

CREATURES FROM PRIMORDIAL SILICON

Let Darwinism loose in an electronics lab and just watch what it creates. A lean, mean machine that nobody understands.

Clive Davidson reports:

"GO!" barks the researcher into the microphone. The oscilloscope in front of him displays a steady green line across the top of its screen. "Stop!" he says and the line immediately drops to the bottom.

Between the microphone and the oscilloscope is an electronic circuit that discriminates between the two words. It puts out 5 volts when it hears "go" and cuts off the signal when it hears "stop".

It is unremarkable that a microprocessor can perform such a task--except in this case. Even though the circuit consists of only a small number of basic components, the researcher, Adrian Thompson, does not know how it works. He can't ask the designer because there wasn't one. Instead, the circuit evolved from a "primordial soup" of silicon components guided by the principles of genetic variation and survival of the fittest.

Thompson's work is not aimless tinkering. His brand of evolution managed to construct a working circuit with fewer than one-tenth of the components that a human designer would have used. His experiments--which began four years ago and earned him his PhD--are already making waves. Chip manufacturers, robot makers and satellite builders are interested because the technique could produce smaller, more efficient devices than those designed today using traditional methods. Thompson's experiments have also inspired other research projects and some serious speculation about whether technology is poised to evolve in ways that will take it well beyond human understanding.

Looking for inspiration

Computer scientists have long looked to biology for inspiration. From simplified models of the brain they developed neural networks that have proved particularly good at recognising patterns such as signatures on credit cards and fingerprints. They have also worked out ways to mate and mutate programs and allow the resulting programs to compete with one another to generate the "fittest" software for a task. These "genetic algorithms" have been used to evolve software that does everything from creating works of art to selecting high-performing shares on the stock market.

To Thompson, who works with Phil Husbands at the Centre for Computational Neuroscience and Robotics at the University of Sussex, all these techniques leave something to be desired. They are too tightly constrained by the rules of chip designers and software engineers. The behaviour of living neurons, for example, is inseparable from the biochemicals from which they are made. But it doesn't matter what material the circuits of a neural network chip are etched in, so long as they operate in a digital fashion.

Digital computers break down all data into strings of 1s and 0s, which the hardware stores as "ons" and "offs" in its memory. This forces the transistors inside computer chips to work as switches--they're either on or off. But transistors are not intrinsically digital. Between on and off they pass through a smooth series of values, and in these regions they can behave as amplifiers, for example. Computer designers, however, make little or no use of these properties.

Likewise, programmers are constrained by the digital nature of computers. A program is a sequence of logic instructions that the computer applies to the 1s and 0s as they pass through its circuitry. So the evolution that is driven by genetic algorithms happens only in the virtual world of a programming language.

What would happen, Thompson asked, if it were possible to strip away the digital constraints and apply evolution directly to the hardware? Would evolution be able to exploit all the electronic properties of silicon components in the same way that it has exploited the biochemical structures of the organic world?

"I wanted to see what happens if you let evolution break out of the constraints that humans have," says Thompson. "If you give it some hardware, does it do new things?" These questions could only be answered if a way were found to combine the "wet" processes of biological evolution with the "dry" world of silicon chips. Thompson found the solution in a field-programmable gate array (FPGA).

The transistors in a conventional microprocessor are hardwired into logic gates, which carry out the processing. By contrast, the logic gates in an FPGA and their interconnections can be changed at will. The transistors are arranged into an array of "logic cells" and simply by loading a special program into the chip's configuration memory, circuit designers can turn each cell into any one of a number of logic gates, and connect it to any other cell. So by loading first one program, then another, the chip can be changed at a stroke from, say, an amplifier to a modem ("Software, who needs it?", New Scientist, 2 November 1996, p 41).

Mission impossible

Thompson realised that he could use a standard genetic algorithm to evolve a configuration program for an FPGA and then test each new circuit design immediately on the chip. He set the system a task that appeared impossible for a human designer. Using only 100 logic cells, evolution had to come up with a circuit that could discriminate between two tones, one at 1 kilohertz and the other at 10 kilohertz.

To kick off the experiment, Thompson created a population of 50 configuration programs on a computer, each consisting of a random string of 1s and 0s. The computer downloaded each program in turn to the FPGA to create its circuit and then played it the test tones (see Diagram, below). The genetic algorithm tested the fitness of each circuit by checking how well it discriminated between the tones. It looked for some characteristic that might prove useful in evolving a solution. At first, this was just an indication that the circuit's output was not completely random. In the first generation, the fittest individual was one with a steady 5-volt output no matter which audio tone it heard.

After testing the initial population, the genetic algorithm killed off the least fit individuals by deleting them and let the most fit produce copies of themselves--offspring. It mated some individuals, swapping sections of their code. Finally, the algorithm introduced a small number of mutations by randomly switching 1s and 0s within individual programs. It then downloaded the new population one at a time onto the FPGA and ran the fitness tests once more.

By generation 220, the fittest individual produced outputs almost identical to the inputs--two waveforms corresponding to 1 kilohertz and 10 kilohertz--but not yet the required steady output at 0 volts or 5 volts (see Diagram, below right). By generation 650, the output stayed mostly high for the 1 kilohertz input, although the 10 kilohertz input still produced a waveform. By

generation 1400, the output was mostly high for the first signal and mostly low for the second. By generation 2800, the fittest circuit was discriminating accurately between the two inputs, but there were still glitches in its output. These only disappeared completely at generation 4100. After this, there were no further changes.

Once the FPGA could discriminate between the two tones, it was fairly easy to continue the evolutionary process until the circuit could detect the more finely modulated differences between the spoken words "go" and "stop".

So how did evolution do it? If a human designer, steeped in digital lore, were to tackle the same problem, one component would have been essential--a clock. The transistors inside a chip need time to flip between on and off, so the clock is set to keep everything marching in step, ensuring that no transistor produces an output between 0 and 1. A human designer would also use the clock to count the number of ticks between the peaks of the waves of the input tones. There would be 10 times as many ticks between the wave peaks of the 1 kilohertz tone as those of the 10 kilohertz tone.

In order to ensure that his circuit came up with a unique result, Thompson deliberately left a clock out of the primordial soup of components from which the circuit evolved. Of course, a clock could have evolved. The simplest would probably be a "ring oscillator"--a circle of cells that change their output every time a signal passes through. It generates a sequence of 1s and 0s rather like the ticks of a clock. But Thompson reckoned that a ring oscillator was unlikely to evolve because it would need far more than the 100 cells available.

So how did evolution do it--and without a clock? When he looked at the final circuit, Thompson found the input signal routed through a complex assortment of feedback loops. He believes that these probably create modified and time-delayed versions of the signal that interfere with the original signal in a way that enables the circuit to discriminate between the two tones. "But really, I don't have the faintest idea how it works," he says.

One thing is certain: the FPGA is working in an analogue manner. Up until the final version, the circuits were producing analogue waveforms, not the neat digital outputs of 0 volts and 5 volts. Thompson says the feedback loops in the final circuit are unlikely to sustain the 0 and 1 logic levels of a digital circuit. "Evolution has been free to explore the full repertoire of behaviours available from the silicon resources," says Thompson.

That repertoire turns out to be more intriguing than Thompson could have imagined. Although the configuration program specified tasks for all 100 cells, it transpired that only 32 were essential to the circuit's operation. Thompson could bypass the other cells without affecting it. A further five cells appeared to serve no logical purpose at all--there was no route of connections by which they could influence the output. And yet if he disconnected them, the circuit stopped working.

It appears that evolution made use of some physical property of these cells--possibly a capacitive effect or electromagnetic inductance--to influence a signal passing nearby. Somehow, it seized on this subtle effect and incorporated it into the solution.

To solve this mystery, Thompson needs to measure the input and output values of each cell when the circuit is operating. But the FPGA allows only digital access to these points, so he can't measure the analogue values. Thompson's colleague, Paul Layzell, is building a circuit board that will allow all the components to be measured with analogue instruments.

However it works, Thompson's device is tailor-made for a single 10 by 10 array of logic cells. But how well would that design travel? To test this, Thompson downloaded the fittest configuration program onto another 10 by 10 array on the FPGA. The resulting circuit was unreliable. Another individual from the final generation of circuits did work, however. Thompson thinks it will be possible to evolve a circuit that uses the general characteristics of a brand of chip rather than relying on the quirks of a particular chip. He is now planning to see what happens when he evolves a circuit design that works on five different FPGAs.

Another challenge is to make the circuit work over a wide temperature range. On this score, the human digital scheme proves its worth. Conventional microprocessors typically work between -20 °C and 80 °C. Human designers set the clock so that chip components have enough time to settle into a digital value. As many computer hackers know, they can turn up the clock speed if they keep the temperature of the microprocessor low because the transistors settle into their on or off states more quickly when cold.

Thompson's evolved circuit only works over a 10 °C range--the temperature range in the laboratory during the experiment. This is probably because the temperature changes the capacitance, resistance or some other property of the circuit's components. Whatever the cause, this is a serious drawback. If the circuit needs a temperature controller to enable it to operate, then it is no longer a cheap, low-power device. But evolution could come to the rescue here as well. In a future genetic algorithm, Thompson plans to score circuits not only on how well they perform an electronic task, but also on how well they cope with temperature variation. Evolution might, for example, create a design that includes a set of subcircuits each of which operates over a different temperature range. If this fails to solve the problem, Thompson will try giving the FPGA a clock. But he won't tell the circuit what to do with it. "It will be a resource--we'll see what use evolution makes of it," he says.

Thompson's circuits have so far solved only simple problems. If they succeed at more complex tasks, they could prove useful for all kinds of applications. Thompson has evolved controllers for miniature robots for Xilinx, the Edinburgh firm that makes FPGAs. And the American company Motorola is showing interest in his ideas because they may mesh well with a new analogue FPGA the company has produced. British Telecom, which has an obvious interest in the sort of signal-processing problem that Thompson started with, is sponsoring work by Layzell, who is extending Thompson's ideas.

Suspicious minds

Already, at Napier University in Edinburgh, Julian Miller and Peter Thomson have picked up on Thompson's concept and are evolving their own digital circuits. They do this at a slightly higher level than Thompson, by creating lists of logic gates and connections, and putting evolution to work on these lists. They've evolved simple arithmetic units such as a multiplier. "It uses a lot fewer resources than a human would design," says Thomson.

If evolutionary design fulfils its promise, we could soon be using circuits that work in ways we don't understand. And some see this as a drawback. "I can see engineers in industry who won't trust these devices," says Thomson. "Because they can't explain how these things work, they might be suspicious of them."

If the chips ever make their way into engine control systems or medical equipment we could well face an ethical dilemma, says Inman Harvey, head of the Centre for Computational Neuroscience

and Robotics. "How acceptable is a safety-critical component of a system if it has been artificially evolved and nobody knows how it works?" he asks. "Will an expert in a white coat give a guarantee? And who can be sued if it fails?"

This is only a problem for people who don't understand how today's microprocessors are tested, says Pierre Marchal, who leads research into new computer architectures at the Swiss Centre for Electronics and Microtechnology in Neuchâtel. "I have no problem with this," he says. "You never test every possibility inside a microprocessor." That is why the bug in Intel's Pentium chip was found only a year after the first one was made.

Harvey and Marchal agree that the safety of future chips will have to be assured through exhaustive testing. If the chip operates properly under all the likely combinations of inputs and environmental conditions, then it doesn't matter how the chip works internally. The great thing about Thompson's idea, says Marchal, is that if you find a problem you add another constraint to the fitness test and evolve a better solution. "You can adapt it, just as the immune system adapts to new diseases," he says.

Marchal believes there is a "real possibility" that machines will evolve in ways that will be beyond human reasoning. To some, this prospect is frightening. But not to Marchal. "I'm not sure this is a real problem. The risk from a bomb is higher," he says. "Humanity can destroy itself far easier than these alien technologies."

For the moment, though, the thinking is more down to earth. At Napier, Thomson and Miller hope that evolution will teach them new design tricks. "It gives us a new way of looking at things," says Thomson. And at Sussex, Adrian Thompson has his own goal. "I'm just trying to explore what evolution will do."

Perhaps this is where the real value of his work lies. Whether or not his approach produces useful devices, it may help us to understand more about how the evolutionary process itself works. But that's another story.

Further reading: A collection of Adrian Thompson's papers is posted on his Web site at <http://www.cogs.susx.ac.uk/users/adrianth/ade.html>

Clive Davidson is a freelance journalist specialising in new applications for computers