A Neural-Network Audio Synthesizer

Generating natural and space-age sounds in hardware

Mark Thorson, Forrest Warthman, and Mark Holler

Although neural networks got their start in software, the computation power required by applications such as control systems makes hardware implementations of neural nets a natural evolution. In fact, within the next decade we'll likely see more neural nets in hardware than software—and Intel's 80170NX Electrically Trainable Analog Neural Network (ETANN) chip is the first widely available silicon implementation of the technology.

This article describes an 80170NX-based musical instrument of unique design. The instrument, which synthesizes analog audio signals, evolved from a project begun in 1989 with David Tudor, a pioneering electronic-music composer and a musician with the Merce Cunningham Dance Company in New York. Tudor and his colleague, Takehisa Kosugi, introduced the synthesizer in a series of performances by the Merce Cunningham Dance Company at the Paris Opera House in November 1992.

Mark Thorson designed and implemented the synthesizer's primary hardware. He received his AB in neurobiology and is associate editor of Microprocessor Report. Forrest Warthman conceived the synthesizer project, designed the user interface, and maintains the project's momentum. He is president of Warthman Associates, Palo Alto, California. Mark Holler is Intel's program manager for neural-network products. The authors can be contacted at 240 Hamilton Ave., Palo Alto, CA 94301.
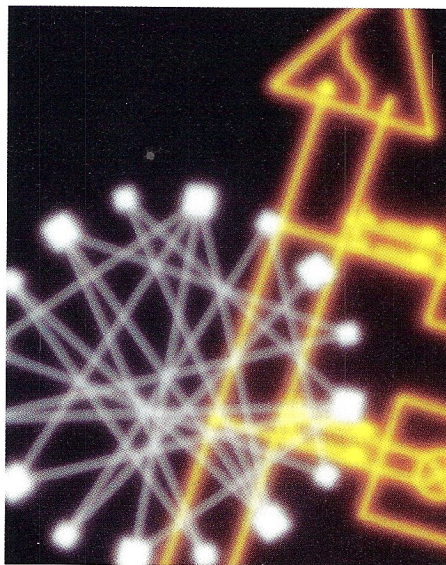
The synthesizer can generate a remarkable range of audio effects, from unique space-age and science-fiction sounds to passages that sound like heart beats, drums, gongs, porpoises, birds, engines, and musical instruments such as violas and flutes.

Sounds are generated internally by the synthesizer, without external inputs, using the neural-network chip's 64 artificial neurons. The neurons are connected on-chip in loops, using programmable synaptic weights, or off-chip, using patch cables and feedback circuits. Oscillations occur as a result of delay in the feedback paths. The sounds are generally rich because of the complexity of the circuitry. External inputs

such as voice, music, or random sounds can be used to enrich or control the internally generated sounds.

In this article, we present the design and implementation of the synthesizer—from circuits to firmware—as an example of a typical, hardware-based, neural-net embedded system. For background on neural networks, see the sources listed at the end of this article and "Untangling Neural Nets" by Jeannette Lawrence (DDJ, April 1990).

Synthesizer Architecture

The synthesizer's console housing has dozens of audio jacks buffered to the analog inputs and outputs of the neural-network chip; see Figure 1(a). Patch cables are routed to and from the jacks to feed chip outputs back to chip inputs; to connect external inputs to chip inputs; and to connect chip outputs to external amplifiers, recorders, or display devices. Some of the chip outputs have multiple console jacks so that a single neuron on the chip can drive several destinations.

The 80170NX is at the heart of the synthesizer; see Figure 1(b). The chip contains 64 artificial neurons, each with 128 analog inputs. Artificial "synapses" connect each neuron with the 128 inputs. Each synapse in an artificial neuron consists of a multiplier and a nonvolatile weight; see Figure 2. The function of a neuron is to sum the products of all inputs×weights (the "inner product" or "dot product" of the vectors) and output a result that is a sigmoid function of the inner product. The

sigmoid function has a nonlinear threshold shape, like a stretched out letter "S."

There are two 64×64 arrays of synapses—an input array and a feedback array. The input array is programmed with weights, and the results produced by the 64 neurons can optionally be fed back on-chip to the feedback array. This allows any neuron to be connected to any other neuron by programming weights at the appropriate synapses in the feedback array.

Chip outputs can also be fed back to inputs externally, through feedback circuits in the synthesizer (Figure 3), or they can be brought to the synthesizer's console to drive multiple audio and/or oscilloscope channels.

The music synthesizer is unique in that it relies heavily on feedback and the dynamics of the analog circuitry, rather than just the feed-forward computations of the artificial neurons. The behavior of the synthesizer can only be described by a set of coupled, nonlinear differential equations. It's not feasible to simulate circuits of this complexity on today's digital computers. Only by building the synthesizer could its behavior be discovered.

Synthesizer Circuitry

The synthesizer circuit (Figure 4) has simple buffer structures on all inputs and outputs of the neural-network chip. These buffers consist primarily of LM324 op-amps in a unity-gain configuration, wired as analog buffers. Their main purpose is to protect the expensive neural-network chip against damage from high-voltage signals and short-circuit loads. Each audio input has a 0.27μF capacitor in series, which strips any DC component from the input signal. A 100K resistor connected to the unity-gain op-amp supplies the DC offset of the signal. For protection against extreme inputs, the input section has heavy rectifier diodes to clamp signals more than a diode drop (0.7V) above Vcc or below ground. At worst, an inexpensive quad op-amp chip or a diode needs to be replaced if an errant signal appears.

The input section is also equipped with a passive network for adjusting the center voltage (the DC offset) of the analog inputs. The inputs are AC-coupled to nodes that are weakly coupled to a static voltage defined by a potentiometer on the front panel. We considered this feature important because the neuron amplifiers are only linear in a small range of input voltages. We feared that operation outside this range would cause distortion of external audio sources fed into the network. The potentiometer for controlling DC offset allows external signals to be centered on the "sweet spot" (the linear region) of the gain function.

Originally, there were four front-panel potentiometers for defining static voltage levels: In addition to the DC-offset control, there were three special-purpose controls for inputs to the neural-network chip. These controlled the gain of the neuron amplifiers, the input reference level (the zero level), and the output range. After some experience with the unit, the latter two signals were tied to static voltage levels (1.5V). The potentiometers are implemented as simple voltage dividers between Vcc and ground, with a capacitor for filtering out noise.

At first, we tied DC-offset control directly to the analog inputs through an array of 100K resistors. Later, we wired spare op-amps in as unity-gain buffers between the voltage divider and the resistors to prevent cross-coupling (leak-
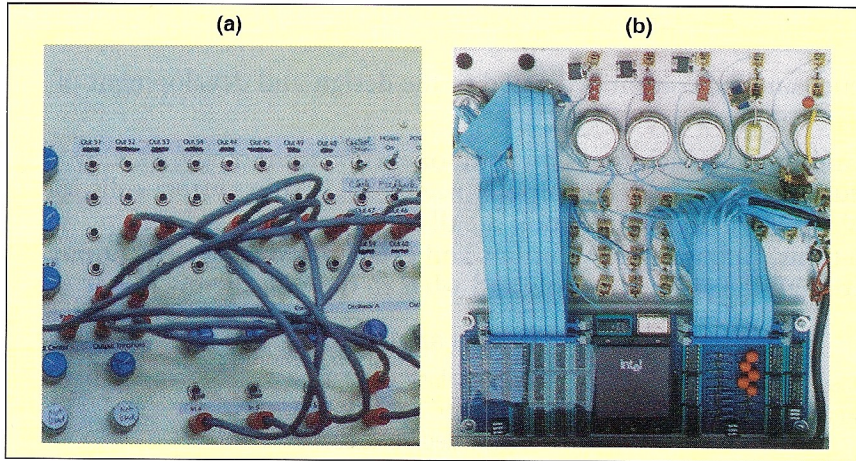

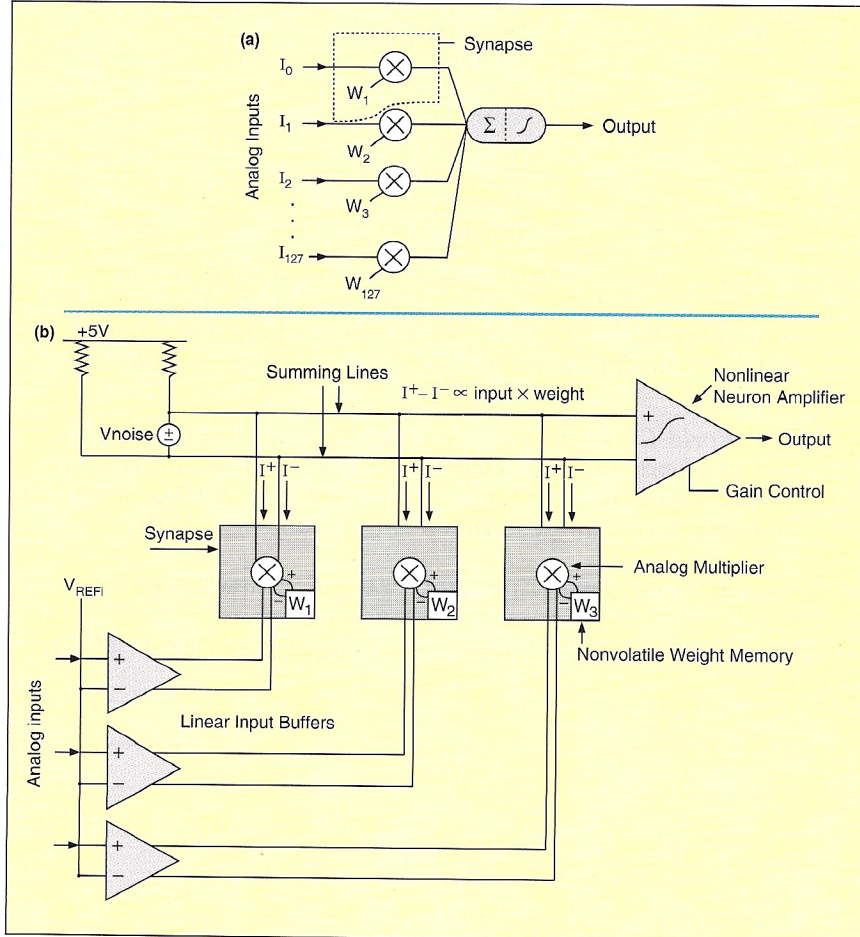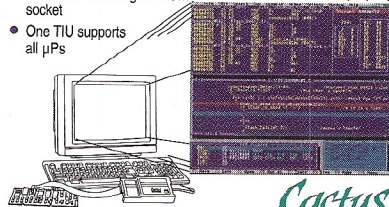**Figure 1:** Synthesizer console: (a) Top view; (b) bottom view.

Figure 2: Architecture of one of the 80170NX's neurons: (a) Neuron function; (b) electrical analog.

age of signals between audio-input channels).

Certain inputs were dedicated as control inputs, our intent being that one audio source could modulate another. However, there does not seem to be a simple way to make the neural-network chip do true modulation, because it only performs multiplication by constant coefficients stored on-chip. It can almost modulate one signal by another when the modulating signal is fed into the gain input—and we tried that—but there is significant feedthrough of the modulating signal to the output. Besides, this technique would only provide one modulation channel.

Instead, we performed a sort of on/off modulation in which control signals connected to the neurons by large synaptic weights are used to blot out the audio source by driving the neurons into saturation. Our first step was to implement a simple switcher, in which audio inputs could be routed to audio outputs under the control of inputs (control signals) that were themselves other audio sources. To get this to work nicely, we made three modifications to the input circuits for the control signals: The gain of the op-amps had to be increased from unity to about 1000; the signals had to be rectified; and heavy low-pass filtering had to be added. Depending on the potentiometer settings, the time constant of the low-pass filtering was about 0.1 to 0.5 seconds, chosen to correspond roughly to a spoken syllable or a musical note.
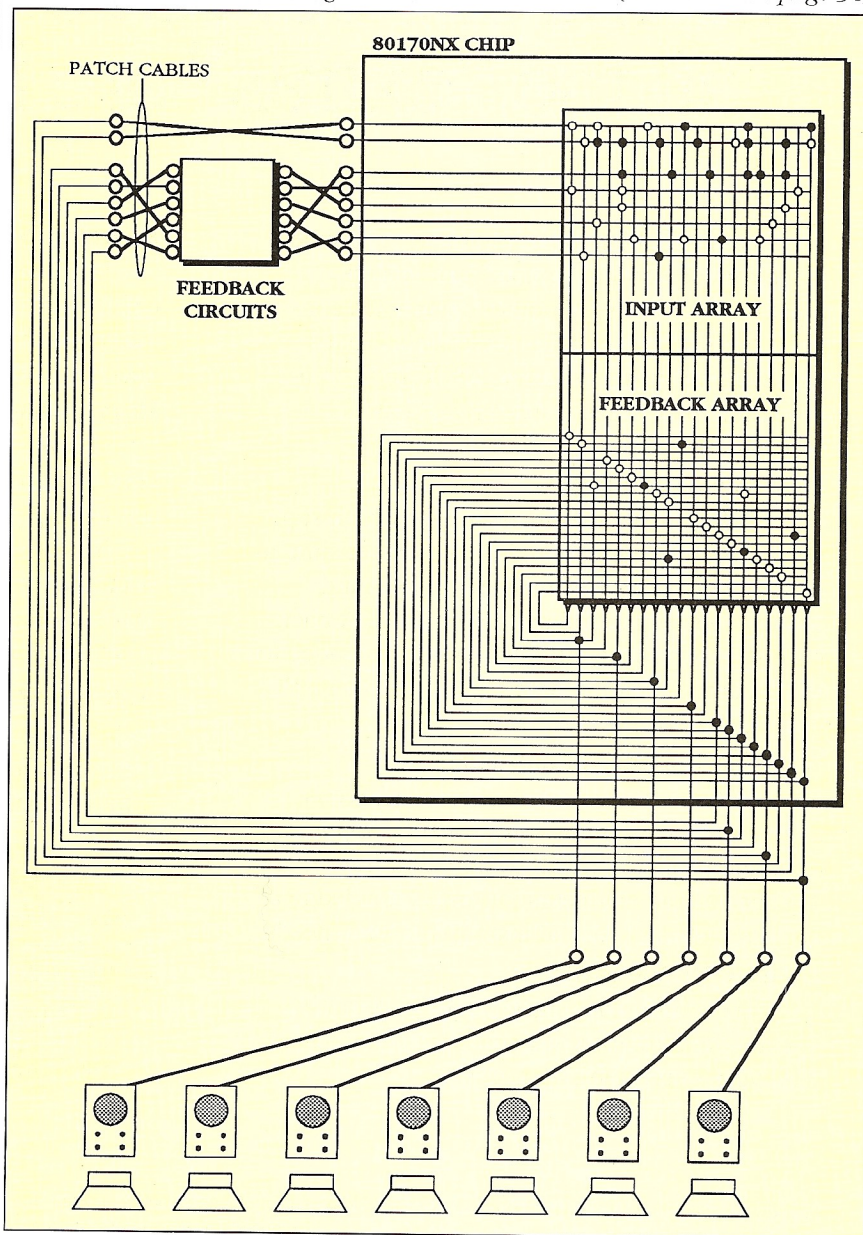
Figure 3: Synthesizer architecture.

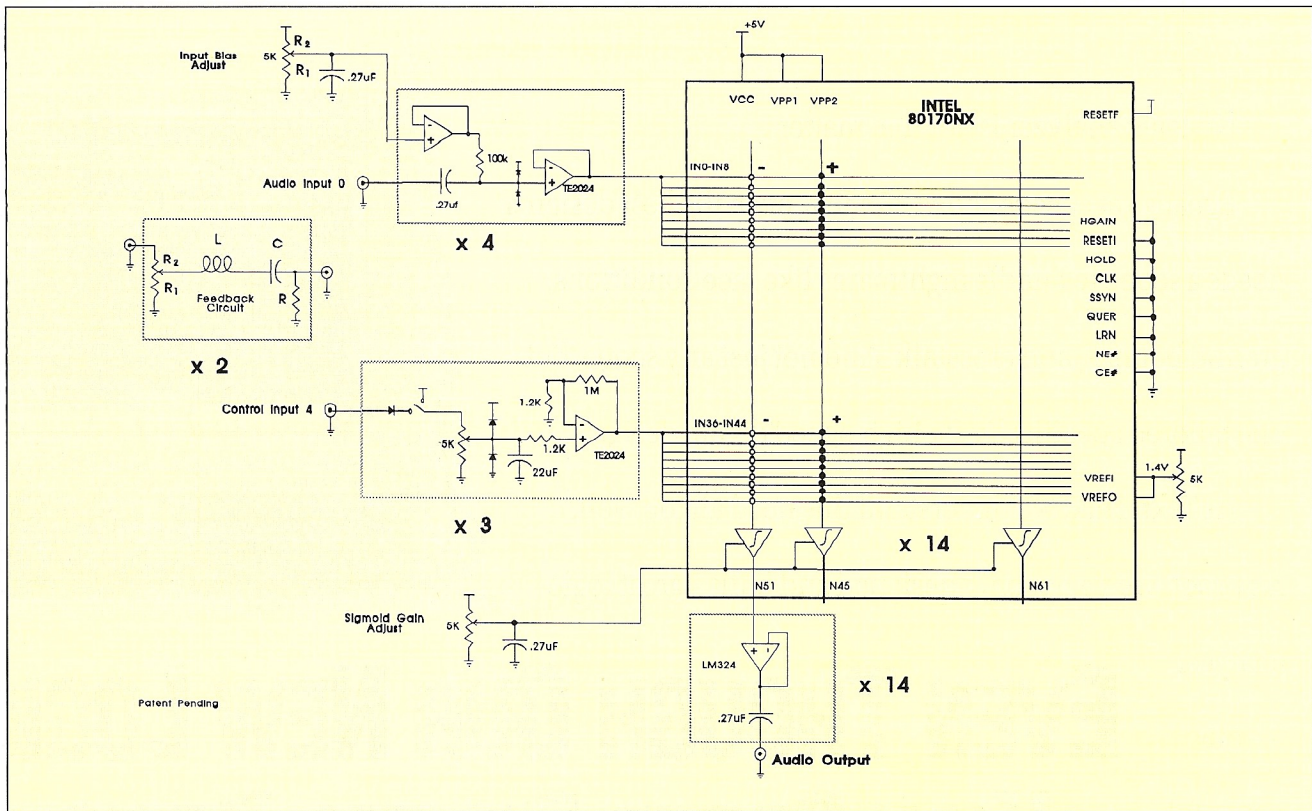**Figure 4:** *Synthesizer circuit schematic.*

(continued from page 55)

One of our first experiments allowed us to modulate music using a tape of a lecture. It resulted in the odd experience of hearing word-sized snatches of music with the cadence of speech— something like hearing a strange foreign language.

After we implemented the basic functions, we began exploring the capability of the system to generate sounds using feedback networks. During this time, the flexible construction of the unit proved invaluable. For example, the high-amplification factor on the control inputs was undesirable for oscillation experiments, and this could be easily changed by swapping a socketed resistor pack. We added DIP switches to allow the heavy filtering and rectification on the control inputs to be temporarily removed.

Noise was an early problem. Although the 50μV peak-to-peak noise on the summing lines was small, it was large enough to be annoying when the synthesizer was used to process external audio signals. We surmised that the cause was thermal noise on the neuron summing lines; see Figure 2(b). The amplification factor between these nodes and the neural-network outputs is a factor of about 1000. Our solution was simple and worked quite well: Because we were using a relatively small number of chip inputs, we could afford to run each audio signal into several input pins in parallel. With the same synapse weighting on each chip input for a given audio signal, the strength of the signal at the summing nodes is increased, while the noise level is unchanged. By using nine parallel chip inputs for each audio signal, the signal-to-noise ratio was improved by a factor of nine.

Noise is not always bad. It is useful during synthesis for adding randomness to the sounds. The neuron gain is set high to maximize amplification of the noise, and then feedback attenuation is adjusted until the network is just at the edge of oscillation. The noise intermittently stimulates oscillation of the network.

Firmware

The synthesizer's firmware consists of signed weights that represent the strength of connections between inputs and neurons. The weights are downloaded to the neural-network chip with the Intel Neural Network Training System (iNNTS), a software/hardware kit used to train the 80170NX. After downloading, the weights are analogous to the strength of synaptic connections in a biological neural network. Unlike typical neural networks, which use input/output pattern pairs and a learning algorithm to derive a set of weights, the synthesizer's weights are manually set to be unique for each neuron. Since our first goal was to synthesize original sounds, we did not use existing examples from which to learn.

Figure 5 shows an early version of this firmware. Here, six synthesizer inputs are shown as the rows, and the 14 neurons are shown as the columns. The weights are at the matrix intersections. The sigmoidal neuron amplifiers appear as triangles at the top of the matrix diagram. These represent the synthesizer outputs that can be routed back to inputs, to amplifier-speaker channels, and/or to oscilloscope channels.

In a later version of the firmware, two additional chip inputs, virtually all of the 64 neurons, and a large number of the chip's synapse connections were used to achieve greater complexity of sound.

Although the weights on the neural-network chip can be set with at least 6-bit precision, only values of +2.5V and −2.5V were used for the earliest version. The weights could have been changeable under the training system's control during audio synthesis if the synthesizer had been built on one of Intel's multichip prototyping boards. This approach, though more costly, would have facili-

tated easy reconfiguration and reduced the number of potentiometers and patch cables.

Inputs and Outputs

The first version of the synthesizer had seven inputs (four audio inputs and three control inputs), as shown in Figure 4. An input-bias adjust circuit on the audio inputs generates a DC voltage used to bias the inputs to the neural network in the middle of their operating range. The synapse multipliers are most linear when the inputs are near V_{REFi}, an input reference voltage supplied to the chip. V_{REFi} is supplied by another voltage divider shown on the right side of Figure 4.

The three high-gain control inputs are switchable between an audio source, shown in Figure 4 as an audio connector, and the +5V power supply. When a control input is connected to the power supply by the switch, the potentiometer associated with that input sets the control input to a static level. This ability to set individual input levels allows biasing of individual neurons at different operating points, some at high-gain and others saturated high or low.

A sigmoid-gain adjust is provided for all audio outputs as a group. This circuit adjusts the slope of the neuron output's threshold function. Finally, each audio output has a unity-gain op-amp for short-circuit protection and a decoupling capacitor.

Feedback

Two types of feedback are used to generate two different types of oscillations. The first type of feedback, used to synthesize sinusoidal oscillations, is generated by the phase-shifting bandpass-filter feedback circuit shown at the left side of Figure 4. This circuit can be patch-cabled between any audio output and any audio input. A potentiometer associated with the feedback circuit allows attenuation of the feedback signal; the more feedback, the larger the oscillations and the higher the frequency of the oscillations. The lower cut-off frequency of the bandpass filter is proportional to $1/RC$, and the upper cut-off frequency is proportional to R/L. The dominant R and C in the feedback path are actually the 100K resistor and the 0.27μF capacitor in the audio-input buffer circuitry.

The second type of feedback produces relaxation oscillations. It is accomplished by directly connecting audio outputs to audio inputs. The 100K resistor and the 0.27μF decoupling capacitors again are the dominant elements in

this oscillation circuit. The oscillations generated by this type of feedback are abrupt switching transitions followed by an RC decay back toward a switch point. The abrupt transition has the sound of a pop. Figure 6 shows the type of waveforms that can be produced. The waveforms generated are often similar to those of the action potentials or spikes in biological neurons.

Synthesizer Operation

The synthesizer is operated by configuring the cables (inputs, feedback loops, and outputs) and setting the potentiometers. First, the input-bias adjust potentiometer is set to bias the chip's neurons in the narrow region where they amplify linearly. The correct bias is detectable by listening for maximum noise output. Next, the Sigmoid gain is increased and feedback attenuation is reduced until the network breaks into oscillations. Changes in the relative gain of the various feedback paths or the network architecture produce different sounds.

In some configurations, the synthesizer generates predictable or semi-predictable rhythms, the periodicity and complexity of which can be varied. Some of these responses suggest the biological analogy of the circuit—the firing of neurons in an organic, not-quite-predictable sequence. In other configurations, the synthesizer generates remarkably complex and unique sounds that cannot be repeated predictably due to the high sensitivity of the oscillations to small changes in the feedback gain when the synthesizer is set just at the threshold of oscillation. At this bias point, very random behavior is often observed, much like the random ticks of a Geiger counter. This behavior is due to the thermal noise on the summing lines stimulating the network to oscillate for a few cycles, then dying out.
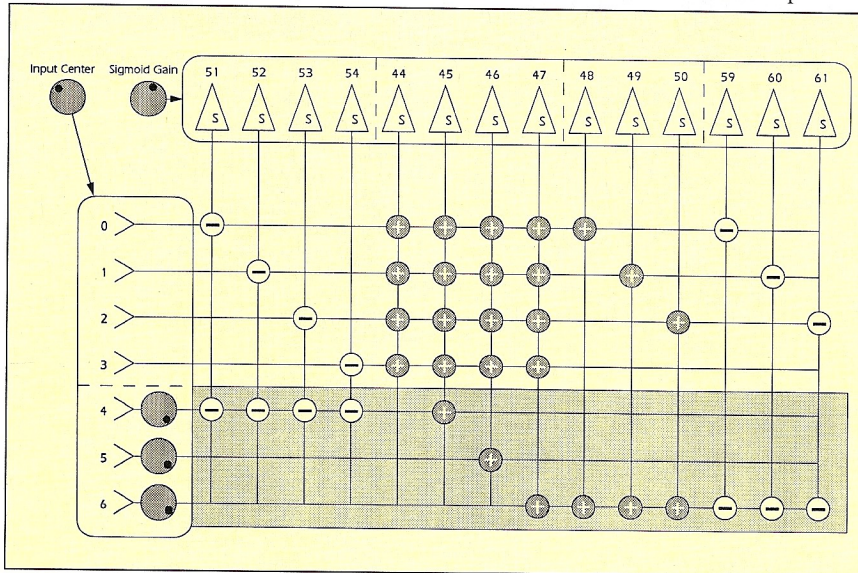
Figure 5: *Synthesizer firmware.*

Summary

The synthesizer provides unique insights into the dynamics of neural networks and complex nonlinear systems in general. These insights are novel because they're experienced in terms of audio and visual (oscilloscope) responses.

Potential near-term applications include musical instruments and controls for audio/visual entertainment perfor- mances. Long-range applications are an open frontier. Further development could result in products that respond to the unique pitch and volume of audio inputs with specific synthesized sounds. Development would likely include experimentation with larger networks and more complex feedback circuits. By making connections with weights on the neural-network chip rather than with patch cables, network architecture could be reconfigured in milliseconds under computer control. This approach would facilitate the modification of network architecture rhythmically, during synthesis.

References

80170NX Electrically Trainable Analog Neural Network (ETANN) Data Sheet. Santa Clara, CA: Intel Corp., 1991. Literature orders: 800-548-4725.

Hopfield J.J. and D.W. Tank. "Computing with Neural Circuits: A Model." *Science* (August, 1986).

Kandel, E.R. and J.H. Schwartz. *Principles of Neural Science,* second edition. New York, NY: Elsevier, 1985.

Mead, C. *Analog VLSI and Neural Systems.* Reading, MA: Addison-Wesley, 1989.

Rumelhart, D.E. and J.L. McClelland. *Parallel Distributed Processing Explorations in the Microstructure of Cognition,* volumes 1 through 3. Cambridge, MA: MIT Press, 1988.

Todd, P.M. and D.G. Log, eds. *Music and Connectionism.* Cambridge, MA: MIT Press, 1992.
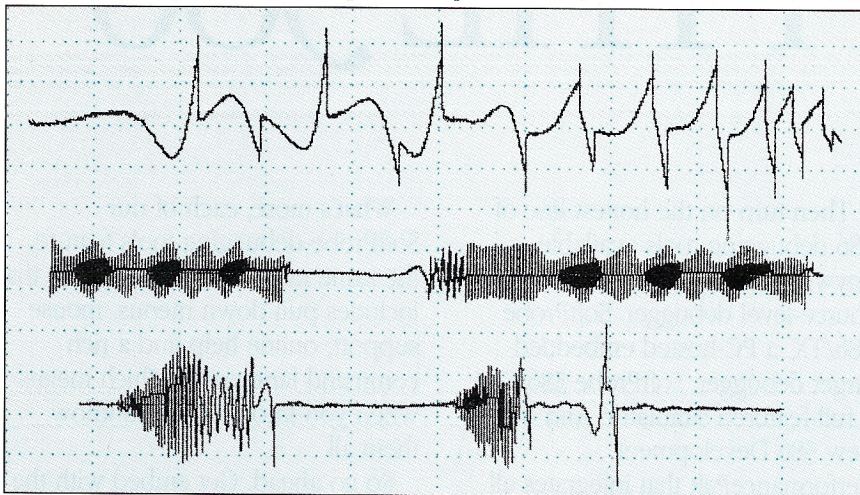
DDJ

Figure 6: *Sample waveform generated by the synthesizer.*