Acceptable Syntax for the SIMPLIS PDN Parser

The SIMPLIS PDN Parser expects a SPICE-format description of a parasitic network. This description will be referred to as the input file for the parser. The syntax acceptable by this parser is by-and-large the SPICE-3 format, with a slight change to support the usage of Laplace Transfer functions.

Since it is a parasitic network, it is expected all elements defined in the input file are linear. If the first non-whitespace character in a line is the asterisk ('*'), then this line is considered a comment line. If the ***first character*** in a line is the plus sign ('+'), then this line is considered the typical SPICE continuation line. It is a good practice to have a space character immediately following this line continuation character to improve the readability. Empty lines and comment lines are ignored.

The PDN parser accepts the following types of statements:

- A statement defining the start of a subcircuit
- A statement defining the end of a subcircuit
- A statement defining a resistor
- A statement defining an inductor
- A statement defining a capacitor
- A statement defining a DC voltage source
- A statement defining a DC current source
- A statement defining a voltage-controlled voltage source
- A statement defining a voltage-controlled current source
- A statement defining a current-controlled voltage source
- A statement defining a current-controlled current source
- A statement defining coupling coefficient between two inductors

These statements would involve naming of nodes. Node names are made up of alphanumeric characters and the underscore character (‘_’). The minus sign (‘-‘) and the dollar sign (‘$’) are reserved by the parser for its own usage. Avoid using these two characters in the node names.

The above restriction for the node names applies to the reference designators as well. Hence, it is not a good idea to name a capacitor C$23-5.

All reference designators, node names, and subcircuit names are treated in a case insensitive manner. Having a C123 and a c123 defined in two different statements in the same subcircuit is considered an error. While the PDN parser will not report these two statements as an error, the SIMetrix and SIMPLIS simulators will eventually report these two statements as errors.

Statement defining the start of a subcircuit

The start of a subcircuit is defined by the .SUBCKT statement. The .SUBCKT statement has the following syntax:

```
.SUBCKT  name_of_subckt  node1  node2  ...
```

name_of_subckt is the name of the subcircuit to be defined and the restriction on the characters that can be used in node names apply to the name of the subcircuit as well. node1 and node2 are node names and ... represents more node names if the subcircuit has more than two nodes to connect to the outside world.

Statement defining the end of a subcircuit

The end of a subcircuit is defined by the .ENDS statement. The .ENDS statement has the following syntax:

```
.ENDS name_of_subckt
```

Statements defining resistors, inductors, and capacitors

The statements defining these three types of components are:

```
R???    node_P  node_N       resistance
L???    node_P  node_N       capacitance
C???    node_P  node_N       inductance
```

R???, L???, and C??? are the reference designators. resistance, capacitance, and inductance are constant component values defined in the typical %e format of C or similar programming languages. For instance, resistance could be 714.374653476 or 7.14374653476e+02.

Statements defining DC voltage and current sources

The statements defining these two types of components are:

```
V???    node_P  node_N       DC      0.0
I???    node_P  node_N       DC      0.0
```

V??? and I??? are the reference designators. A DC voltage source can have a non-zero voltage. But it is unlikely that a DC voltage source used to model a PDN has a non-zero voltage.

Statements defining simple voltage-controlled voltage sources and voltage-controlled current sources

The statements defining these two types of components are:

```
E???    out_node_P  out_node_N  cntl_node_P  cntl_node_N  gain
G???    out_node_P  out_node_N  cntl_node_P  cntl_node_N  gain
```

E??? and G??? are the reference designators. out_node_P and out_node_N are the two output nodes and cntl_node_P and cntl_node_N are the two input controlling nodes. gain is the gain constant for these sources and it is defined in a format similar to the resistance of a resistor.

Statements defining simple current-controlled voltage sources and current-controlled current sources

The statements defining these two types of components are:

```
H???    out_node_P  out_node_N  V_cntl  gain
F???    out_node_P  out_node_N  V_cntl  gain
```

H??? and F??? are the reference designators. gain is the gain constant defined in a format similar to the resistance of a resistor. V_cntl is the reference designator of a DC voltage source defined in the same subcircuit.

Statement defining coupling coefficients between inductors

The statement defining the coupling coefficient between two inductors is:

```
K???    L_ind1  L_ind2  coeff
```

K??? is a reference designator and coeff is the coupling coefficient defined in a format similar to the resistance of a resistor. L_ind1 and L_ind2 are the reference designators of two different inductors defined in the same subcircuit.

Statement defining a voltage-controlled current source using a Laplace transfer function

The statement defining a voltage-controlled current source using a Laplace transfer function is:

G??? out_node_P out_node_N LAPLACE cntl_node_P cntl_node_N coeffs

G??? is the reference designator. out_node_P and out_node_N are the two output nodes, and cntl_node_P and cntl_node_N are the two controlling input nodes. coeffs is a set of coefficients with the numerator coefficients and the denominator coefficients separated by the forward slash (/) character and the forward slash character must be separated by whitespace characters. Only 1st-order and 2nd-order Laplace transfer functions are supported.

Example 2nd-order Laplace transfer function:

G34 ref 34 LAPLACE 35 ref -1.07037e+21 -2.01023e+10 / 3.59220e+21 1.02751e+11 1

In this case, the Laplace transfer function is

$$\frac{-1.07037\times10^{21} - 2.01023\times10^{10}\,s}{3.59220\times10^{21} + 1.02751\times10^{11}\,s + s^2}$$

The traditional line continuation for SPICE is also accepted. Hence, the above G34 could also be defined by the lines as follows:

```
G34  ref 34 LAPLACE 35 ref
+ -1.07037e+21
+ -2.01023e+10
+ /
+ 3.59220e+21
+ 1.02751e+11
+ 1
```

The lines above have been indented for improved visual appearance in this document. In the actual input file, the actual ('+') character for line continuation must be the first character on those lines in the input file.

Example 1st-order Laplace transfer function:

G53 ref 34 LAPLACE 47 ref 1.34726e+09 / 1.33368e+10 1

In this case, the Laplace transfer function is

$$\frac{1.34726\times10^{9}\,s}{1.33368\times10^{10} + s}$$

Nested subcircuits

The PDN parser does support nested subcircuit definition. To instantiate a subcircuit, the typical X statement of SPICE is supported:

```
X???  node1  node2  ...  name_of_subckt
```

X??? is the reference designator and must be unique within the subcircuit definition where this statement resides. node1 and node2 are node names and ... are more node names if appropriate. The last token is the name of subcircuit to instantiate. For better visual appearance, this statement can be broken into multiple physical lines by using the line continuation ('+') character.