Converting SPICE2 Polynomial Transfer Functions to Spice3 Equations

Mark Sitkowski
Design Simulation Systems Ltd
http://www.designsim.com.au

Introduction

It is not always practical to make simulation models of complex devices by reproducing every component used in their construction. Frequently, as in the case of an integrated circuit, it is impossible, since the characteristics of the components used are unknown, as is the schematic, defining their interconnections.

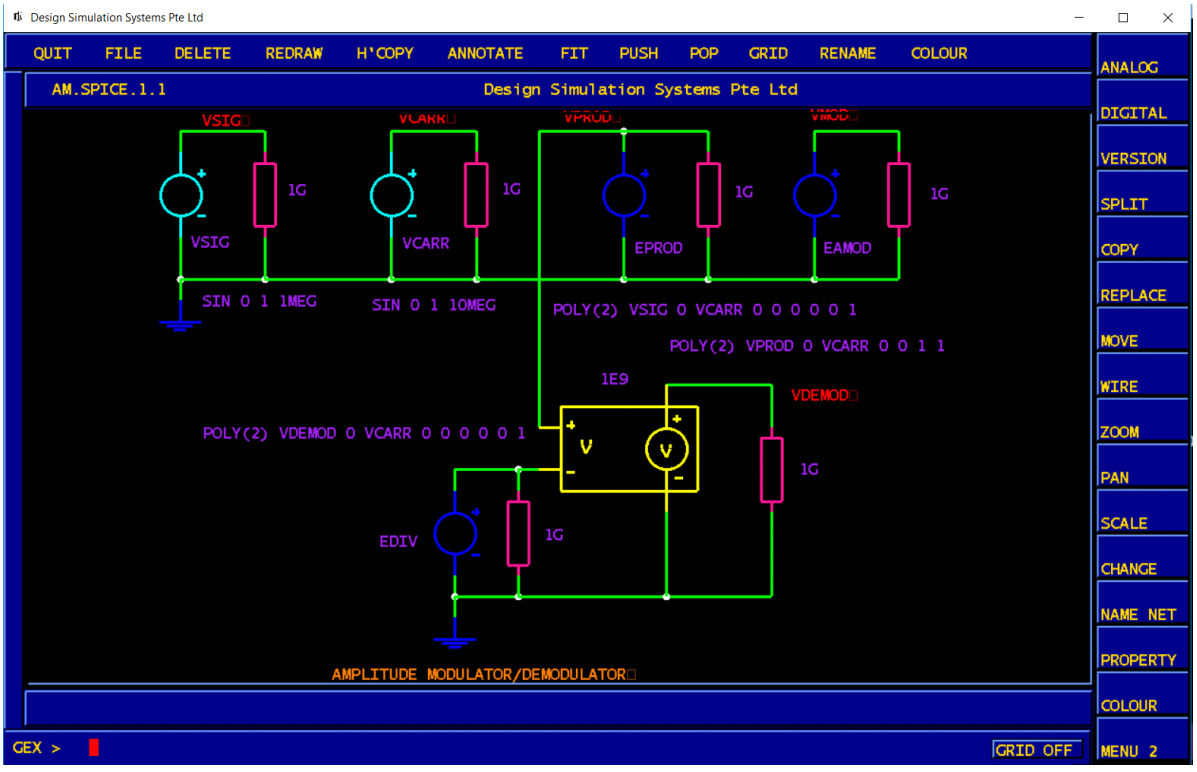
However, the black box parameters of the device can be known with sufficient accuracy, or may be measured.

Engineers working with SPICE2G6 can create behavioural simulation models by defining the transfer function of a non-linear voltage or current source as the sum of the terms of a multidimensional polynomial.
For example, the circuit shown below produces an amplitude-modulated signal, by first producing a suppressed-carrier wave, as per

M = (sinA * sinB)

Then, adds the carrier by calculating

N = (M + sinA)



Unfortunately, serious bugs, introduced into the 64-bit versions of the GNU g77 and gfortran compilers, have resulted in these compilers producing unusable executables of the SPICE2G6 simulator.

Migration to Spice3 is not directly possible, since Spice3 does not support the polynomial-defined non-linear sources, but chooses to define transfer functions by means of an equation where, for example, the SPICE2G6 definition

e1 4 0 poly(2) 2 0 3 0 0 0 0 0 1

is replaced by

B1 4 0 V=(V(2)*V(3))

Performing such replacements by hand for individual simulator netlists may be practical but, where the number of netlists is in the tens or, even, the hundreds, representing several years of work, the task becomes impractical – especially for high-order polynomials with many dimensions.

Automating the Conversion

Any automatic conversion tool would need to perform the following tasks:

- scan the netlist file
- extract any line with the ‘poly’ definition
- rename the source from E/F/G/Hxx to ‘Bxx’
- parse the polynomial and identify the function of each term
- weight and sum all of the non-zero terms of the polynomial
- replace the original line with the Spice3 line
- rewrite the netlist to a new file.
- possibly, create a drawing from the netlist.

Scanning the file and extracting the data are trivial tasks. The first real issue is to determine the higher terms of polynomials not covered by the limited examples shown in the SPICE2G6 manual.

The terms follow a simple pattern of

(a+b+c+...)^2 + (a+b+c+..)^3 + (a+b+c+..)^4 + ...

Where the terms in each expansion follow the order which would be produced by a manual expansion, with constants and duplicate terms omitted.

Rather than perform the expansions by hand, a fairly simple piece of code can generate the terms of any polynomial and, to avoid having to do this repeatedly, can place them into a lookup table, which is hardcoded into the tool.

The table for poly(2) starts like this:

term[0], ""
term[1], "V(a)"
term[2], "V(b)"
term[3], "V(a)^2"
term[4], "V(a)*V(b)"
term[5], "V(b)^2"
term[6], "V(a)^3"
term[7], "V(a)^2*V(b)"
term[8], "V(a)*V(b)^2"
term[9], "V(b)^3"
term[10], "V(a)^2*V(b)"
term[11], "V(a)*V(b)^2"
term[12], "V(b)^3"
term[13], "V(a)^4"
term[14], "V(a)^2*V(b)^2"
term[15], "V(a)*V(b)^3"

The first term of any SPICE polynomial definition is always a constant, and the next ‘n’ terms of an ‘n’ dimension polynomial are the ‘n’ input variables.

By arranging the terms in the order shown above, we could use the constant of the loop scanning the polynomial as an index into the lookup table, to extract the Spice3 equation. As each term was extracted, it was concatenated in a running sum with preceding terms, to form the final transfer function of the controlled source.

Due to the limitation of the number of characters permitted in a single line of text, and the fact that searching for continuation lines (beginning with a ‘+’) would represent more effort than we were prepared to put into this project, we decided to apply some limits.

1. We would only support up to 10 dimensions
2. We would only expand the expressions to the following numbers of terms

Dimension	No. of terms
1	unlimited
2	30
3	35
4	34
5	55
6	65
7	35
8	44
9	54
10	65

The final term was arbitrarily determined by its algebraic significance, or degree of usefulness.

While considering the polynomial's parser, we had to make allowance for the fact that SPICE2G6 permits each coefficient to be a digit or a floating-point number, expressed as 1000, 1000.0, 1e3 or 1k. Rather than unravel these, we decided to import them as literals, and warn the user that Spice3 won't accept the '1k' format inside an equation.
A minor complication was that SPICE2G6 permits input definitions with implicit polarity, such as:

g1 0 1 poly(3) (7 4) (9 0) (12 0) 0 17.05e-3 -0.01 0 0 0 1.0

where the first input has neither node referred to ground, but is the voltage between node 7 and node 4. This had to be translated as

B2 0 1 I=(0 +((V(7)-V(4)))*17.05e-3+(V(9))*-0.01+((V(7)-V(4))*V(9))*1.0)

Whereas
E6 39 0 POLY(1) 0 9 0 1E6

became

B6 39 0 V=(0 +(1e6*(v(0)-v(9)))^1)
We also decided not to interpret the meaning of 'x^1' on the assumption that this trivial calculation would not significantly affect the simulation run-time.

Converting the Amplitude Modulation Circuit

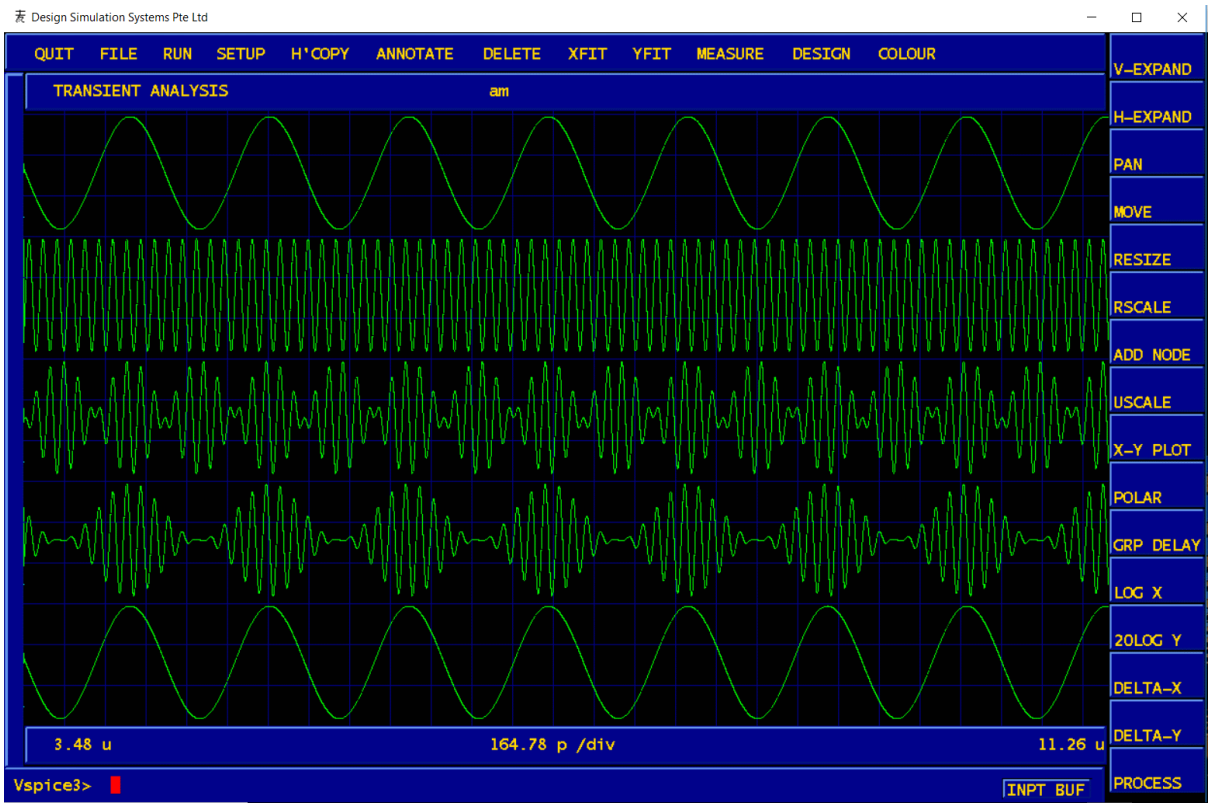
With reference to the circuit diagram shown on the first page, the compiler produced the following netlist:

```
am
.tran
.print
V1 2 0  SIN 0 1 1MEG
V2 3 0  SIN 0 1 10MEG
E1 4 0  POLY(2) 2 0 3 0 0 0 0 0 1
E2 5 0  POLY(2) 4 0 3 0 0 1 1
R1 2 0  1G
R2 3 0  1G
R3 4 0  1G
R4 5 0  1G
E3 7 0  POLY(2) 8 0 3 0 0 0 0 0 1
R5 7 0  1G
R6 8 0  1G
E4 8 0 4 7  1E9
.END
```

Running the netlist through our conversion tool, which had been imaginatively titled “Vpoly” gave us:

```
am
.tran
.print
v1 2 0  sin 0 1 1meg
v2 3 0  sin 0 1 10meg
B1 4 0 V=(0 +(V(2)*V(3))*1)
B2 5 0 V=(0 +(V(4))*1+(V(3))*1)
r1 2 0  1g
r2 3 0  1g
r3 4 0  1g
r4 5 0  1g
B3 7 0 V=(0 +(V(8)*V(3))*1)
r5 7 0  1g
r6 8 0  1g
e4 8 0 4 7  1e9
.END
```

This new netlist produced exactly the same simulation results as the original,



Completing the Picture

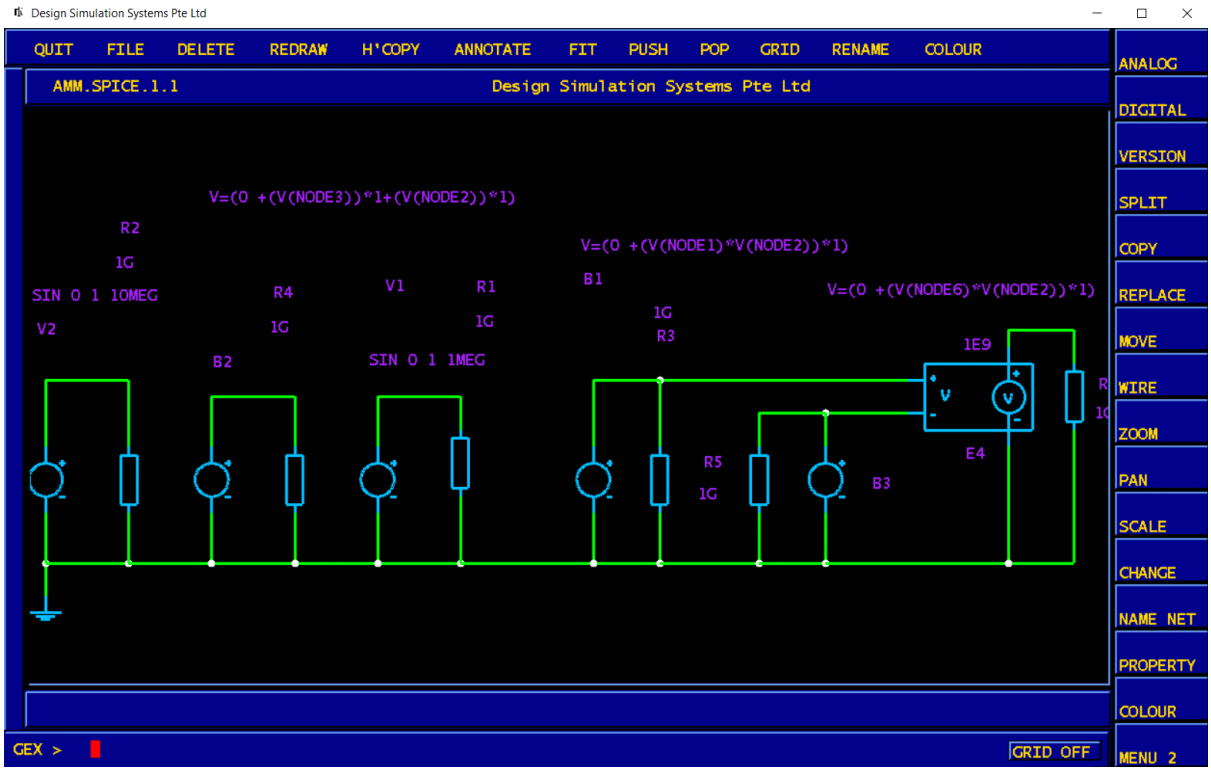
We felt there was still something else we could do.

In this case, we already had a circuit diagram, so the next step would have been, to simply edit the properties of the controlled sources, renaming them to ‘Bxx’ and replacing the ‘poly’ properties.

However, where there is no circuit diagram, it might be useful to create one from the new netlist. A tool called ‘net2gex’ achieves this, by creating a row-column layout, with point-to-point wiring, which it then optimises by changing the placement of the components for minimum crossover of connections.

The results are less than aesthetically-pleasing, as may be seen from the output, shown below. However, the interconnections are accurate, so some manual re-laying out of the circuit components, will produce an acceptable result.

It should be noted, that the operands of the equations contain node numbers inherited from the netlist, which do not correspond to any of the nets in this diagram. The first task, therefore, should be, to name all the nets in the circuit, and replace the operands with symbolic net names.

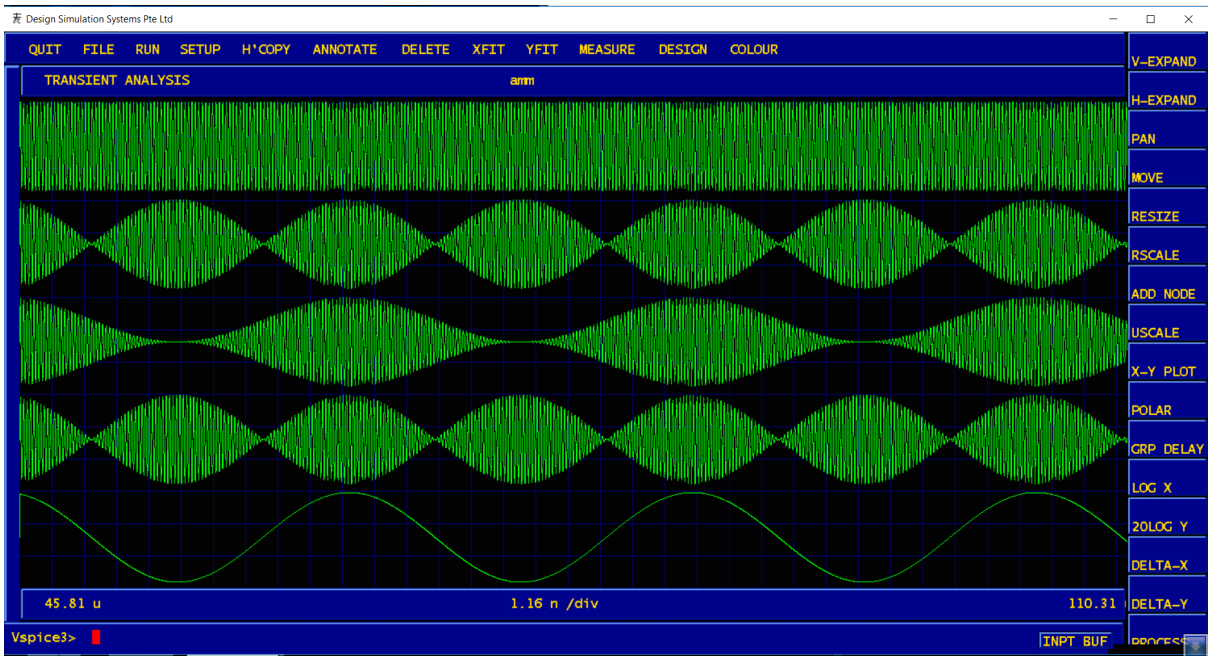


To test the connectivity of the newly-created circuit, we run it through the netlist compiler, to produce the netlist shown below, and add simulation directives.

```
amm
.options itl4=2
.tran 100n 1m 0.0000e+00 uic
.print tran  v(2) v(3) v(4) v(5) v(1)
E1 6 0 3 5  1E9
V1 2 0  SIN 0 1 5meg dc 1
B1 3 0  V=(0 +(V(1)*V(2))*1)
B2 4 0  V=(0 +(V(3))*1+(V(2))*1)
R1 1 0  1G
R2 2 0  1G
R3 3 0  1G
R4 4 0  1G
B3 5 0  V=(0 +(V(6)*V(2))*1)
R5 5 0  1G
R6 6 0  1G
V2 1 0  SIN 0 1 50k dc 0
.END
```

For obvious reasons, the nodes have all been renumbered, but the topology is still the same.

Finally, we run the simulation, and check the results:



Apart from a re-ordering of the waveforms, and a longer simulation period, the simulation results match those obtained with SPICE2G6.

LIMITATIONS

1. We don't do continuation lines. If it doesn't fit on one line (255 characters) you're on your own.
2. We only support the number of terms shown in the table of any polynomial

3. We only support up to 10 dimensions.
4. Your polynomial must not include abbreviations like ‘u’, ‘p’ ‘m’. Use ‘e-6’, ‘e-12’, ‘e-3’ etc. Spice3 does not understand such abbreviations in equations.
5. If your netlist contains the DOS line termination of ‘\r\n’ instead of ‘\n’, Spice3 will choke on it. You can remove the ‘\r’ like this:
   tr –d ‘\015’ < xxx.netlist > temp
   mv temp xxx.netlist