To construct a library of small cells that are individually controlled from within, set *local* parameters and build up to the block level.

This section describes the scope of parameter names, and how IBIS-ISS resolves naming conflicts between levels of hierarchy.

Library Integrity (Needs careful discussion)

Integrity is a fundamental requirement for any symbol library. Library integrity can be as simple as a consistent, intuitive name scheme, or as complex as libraries with built-in range checking.

Library integrity might be poor if you use libraries from different vendors in a circuit design. Because names of circuit parameters are not standardized between vendors, two components can include the same parameter name for different functions. For example, one vendor might build a library that uses the name Tau as a parameter to control one or more subcircuits in their library. Another vendor might use Tau to control a different aspect of their library. If you set a global parameter named Tau to control one library, you also modify the behavior of the second library, which might not be the intent. This is why Best Practices recommends that Global Parameters be avoided.

Subcircuits

X<*subcircuit_name*> adds an instance of a subcircuit to your netlist. You must already have defined that subcircuit in your netlist by using a `.SUBCKT` command.

Syntax

X<*subcircuit_name*> *n1* <*n2 n3* …> *subnam*
<*parnam = val* &> <*M = val*> <S=*val*> <DTEMP=*val*>

Argument	Definition
X<*subcircuit_name*>	Subcircuit element name. Must begin with an X, followed by up to 15 alphanumeric characters.
n1 …	Node names for external reference.
subnam	Subcircuit model reference name.
parnam	A parameter name set to a value (val) for use only in the subcircuit. It overrides a parameter value in the subcircuit definition, but is overridden by a value set in a `.PARAM` statement.

Subckt scoping rules

A .subckt or .model definition must occur in the subckt in which the subckt or model is referenced, or in a calling subckt at any level above.

.INCLUDE

Includes another netlist as a subcircuit of the current netlist.

Syntax

.INCLUDE ‘file_path file_name’

Arguments

Argument	Description
file_path	Path name of a file for computer operating systems that support tree-structured directories. An include file can contain nested .INCLUDE calls to itself or to another include file. If you use a relative path in a nested .INCLUDE call, the path starts from the directory of the parent .INCLUDE file, not from the current working directory. If the path starts from the current working directory, IBIS-ISS can also find the .INCLUDE file, but prints a warning.
file_name	Name of a file to include in the data file. The file path, plus the file name, can be up to 16 characters long. You can use any valid file name for the computer’s operating system.

Description

Use this command to include another netlist in the current netlist. You can include a netlist as a subcircuit in one or more other netlists. You must enclose the file path and file name in single or double quotation marks. Otherwise, an error message is generated.

Example

.INCLUDE `/myhome/subcircuits/diode_circuit´

Node Name (or Node Identifier) Conventions

Nodes are the points of connection between elements in the input netlist. Either names or numbers may be used to designate nodes. Node numbers can be from 1 to 999999999999999 (1 to 1e16-1); node number 0 is always ground. Letters that follow numbers in node names are ignored.

When the node name begins with a letter or a valid special character, the node name can contain a maximum of 1024 characters.

Subcircuit Node Names

Two subcircuit node names are assigned in this format.

To indicate the ground node, use either the number 0, the name `GND`, or `!GND, or GROUND, GND!`. Every node should have at least two connections, except for transmission line nodes (unterminated transmission lines are permitted) and MOSFET substrate nodes (which have two internal connections).

Element, Instance, and Subcircuit Naming Conventions

Instances and subcircuits are elements and as such, follow the naming conventions for elements.

Element names begin with a letter designating the element type, followed by up to 1023 alphanumeric characters. Element type letters are R for resistor, C for capacitor and so on.

Comments and Line Continuation

Comments require an asterisk (*) as the first character in a line or a dollar sign ($) directly in front of the comment anywhere on the line. For example:

```
* <comment_on_a_line_by_itself>
```

or

```
< IBIS-ISS statement> $ <comment following input>
```

Comment statements may appear anywhere in the circuit description. The dollar sign ($) must be used for comments that do *not* begin in the first character position on a line (for example, for comments that follow simulator input on the same line). If it is not the first nonblank character, then the dollar sign must be preceded by either:

- Whitespace
- Comma (,)
- Valid numeric expression

The dollar sign may also be used within node or element names. For example:

```
* RF=1K GAIN SHOULD BE 100
$ MAY THE FORCE BE WITH MY CIRCUIT
VIN 1 0 PL 0 0 5V 5NS $ 10v 50ns
R12 1 0 1MEG $ FEED BACK
.PARAM a=1w$comment a=1, w treated as a space and ignored
.PARAM a=1k$comment a=1e3, k is a scale factor
```

A dollar sign is the preferred way to indicate comments, because of the flexibility of its placement within the code.

Line continuations require a plus sign (+) as the first character in the line that follows. Here is an example of comments and line continuation in a netlist file:

```
.ABC Title Line
* on this line, because the first line is always a comment)
* This is a comment line
.MODEL n1 NMOS $ this is an example of an inline comment
* This is a comment line and the following line is a continuation
+ LEVEL=3
```

Elements

Linear Resistors

R*xxx node1 node2* [R =] *value*

The value of a linear resistor can be a constant, or an expression of parameters.

Parameter	Description
Rxxx	Name of a resistor
node1 and node2	Names or numbers of the connecting nodes
value	resistance value, in ohms

Linear Capacitors

C*xxx node1 node2* [C=]*val*

The value of a linear capacitor can be a constant, or an expression of parameters.

Parameter	Description
Cxxx	Name of a capacitor. Must begin with C, followed by up to 1023 alphanumeric characters.
node1 and node2	Names or numbers of connecting nodes.
value	capacitance value, in Farads.

Voltage Shunt

Vxxx node1 node2 [DC=]0

This creates a short between nodes node1 and node2

Mutual Inductors

General form:

K*xxx* L*yyy* L*zzz* [K=] *coupling*

Parameter	Description
Kxxx	Mutual inductor element name. Must begin with K, followed by up to 1023 alphanumeric characters.
Lyyy	Name of the first of two coupled inductors.
Lzzz	Name of the second of two coupled inductors.
K=coupling	Coefficient of mutual coupling. K is a unitless number, with magnitude > 0. If K is negative, the direction of coupling reverses. This is equivalent to reversing the polarity of either of the coupled inductors. Use the K=coupling syntax when using a parameter value or an equation, and the keyword “k=” can be omitted.

Linear Inductors

L*xxx node1 node2* [L =] *inductance*

Parameter	Description
Lxxx	Name of an inductor.
node1 and node2	Names or numbers of the connecting nodes.
inductance	inductance value, in Henries.

T-element (Ideal Transmission Lines)

General form:

Txxx in refin out refout Z0=val TD=val [L=val]

+ [IC=v1,i1,v2,i2]

Parameter	Description
Txxx	Lossless transmission line element name. Must begin with T, followed by up to 1023 alphanumeric characters.
in	Signal input node.
refin	Ground reference for the input signal.
out	Signal output node.
refout	Ground reference for the output signal.
Z0	Characteristic impedance of the transmission line (Ohms).
TD	Propagation time delay of the transmission line (in seconds). If physical length (L) is specified, then units for TD are considered in seconds per meter.
L	Physical length of the transmission line, in units of meters. Default=1.

W-element Modeling of Coupled Transmission Lines

Describes how to use basic transmission line simulation equations and an optional method for computing the parameters of transmission line equations.

The W-element is a versatile transmission line model that you can apply to efficiently and accurately simulate transmission lines, ranging from a simple lossless line to complex frequency-dependent lossy-coupled lines.

Input Syntax for the W-element

Syntax:

```
Wxxx i1 i2 ... iN iR o1 o2 ... oN oR N=val L=val
+ [RLGCMODEL=name | TABLEMODEL=name ]
```

Parameter	Description
N	Number of signal conductors (excluding the reference conductor).
i1...iN	Node names for the near-end signal-conductor terminal
iR	Node name for the near-end reference-conductor terminal.
o1... oN	Node names for the far-end signal-conductor terminal
oR	Node name for the far-end reference-conductor terminal.
L	Length of the transmission line.
RLGCMODEL	Name of the RLGC model.
TABLEMODEL	Name of the frequency-dependent tabular model

The W-element supports these formats to specify transmission line properties:

- Model 1: RLGC-Model specification
  - Internally specified in a `.MODEL` statement.
  - Externally specified in a different file.
- Model 4: Frequency-dependent tabular model.

Normally, you can specify parameters in the W-element card in any order. Specify the number of signal conductors, N, after the list of nodes. You can intermix the nodes and parameters in the W-element card.

Input Model 1: W-element, RLGC Model

Equations and Parameters on page 96 (NOTE: Do we want to include these explanations) describes the inputs of the W-element per unit length matrices: R_o (DC resistance), L, G, C, R_s (skin effect), and G_d (dielectric loss)

The W-element does not limit any of the following parameters:

- Number of coupled conductors.
- Shape of the matrices.
- Line loss.
- Length or amount of frequency dependence.

The RLGC text file contains frequency-dependent RLGC matrices per unit length. The W-element also handles frequency-independent RLGC, and lossless (LC) lines. It does not support RC lines.

Because RLGC matrices are symmetrical, the RLGC model specifies only the lower triangular parts of the matrices. The syntax of the RLGC model for the W-element is:

```
.MODEL name W MODELTYPE=RLGC N=val
+ Lo=matrix_entries
+ Co=matrix_entries [Ro=matrix_entries Go=matrix_entries]
+ Rs=matrix_entries wp=val Gd=matrix_entries Rognd=val
+ Rsgnd=val Lgnd=val
```

Parameter	Description
N	Number of conductors (same as in the element card).
L	DC inductance matrix, per unit length $\left[\frac{H}{m}\right]$.

C

DC capacitance matrix, per unit length $\left[\frac{F}{m}\right]$.

Ro

DC resistance matrix, per unit length $\left[\frac{\Omega}{m}\right]$.

Go

DC shunt conductance matrix, per unit length $\left[\frac{S}{m}\right]$.

Rs

Skin effect resistance matrix, per unit length $\left[\frac{\Omega}{m\sqrt{Hz}}\right]$.

Gd

Dielectric loss conductance matrix, per unit length $\left[\frac{S}{m \cdot Hz}\right]$.

wp

Angular frequency of the polarization constant [radian/sec] (see Introduction to the Complex Dielectric Loss Model on page 99)). When the wp value is specified, the unit of Gd becomes [S/m].

Lgnd

DC inductance value, per unit length for grounds $\left[\frac{H}{m}\right]$ (reference line).

Rognd

DC resistance value, per unit length for ground $\left[\frac{\Omega}{m}\right]$.

Rsgnd

Skin effect resistance value, per unit length for ground $\left[\frac{\Omega}{m\sqrt{Hz}}\right]$.

The following input netlist file shows RLGC input for the W-element:

```
* W-Element example, four-conductor line
W1 N=3 1 3 5 0 2 4 6 0 RLGCMODEL=example_rlc l=0.97


* RLGC matrices for a four-conductor lossy
.MODEL example_rlc W MODELTYPE=RLGC N=3
+ Lo=
+ 2.311e-6
+ 4.14e-7 2.988e-6
+ 8.42e-8 5.27e-7 2.813e-6
```

```
+ Co=
+ 2.392e-11
+ -5.41e-12 2.123e-11
+ -1.08e-12 -5.72e-12 2.447e-11
+ Ro=
+ 42.5
+ 0 41.0 + 0 0 33.5
+ Go= + 0.000609
+ -0.0001419 0.000599
+ -0.00002323 -0.00009 0.000502
+ Rs=
+ 0.00135
+ 0 0.001303
+ 0 0 0.001064
+ Gd=
+ 5.242e-13
+ -1.221e-13 5.164e-13
+ -1.999e-14 -7.747e-14 4.321e-13
```

Using RLGC Matrices

RLGC matrices in the RLGC model of the W-element are in the Maxwellian format

---

Input Model 4: Frequency-Dependent Tabular Model

You can use the tabular RLGC model as an extension of the analytical RLGC model to model any arbitrary frequency-dependent behavior of transmission lines (this model does not support RC lines).

You can use this extension of the W-element syntax to specify a table model (use a `.MODEL` statement of type w). To accomplish this, the `.MODEL` statement refers to .MODEL statements where the “type” is SP (described in Small-Signal Parameter Data Frequency Table Model (SP Model) on page 77), which contain the actual table data for the RLGC matrices.

N

To ensure accuracy, the W-element tabular model requires the following:

- R and G tables require zero frequency points.
- L and C tables require infinity frequency points as well as zero frequency points.

To specify a zero frequency point, you may use DC keyword or f=0 data entry in the DATA field of the SP model. To specify an infinity frequency point, use the INFINITY keyword of the SP model.

See also, Small-Signal Parameter Data Frequency Table Model (SP Model) on page 77.

Notation Used

- Lower-case variable: Scalar quantity
- Upper-case variable: Matrix quantity
- All upper-case words: Keyword
- Parentheses and commas: Optional

Table Model Card Syntax

```
.MODEL name W MODELTYPE=TABLE [FITGC=0|1] N=val
+ LMODEL=l_freq_model CMODEL=c_freq_model
+ [RMODEL=r_freq_model GMODEL=g_freq_model]
```

Parameter	Description
FITCG	Keyword for W Model (w/ MODELTYPE=TABLE) 1=causality check on, 0= causality check off (default)
N	Number of signal conductors (excluding the reference conductor).
LMODEL	SP model name for the inductance matrix array.
CMODEL	SP model name for the capacitance matrix array.
RLMODEL	SP model name for the resistance matrix array. By default, it is zero.
GMODEL	SP model name for the conductance matrix array. By default, it is zero.

S-element Syntax

Use the following S-element syntax to show the connections within a circuit:

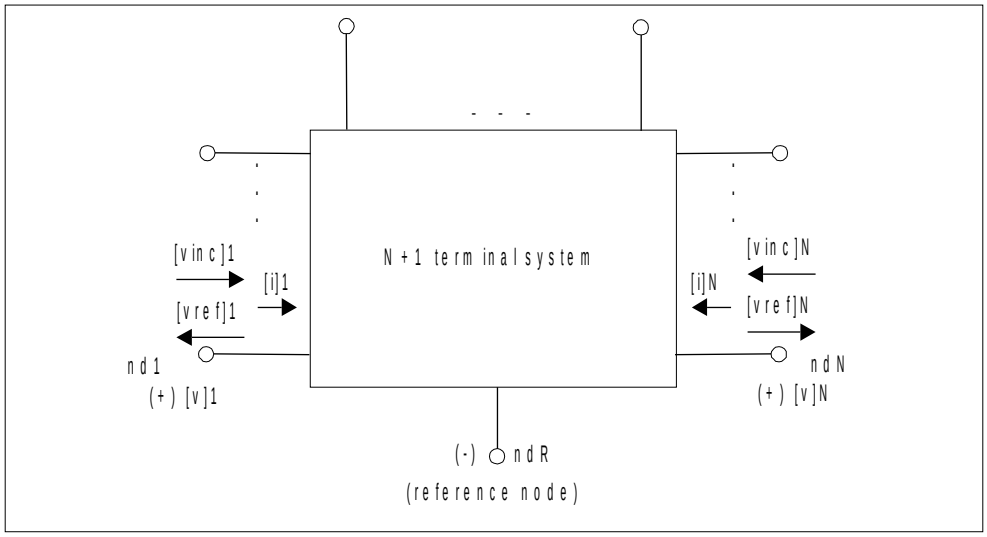
```
Sxxx nd1 nd2 ... ndN [ndRef]
+ [MNAME=Smodel_name]
+ [FBASE = base_frequency] [FMAX=maximum_frequency]
```

Parameter	Description
nd1 nd2...ndN	Nodes of an S-element Three kinds of definitions are present: ■With no reference node ndRef, the default reference node is GND. Each node ndi (i=1~N) and GND construct one of the N ports of the S-element. ■With one reference node, ndRef is defined. Each node ndi (i=1~N) and the ndRef construct one of the N ports of the S-element. ■With an N reference node, each port has its own reference node. You can write the node definition in a clearer way as: nd1+ nd1- nd2+ nd2- ... ndN+ ndN- Each pair of the nodes (ndi+ and ndi-, i=1~N) constructs one of the N ports of the S-element.
ndRef	Reference node
MNAME	Name of the S model; Note that string parameters are supported in calling an MNAME.
FBASE	Base frequency to use for transient analysis. This value becomes the base frequency point for Inverse Fast Fourier Transformation (IFFT). ■If you do not set this value, the base frequency is a reciprocal value of the transient period. ■If you set a frequency that is smaller than the reciprocal value of the transient, then transient analysis performs circular convolution, and uses the reciprocal value of FBASE as its base period.

FMAX	Maximum frequency use in transient analysis. Used as the maximum frequency point for Inverse Fast Fourier Transformation (IFFT).

The nodes of the S-element must come first. You can specify all the optional parameters in both the
S-element and S model statements, except for `MNAME` argument.

You can enter the optional arguments in any order, and the parameters specified in the element statement have a higher priority.



F *Terminal Node Notation*

Node Example

The following example illustrates the *nd1 nd2...ndN—no reference*, *single reference*, and multi-reference parameters.

```
**S-parameter example



* no reference
S_no_ref n1 n2 mname=s_model

* single reference
S_one_ref n1 n3 gnd mname=s_model

*multi-reference
S_multi_ref n1 gnd n4 gnd mname=s_model
```

The S-element must have a call to one of the supported S-parameter file formats (IBIS-ISS gets the number of ports from the S-parameter file You can also explicitly specify N=n where ‘n’ is the number of ports.

- For n terminals, the S-element assumes no reference node.
- For n+1 terminals, the S-element assumes one reference node.
- For 2n terminals, the S-element assumes signal nodes and n reference nodes. Each pair of nodes is a signal and a reference node.

S Model Syntax

Use the following syntax to describe specific S models:

```
.MODEL Smodel_name S [N=dimension]
+ [TSTONEFILE=filename

+ [FBASE=base_frequency] [FMAX=maximum_frequency]
```

Parameter	Description
Smodel_name	Name of the S model.
S	Specifies that the model type is an S model.
N	S model dimension, which is equal to the terminal number of an S-element and excludes the reference node.

TSTONEFILE	Specifies the name of a Touchstone file. Data contains frequency-dependent array of matrixes. Touchstone files must follow the .s#p file extension rule, where # represents the dimension of the network. Note that string parameters are supported for TSTONEFILE Example: .subckt sparam n1 n2 tsfile=str('ss_ts.s2p') S1 n1 n2 0 mname=s_model .model s_model S TSTONEFILE=str(tsfile) .ends x1 A B sparam tsfile=str('ss_ts.s2p') … For details, see *Touchstone® File Format Specification* by the EIA/IBIS Open Forum (http://www.eda.org).
FBASE	Base frequency used for transient analysis. IBIS-ISS uses this value as the base frequency point for Fast Inverse Fourier Transformation (IFFT). ■If FBASE is not set, IBIS-ISS uses a reciprocal of the transient period as the base frequency. ■If FBASE is set smaller than the reciprocal value of transient period, transient analysis performs circular convolution by using the reciprocal value of FBASE as a base period.
FMAX	Maximum frequency for transient analysis. Used as the maximum frequency point for Inverse Fast Fourier Transform (IFFT).

The, `TSTONEFILE` parameters describe the frequency-varying behavior of a network.

Voltage-Controlled Voltage Source (VCVS)

Linear

```
Exxx n+ n- [VCVS] in+ in- gain
```

For a description of these parameters, see table VCVS Parameters.

Laplace Transform

Voltage Gain H(s):

```
Exxx n+ n- LAPLACE in+ in-   k0, k1, ..., kn / d0,
   d1, ..., dm
```

For a description of these parameters, see table VCVS Parameters.

H(s) is a rational function, in the following form:You can use parameters to define the values of all coefficients (k_0, k_1, ..., d_0, d_1, ...).

Pole-Zero Function

Voltage Gain H(s):

```
Exxx n+ n- POLE in+ in- a az1, fz1, ..., azn, fzn / b,
+ ap1, fp1, ..., apm, fpm
```

For a description of these parameters, see table VCVS Parameters.

The following equation defines H(s) in terms of poles and zeros:

$$H(s) = \frac{a \cdot (s + \alpha_{z1} - j2\pi f_{z1}) \ldots (s + \alpha_{zn} - j2\pi f_{zn})(s + \alpha_{zn} + j2\pi f_{zn})}{b \cdot (s + \alpha_{p1} - j2\pi f_{p1}) \ldots (s + \alpha_{pm} - j2\pi f_{pm})(s + \alpha_{pm} + j2\pi f_{pm})}$$

The complex poles or zeros are in conjugate pairs. The element description specifies only one of them, and the program includes the conjugate. You can use parameters to specify the a, b, α, and f values.

Example

```
Elow_pass out 0 POLE in 0 1.0 / 1.0, 1.0,0.0 0.5,0.1379
```

The `Elow_pass` statement describes a low-pass filter, with the transfer function:

$$H(s) = \frac{1.0}{1.0 \cdot (s+1)(s+0.5+j2\pi \cdot 0.1379)(s+0.5-(j2\pi \cdot 0.1379))}$$

■

A

Foster Pole-Residue Form

Gain E(s) form

```
Exxx n+ n- FOSTER in+ in- k0 k1
+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})
+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})
+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})
+ ...
```

For a description of these parameters, see table VCVS Parameters.

In the above syntax, parenthesis , commas, and slashes are separators—they have the same meaning as a space. A pole-residue pair is represented by four numbers (real and imaginary part of the residue, then real and imaginary part of the pole).

You must make sure that Re[pi]<0; otherwise, the simulations will certainly diverge. Also, it is a good idea to assure passivity of the model (for an N-port admittance matrix Y, Re{Y} should be positive-definite), or the simulation is likely to diverge).

N

For real poles, half the residue value is entered because it is applied twice. In the above example, the first pole-residue pair is real, but is written as “A1/(s-p1)+A1/(s-p1)”; therefore, 0.0004 is entered rather than 0.0008.

Table VCVS Parameters.

E-element Parameters

The E-element parameters are described in the following list.

Parameter	Description
Exxx	Voltage-controlled element name. Must begin with E, followed by up to 1023 alphanumeric characters.
gain	Voltage gain.
in +/-	Positive or negative controlling nodes. Specify one pair for each dimension.
k	Ideal transformer turn ratio: $V(in+,in-) = k \cdot V(n+,n-)$ or, number of gates input.
n+/-	Positive or negative node of a controlled element.
VCVS	Keyword for a voltage-controlled voltage source. VCVS is a reserved word; do not use it as a node name.

Current-Dependent Current Sources — F-elements

This section explains the F-element syntax and parameters.

N

G-elements with algebraics make F-elements obsolete. You can still use F-elements for backward-compatibility with existing designs.

Current-Controlled Current Source (CCCS) Syntax

Linear

```
Fxxx n+ n- <CCCS> vn1 gain
```

F-element Parameters

The F-element parameters are described in the following list.

Parameter	Description
CCCS	Keyword for current-controlled current source. CCCS is a IBIS-ISS reserved keyword; do not use it as a node name.
Fxxx	Element name of the current-controlled current source. Must begin with F, followed by up to 1023 alphanumeric characters.
gain	Current gain.
n+/-	Connecting nodes for a positive or negative controlled source.
vn1 …	Names of voltage sources, through which the controlling current flows. Specify one name for each dimension.
x1,...	Controlling current, through the *vn1* source. Specify the *x* values in increasing order.

y1,...	Corresponding output current values of *x*.

Voltage-Dependent Current Sources — G-elements

This section explains G-element syntax statements, and their parameters.

```
Gxxx n+ n- <VCCS| > in+ in- ...
```

Voltage-Controlled Current Source (VCCS)

Linear

```
Gxxx n+ n- <VCCS> in+ in- transconductance
```

For a description of the G-element parameters, see Table VCCS Parameters.

Laplace Transform

Transconductance H(s):

```
Gxxx n+ n- LAPLACE in+ in-   k0, k1, ..., kn / d0,
   d1, ..., dm
```

H(s) is a rational function, in the following form:

$$H(s) = \frac{k_0 + k_1 s + \ldots + k_n s^n}{d_0 + d_1 s + \ldots + d_m s^m}$$

You can use parameters to define the values of all coefficients (k_0, k_1, ..., d_0, d_1, ...).

Pole-Zero Function

Transconductance H(s):

```
Gxxx n+ n- POLE in+ in- a az1, fz1, ..., azn, fzn / b,
+ ap1, fp1, ..., apm, fpm
```

The following equation defines H(s) in terms of poles and zeros:

$$H(s) = \frac{a \cdot (s + \alpha_{z1} - j2\pi f_{z1}) \ldots (s + \alpha_{zn} - j2\pi f_{zn})(s + \alpha_{zn} + j2\pi f_{zn})}{b \cdot (s + \alpha_{p1} - j2\pi f_{p1}) \ldots (s + \alpha_{pm} - j2\pi f_{pm})(s + \alpha_{pm} + j2\pi f_{pm})}$$

The complex poles or zeros are in conjugate pairs. The element description specifies only one of them, and the program includes the conjugate. You can use parameters to specify the a, b, α, and f values.

For a description of the G-element parameters, see table VCVS Parameters.

Example

```
Ghigh_pass 0 out POLE in 0 1.0 0.0,0.0 / 1.0 0.001,0.0
```

The `Ghigh_pass` statement describes a high-pass filter, with the transfer function:

$$H(s) = \frac{1.0 \cdot (s + 0.0 + j \cdot 0.0)}{1.0 \cdot (s + 0.001 + j \cdot 0.0)}$$

Foster Pole-Residue Form

Transconductance G(s) form

```
Gxxx n+ n- FOSTER in+ in- k0 k1
+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})
+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})
+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})
+ ...
```

In the above syntax, parenthesis , commas, and slashes are separators—they have the same meaning as a space. A pole-residue pair is represented by four numbers (real and imaginary part of the residue, then real and imaginary part of the pole).

You must make sure that Re[pi]<0; otherwise, the simulations will certainly diverge. Also, it is a good idea to assure passivity of the model (for an N-port admittance matrix Y, Re{Y} should be positive-definite), or the simulation is likely to diverge).

For a description of the G-element parameters, see table VCVS Parameters.

Example

To represent a G(s) in the form,

$$G(s) = 0.001 + 1 \times 10^{-12} s + \frac{0.0008}{s + 1 \times 10^{10}} + \frac{(0.001 - j0.006)}{s - (-1 \times 10^{8} + j1.8 \times 10^{10})} + \frac{(0.001 + j0.006)}{s - (-1 \times 10^{8} - j1.8 \times 10^{10})}$$

You would input:

```
G1 1 0 FOSTER 2 0 0.001 1e-12
+(0.0004, 0)/(-1e10, 0) (0.001, -0.006)/(-1e8, 1.8e10)
```

N

For real poles, half the residue value is entered because it is applied twice. In the above example, the first pole-residue pair is real, but is written as “A1/(s-p1)+A1/(s-p1)”; therefore, 0.0004 is entered rather than 0.0008.

Table VCCS Parameters.

G-element Parameters

The G-element parameters described in the following list.

Parameter	Description
Gxxx	Name of the voltage-controlled element. Must begin with G, followed by up to 1023 alphanumeric characters.
in +/-	Positive or negative controlling nodes. Specify one pair for each dimension.
n+/-	Positive or negative node of the controlled element.
transconductance	Voltage-to-current conversion factor.
VCCS	Keyword for the voltage-controlled current source. VCCS is a reserved IBIS-ISS keyword; do not use it as a node name.
x1,...	Controlling voltage, across the *in*+ and *in*- nodes. Specify the *x* values in increasing order.
y1,...	Corresponding element values of *x*.

Current-Dependent Voltage Sources — H-elements

This section explains H-element syntax statements, and defines their parameters.

N

E-elements with algebraics make H-elements obsolete. You can still use H-elements for backward-compatibility with existing designs.

Current-Controlled Voltage Source (CCVS)

Linear

```
Hxxx n+ n- <CCVS> vn1 transresistance
```

Parameter	Description
CCVS	Keyword for the current-controlled voltage source. CCVS is a IBIS-ISS reserved keyword; do not use it as a node name.
Hxxx	Element name of current-controlled voltage source. Must start with H, followed by up to 1023 alphanumeric characters.
n+/-	Connecting nodes for positive or negative controlled source.
transresistance	Current-to-voltage conversion factor.
vn1 …	Names of voltage sources, through which controlling current flows. You must specify one name for each dimension.
x1,...	Controlling current, through the *vn1* source. Specify the *x* values in increasing order.
y1,...	Corresponding output voltage values of *x*.