

# Listening to PSpice Simulations with LabVIEW\*

DALE H. LITWHILER

*Penn State University, Berks-Lehigh Valley College, Reading, PA 19610, USA. Email: dhl10@psu.edu*

*Engineering technology students at Penn State University are using LabVIEW Virtual Instruments (VI) to actually listen to the signals of PSpice simulations. A text parser VI is used to scan the PSpice simulation output file for the desired data. This text data is then formatted into the required audio format for playback via the computer's sound card. LabVIEW contains many functions to easily facilitate text file manipulation and sound card interfacing. By listening to various signals throughout the circuit, the students get an added dimension not typically experienced during circuit simulation. Students have developed a greater understanding of the operation of several circuits through the experience of being able to hear their circuit simulation results.*

## INTRODUCTION

AT PENN STATE UNIVERSITY, students in the electrical engineering technology (EET) program begin using the circuit simulation tool, PSpice (evaluation version), in their first semester. The students quickly become familiar with entering the circuit, setting up and running simulations and viewing and printing the results. Many of the circuits that are studied have application in the audio band of frequencies from 20 Hz to 20 kHz. Because the frequencies of interest in these circuits are in the audio band, an additional dimension can be added to the simulation experience by also *listening* to the signals [1–3]. Even the more experienced student PSpice users find this added capability very beneficial. Labview VIs have been developed to extract the signals from the PSpice output file and play the signals through the sound card of a personal computer (PC). The built-in functions of LabVIEW make the development of specific virtual instruments for academic laboratory use easy and intuitive [4].

An example circuit is presented and discussed in this paper. The example circuit is a passive crossover network that many of the students are familiar with through personal experience with car and home audio equipment. By varying the component values of this simple circuit, the students can hear dramatic changes in the frequency response. It also becomes quite obvious to the student when circuit errors have been made.

Simulations involving nonlinear audio circuits can provide a rich set of effects for the students to experiment with and experience. The concept of crossover distortion in class B power amplifiers is typically presented in analog electronics courses. Using the ideas presented in this paper, students can listen to the undesirable effects of crossover

distortion in a class B power amplifier and compare it to the much improved output signal obtained with a class AB amplifier. Hearing is believing.

## PSPICE SIMULATION SETUP

The PSpice *transient analysis* simulation is set up such that the output file (*filename.out*) contains the appropriate data in the format that is expected by the parser VI. The desired signals, sampling rate, and duration of the simulation are set up in the PSpice *Schematics* application [5].

The desired signal nodes are selected by placing a PRINT1 symbol on the schematic circuit nodes as shown in Fig. 1. Placing this symbol on the schematic causes the simulator to record the value of the voltage at this node at each of the time points as specified by the print step value of the transient analysis. Note that the default reference designation (REFDES) for each PRINT1 symbol rises incrementally as more symbols are placed on the schematic (PRINT1, PRINT2, etc.). The order in which the signals appear in the output file is determined by the order in which each symbol is *connected* to the circuit. This order can be changed by simply disconnecting the symbols and reconnecting them in the desired order of playback.

The sampling rate of the simulation signals is determined by the *Print Step* value in the transient analysis setup. The print step value is equal to the sampling period and must be set to correspond to a predefined standard audio sampling rate of 8.0 kHz, 11.025 kHz, 22.050 kHz, or 44.10 kHz. The actual VIs are designed to accept values that are within about one percent of these standard values to allow for truncation and rounding of significant digits without appreciable loss of playback accuracy.

The duration of the printed simulation sets the

\* Accepted 12 November 2004.

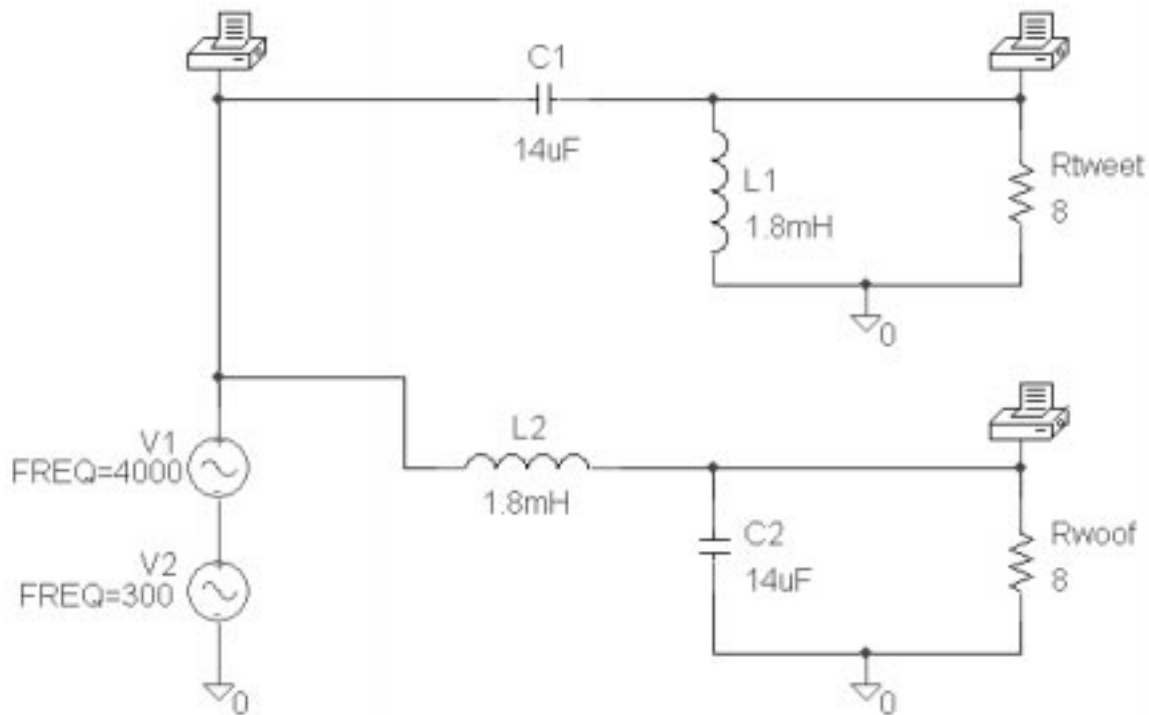


Fig. 1. PSpice schematic showing PRINT1 symbol placement.

length of the audio playback sound clip. A tradeoff must be made between memory usage, simulation time and recognizable playback quality. Saving the data with six significant digits results in about 30 bytes per data point in the PSpice output file. For example, if two signals are sampled at a rate of 22.050 kHz for a duration of three seconds, the resulting data file would be about 4 Mbytes in size. A sound duration of about three seconds per signal was found to be acceptable to the students. Figure 2 shows the transient analysis setup window for a sampling rate of 22.050 kHz and a duration of three seconds. Note that in this setup there is no print delay specified, so samples for the full simulation time will be present in the output file. Some circuits with long time constants may require an adjustment to the *No-Print Delay* as necessary to reach steady state.

It is also important that the data in the PSpice output file maintain enough significant digits to preserve as much fidelity as possible. The audio file playback will be performed with 16-bit precision, so the data in the PSpice output file should contain at least six significant digits. This value is set in the 'Options . . .' section of the PSpice analysis setup by the parameter called NUMDGT. The default value of NUMDGT is 4. Figure 3 shows the option setup window with NUMDGT being set to a value of 6.

### PSPICE OUTPUT FILE PARSER

The PSpice output file (*filename.out*) is generated and/or updated each time a new simulation is

performed. This ASCII text file contains all of the information regarding the circuit and simulation setup parameters as well as the results of the simulation. Scanning such files and extracting the desired information is performed by software referred to as a *parser*. The string functions contained in LabVIEW provide an excellent set of tools to quickly design and implement a parser.

Because the PSpice output file is very tightly structured and repeatable, the parser design is greatly simplified. Here the parser is designed to locate the node signal data in the output file and create a separate substring for each node present. The search is centered around locating the string 'TIME' which signals the beginning of each section of node data. These sections then end either with a page-break character ( $\backslash$ ) or the 'JOB CONCLUDED' string. Each node data section substring contains two columns of data; the first column contains the sample times and the second column contains the node voltage values. These substrings are then collected in an array of strings with each element of the array corresponding to one node.

An additional substring containing the PSpice node name for each of the selected nodes is also extracted by the parser. This string information is then used to indicate to the user which node signal is being played. The string containing the node name is located to the right of the 'TIME' string and is in parentheses. Figure 4 shows the LabVIEW diagram of the *PSpice Transient Output File Parser VI*.

Note that, if no properly formatted data is found in the selected file, a warning message

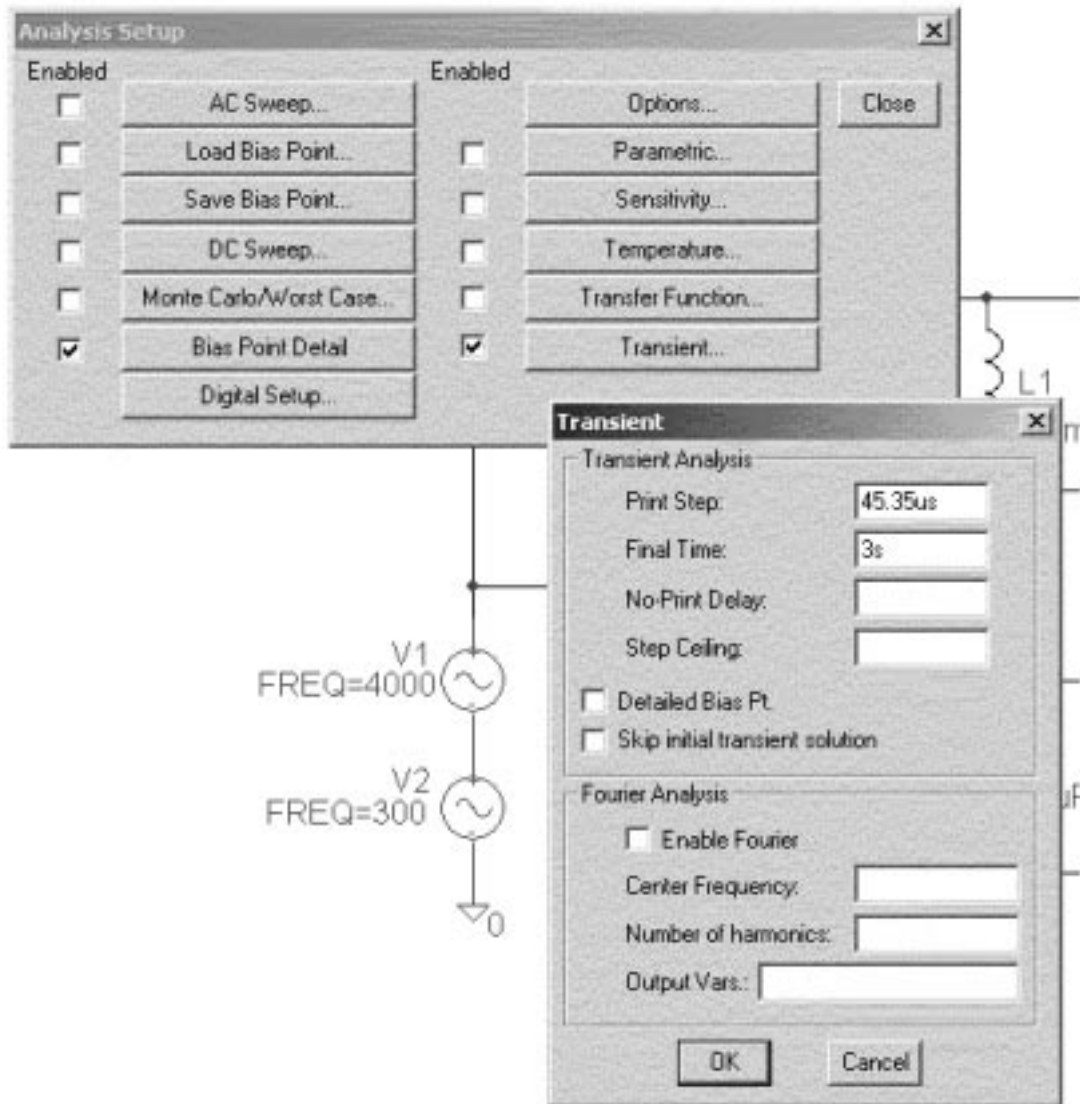


Fig. 2. PSpice transient analysis setup window.

dialog is generated and the error bit (Boolean) of the *error out* cluster is set *true*.

### STRING CONVERSION AND EVALUATION

The string data that is located by the parser VI must be converted to the proper numerical format for use by the .wav playback functions. It is also important to check the data for acceptable sampling time values to avoid sending useless signals to the sound card. These functions are performed by the *Extract Sample Rate and Y Values* and *Y Values to Normalized 16 Bit Integer Array* VIs.

Figure 5 shows the LabVIEW diagram of the *Extract Sample Rate and Y Values* VI. The while-loop section of this VI is similar to the *Extract Numbers* VI which is included with LabVIEW. The input string is scanned for numbers and other mathematical symbols commonly used to

represent numbers. When a group of numbers and symbols is located, it is placed as an element in a double-precision numerical array. With the parsed string node data as input, this produces a one-dimensional array with sample time and node voltage values (Y values) interlaced. This array is then decimated to produce two separate arrays containing the sample times and Y values.

The sampling time interval is determined by taking the difference between the first two values of the sample time array. This sampling time is then compared to the acceptable standard values. If the value is within the acceptable 1% range, the appropriate standard value is passed to the sound format cluster. If the value is unacceptable, a warning message dialog is generated and the error bit (Boolean) of the *error out* cluster is set *true*.

The 16-bit format of the audio signals for playback is required to be a signed integer ranging from -32768 to 32767. The double-precision Y values are converted to 16-bit signed integers by

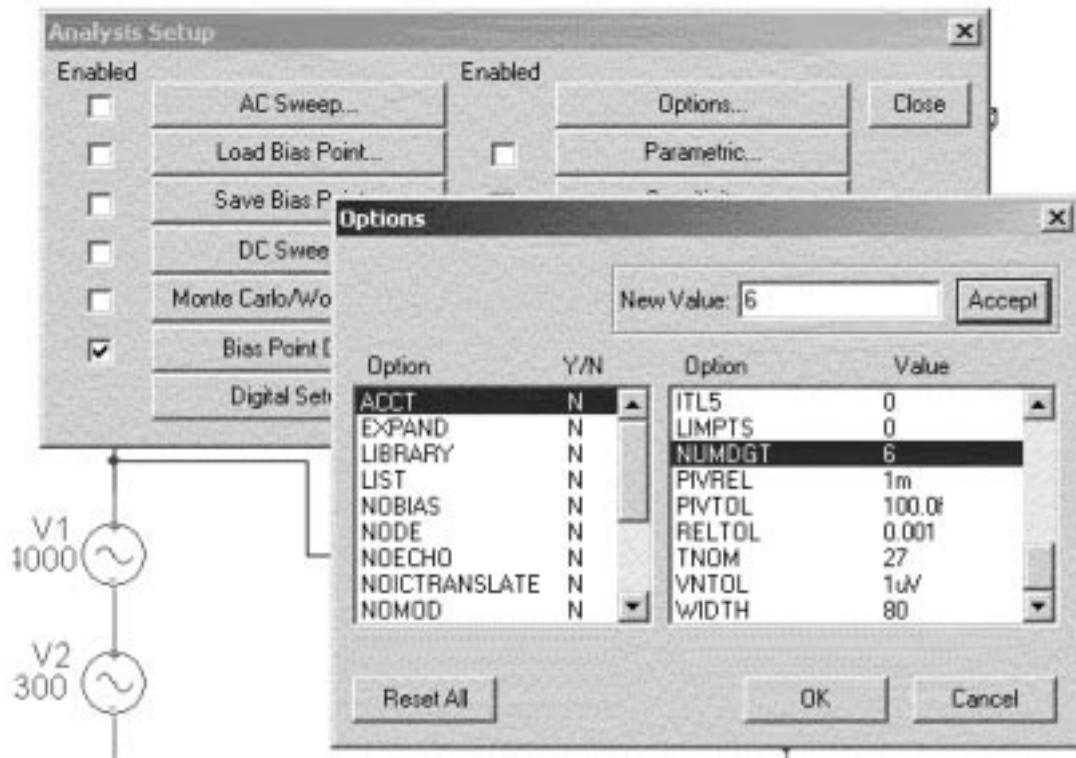


Fig. 3. PSpice analysis options setup window.

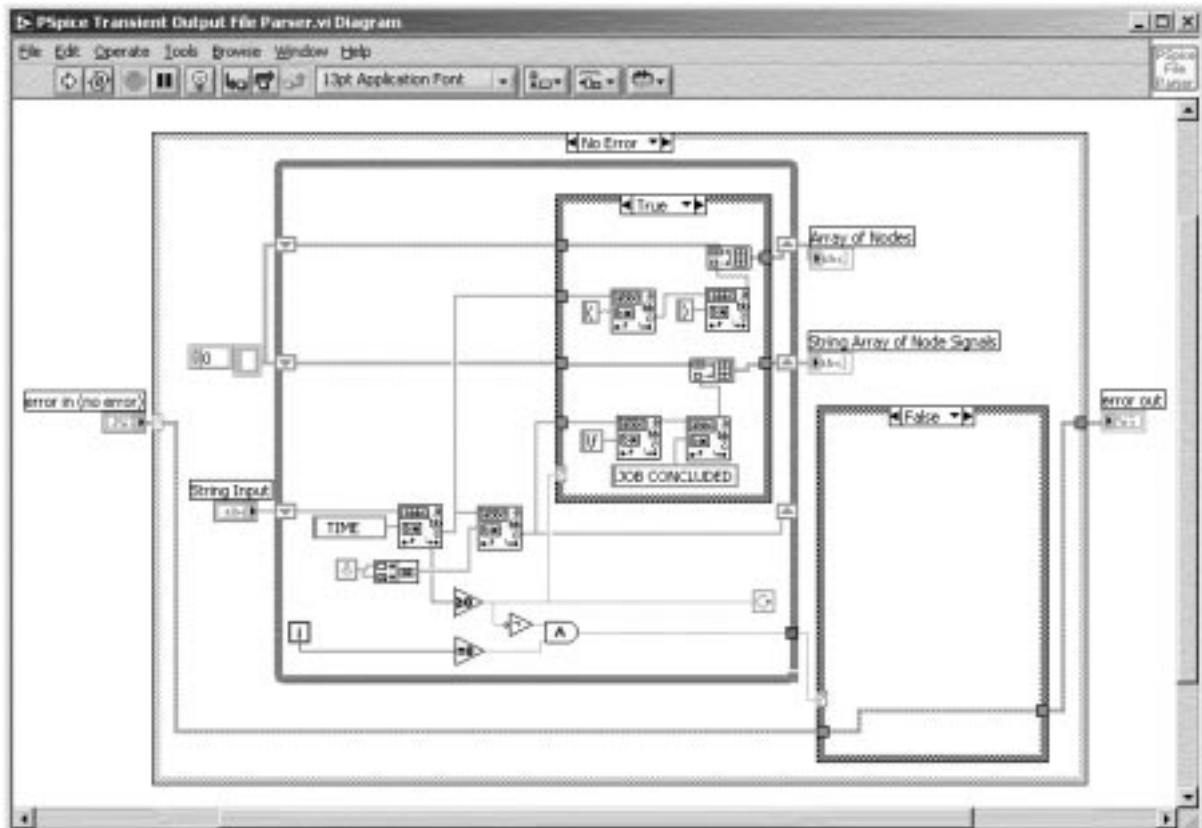


Fig. 4. LabVIEW diagram of the PSpice Transient Output File Parser VI.

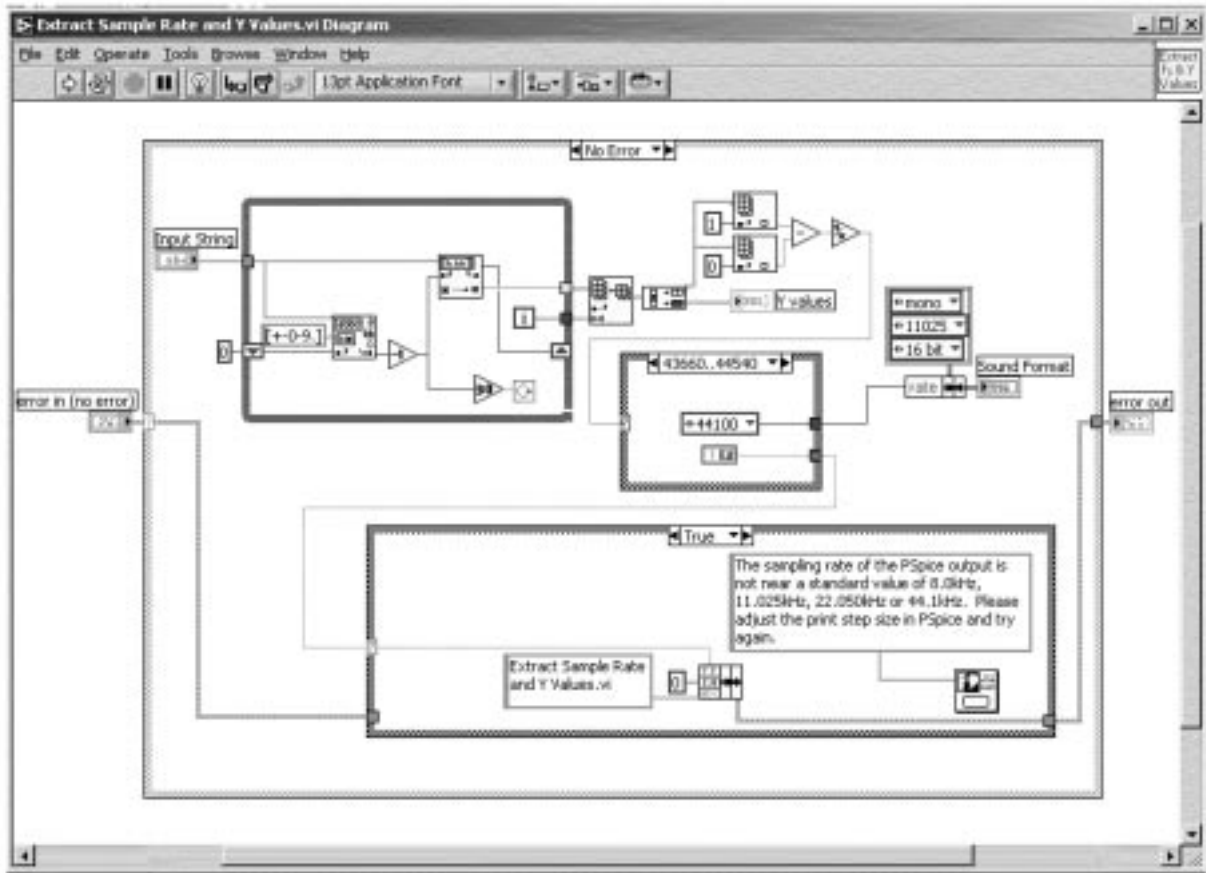


Fig. 5. LabVIEW diagram of the *Extract Sample Rate and Y Values* VI.

the *Y Values to Normalized 16 Bit Integer Array* VI. To avoid different volume levels of the various signals, each signal is normalized to its maximum value and scaled to use the maximum range of the signed integer. Figure 6 shows the LabVIEW diagram of the *Y Values to Normalized 16 Bit Integer Array* VI.

### SIGNAL SOUND PLAYBACK

The *PSpice Output to Sound Output* VI is the main VI that controls the execution of the aforementioned subVIs and provides a basic user interface. The front panel for this VI is shown in Fig. 7. The user simply clicks the *run* button to begin

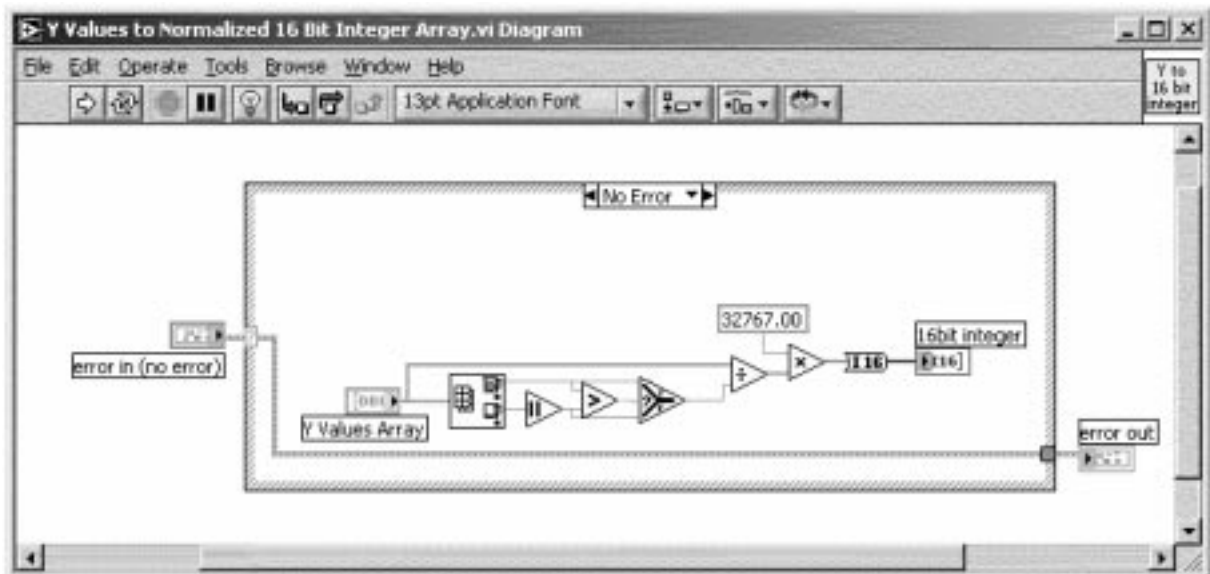


Fig. 6. LabVIEW diagram of the *Y Values to Normalized 16 Bit Integer Array* VI.



Fig. 7. LabVIEW front panel of the *PSpice Output to Sound Output* VI.

execution. The front panel string indicator displays the name of each node as its signal sound is played.

Figure 8 shows the diagram for the *PSpice Output to Sound Output* VI. When this VI is executed, the user is prompted to select an existing PSpice output file. The search is limited by default to files with the *.out* extension but the user can choose to view all files if desired. The user can also

cancel the search, which in turn bypasses the subsequent subVIs and ends the VI execution. When an output data file is selected, the filename is passed to the parser VI. The string array output of the parser VI is passed to the for-loop which is auto-indexed and is therefore executed once for each element of the string array.

Within the for-loop, the *Extract Sample Rate*

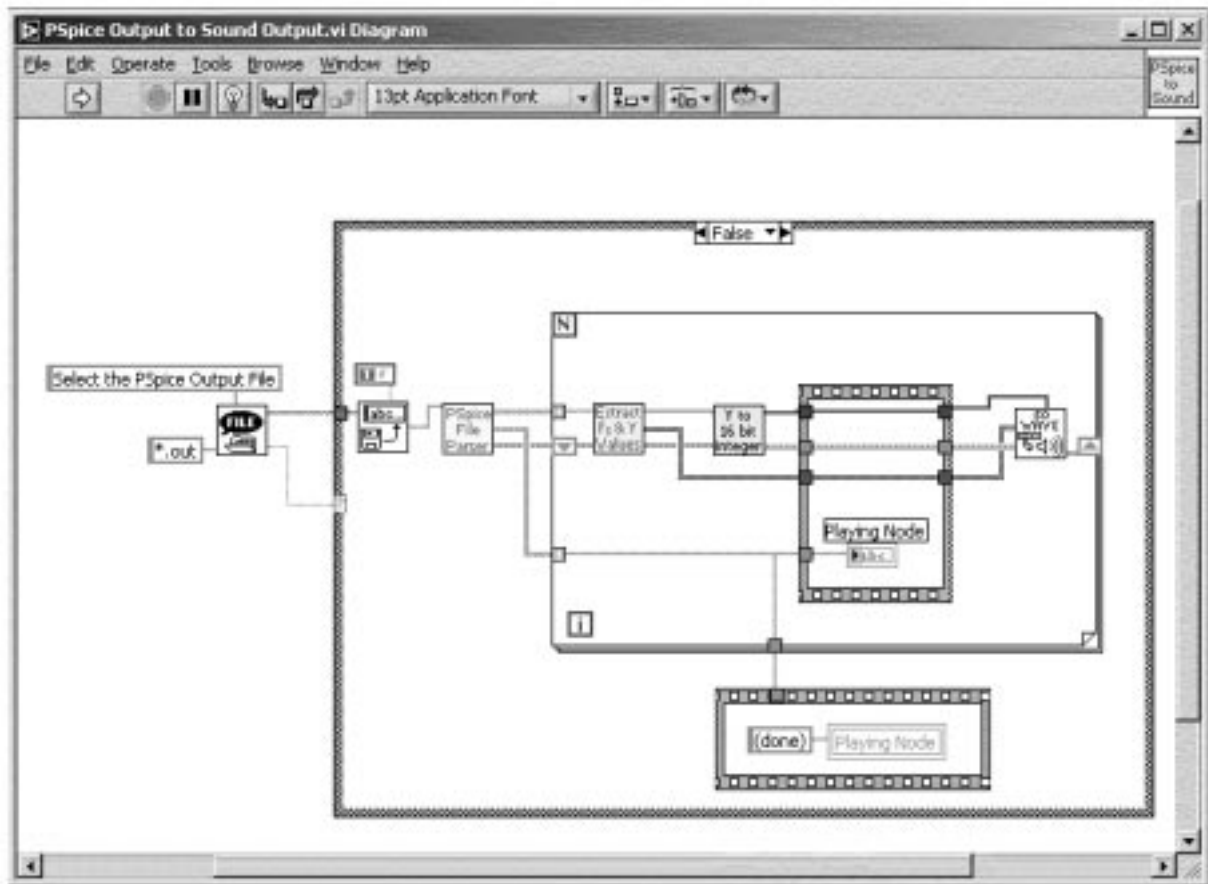


Fig. 8. LabVIEW diagram for the *PSpice Output to Sound Output* VI.

and *Y Values VI* and the *Y Values to Normalized 16 Bit Integer Array VI* convert the string data to 16-bit integer values for each node signal and pass it on to the *Snd Write Waveform VI* which drives the PC's sound card to produce the signal sound. Notice that if any errors occur in the parsing and data extraction and formatting subVIs, all subsequent subVIs do not execute and thus no erroneous output sounds are generated.

### EXAMPLE CIRCUIT

Consumer audio products provide a rich assortment of circuits that are both demonstrative of basic circuit principles and somewhat familiar to many engineering technology students. One such circuit is the passive crossover network found in high-fidelity audio loudspeakers. A 2-way crossover network consists of a second order L-C high-pass and second order L-C low-pass filter. The high-pass filter is used to pass the higher frequencies to the audio transducer (speaker) that is best suited for reproducing these frequencies: the 'tweeter.' Similarly, the low-pass filter passes the lower frequencies to the 'woofer.' The PSpice schematic for this circuit is that shown in Fig. 1.

In this example circuit, the cutoff frequency for both the high-pass and low-pass filters is set to 1 kHz by the choice of inductor and capacitor values. The two sine wave signal sources are set to audible frequencies below and above 1 kHz. The PRINT1 symbols are connected into the schematic in the following order: 1) Top of V1 node; 2) Top of Rwoof; 3) Top of Rtweet. By using this order,

the students first hear the two-toned source signal followed by the lower frequency 'woofer' tone and finally the higher frequency 'tweeter' tone.

### SUMMARY AND CONCLUSION

Adding the capability of actually *hearing* the signals present in a circuit simulation has added greatly to the learning experience of the electrical engineering technology students. Often the student is not sure exactly what the plotted signals of a circuit should look like but they are aware of how the signals should sound (or at least how they should *not* sound). They are then able to more quickly troubleshoot problems with the circuit diagram. The students are also more inclined to experiment (and actually have fun) with different component values to see what happens and thereby gain a more thorough understanding of how the circuit operates. Students comment that it almost becomes like a 'game' to see what happens if certain components are 'tweaked' in different ways and how well (or how poorly) the circuit's performance can be tuned. When the students start to see computer circuit simulation as an interactive, multi-media game, the PSpice assignments seem less mundane and laborious, thus more learning is accomplished.

As shown here, the software required to add this additional capability can easily be implemented with LabVIEW. Most PCs are now manufactured with integrated CD-quality sound cards, so the hardware is already in place to provide the audio user interface.

### REFERENCES

1. D. Mehrl and M. Hagler, Active learning using inexpensive sound cards for circuits and communications experiments, Proceedings of the 1998 Frontiers in Education Conference (1998).
2. A. Spanias *et al.*, Development of a web-based signal and speech processing laboratory for distance learning, *ASEE Computers in Education Journal*, **10(2)** (2000), pp. 21–26.
3. D. Millard, *et al.*, Interactive learning modules for electrical, computer and systems engineering, Proceedings of the 1997 ASEE/IEEE Frontiers in Education Conference (1997).
4. M. Higa, D. Tawy and M. Lord, An introduction to LabVIEW exercise for an electronics class, Proceedings of the 2002 ASEE/IEEE Frontiers in Education Conference (2002).
5. MicroSim Corp., *MicroSim PSpice A/D & Basics+ User's Guide* (1996).

**Dale H. Litwhiler** is currently Assistant Professor of Engineering Technology at Penn State Berks-Lehigh Valley College. He has over 17 years of industrial experience with IBM and Lockheed Martin, where he designed military power electronics hardware as well as spacecraft test hardware and software. He earned his B.Sc., M.Sc. and Ph.D. in electrical engineering from Penn State University, Syracuse University, and Lehigh University respectively. He is a licensed professional engineer in Pennsylvania, USA, and his research interests include data acquisition and analysis and sensor technology.