An evolved circuit, intrinsic in silicon, entwined with physics.

Adrian Thompson*

COGS, University of Sussex, Brighton, BN1 9QH, UK

Abstract. 'Intrinsic' Hardware Evolution is the use of artificial evolution — such as a Genetic Algorithm — to design an electronic circuit automatically, where each fitness evaluation is the measurement of a circuit's performance when physically instantiated in a real reconfigurable VLSI chip. This paper makes a detailed case-study of the first such application of evolution directly to the configuration of a Field Programmable Gate Array (FPGA). Evolution is allowed to explore beyond the scope of conventional design methods, resulting in a highly efficient circuit with a richer structure and dynamics and a greater respect for the natural properties of the implementation medium than is usual. The application is a simple, but not toy, problem: a tone-discrimination task. Practical details are considered throughout.

1 Introduction

This paper describes a case-study in intrinsic hardware evolution: the use of artificial evolution — such as a Genetic Algorithm — to design a circuit automatically, where each fitness evaluation is the measurement of a circuit's performance when physically instantiated in a real reconfigurable VLSI chip. The term 'intrinsic' is used simply to indicate that the circuits are always tried out 'for real' rather than in simulation [1]. However, my dictionary also gives the following meanings to the word: *genuine, inherent, belonging to the point at issue.* I suggest that the point at issue with intrinsic hardware evolution is to allow the genuine inherent physical behaviour of the silicon be used freely, rather than just using hardware as a fast implementation of an idealised simulation or designer's model. I aim to show this through an example.

The following sections consider the first ever [13] intrinsically evolved FPGA configuration in great detail — there are interesting issues at every turn. The results speak for themselves, so I will save until later the underlying theory which motivated the rather unconventional approach taken. Then, in an extended discussion section, these ideas will be portrayed in the light of the experimental results that demonstrate their significance.

---

* Email: adrianth@cogs.susx.ac.uk
WWW: http://www.cogs.susx.ac.uk/users/adrianth/

2   The Evolvable Hardware

The Xilinx XC6216 [15] Field Programmable Gate Array (FPGA) [11] is a re-configurable VLSI chip particularly suitable for evolutionary work. It will soon be commercially available — the work reported here was carried out on a β-test version. A simplified representation of the device is shown in Fig. 1. It has
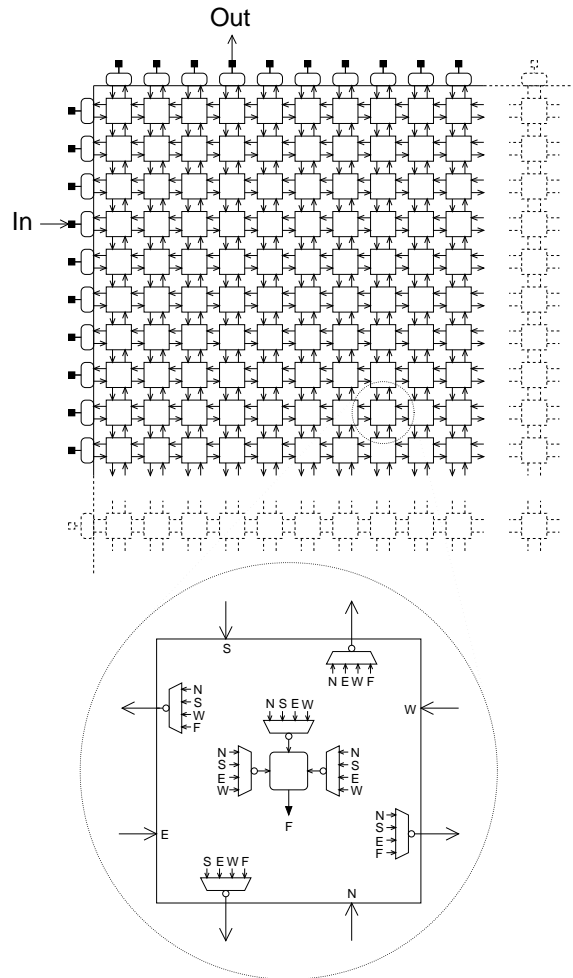

Fig. 1. A simplified view of the XC6216 FPGA. Only those features used in the experiment are shown. **Top:** A 10×10 corner of the 64×64 array of cells; **Below:** the internals of an individual cell, showing the function unit at its centre. The symbol ⌐°⌐ represents a multiplexer — which of its four inputs is connected to the output (via an inversion) is controlled by the configuration memory. Similar multiplexers are used to implement the user-configurable function F.

an array of 64×64 reconfigurable cells, each of which is connected to its four neighbours: North, East, West and South (NEWS) as shown. There is also a hierarchical arrangement of wires spanning 4, 16 and 64 cells, but these were not used in this experiment. Each cell contains a function unit that can be configured to perform any boolean function of two inputs, or multiplexer functions of three inputs. Each of a function unit's three inputs (not all of which are necessarily used) can be configured to be sourced by any of the four NEWS neighbours. The output of a cell in each of the NEWS directions can be configured to be driven either by the output F of its function unit, or by the signal arriving at one of the other NEWS faces. This allows a cell to connect some of its NEWS neighbours directly together at the same time as performing a function; a cell can 'route across itself' in some directions while giving the output of function F in others. The cells are configured independently (they do not all perform the same function), so even using only the nearest-neighbour links a very large range of possible circuits can be implemented.

Around the periphery of the array of cells are Input/Output Blocks (IOBs) and pads that interface the signals at the edge of the array to the pins of the chip. This is done in a more complex and flexible way than shown in the figure; all that is important here is that the chip was configured with a single input and a single output as shown. The choice of input and output positions was made before the experiment started, and then kept fixed. The unused IOBs simply appeared as inputs of a constant value. Only a 10×10 corner of the chip was used, and the unused cells were also configured just to produce a constant value. There are numerous other features of the device that were not used, and have not been mentioned.

At any time, the configuration of the chip is determined by the bits held in an on-chip memory, which can be written from software running on a host computer. No configuration of the cells can cause the device to be damaged — it is impossible to connect two outputs together, for instance, because all internal connections are uni-directional. So an evolutionary algorithm can be allowed to manipulate the configuration of the real chip without the need for legality constraints or checking. Here, we directly encode the configuration bits for the 10×10 corner — determining how the four outputs of each cell are derived and what function is performed by each function unit — onto a linear bit-string genotype of length 1800 bits. This was done in a raster fashion, reading cell-by-cell from left to right along each row, and taking the rows from bottom to top.

3 The Experiment

The task was to evolve a circuit — a configuration of the 10×10 corner of the FPGA — to discriminate between square waves of 1kHz and 10kHz presented at the input. Ideally, the output should go to +5V as soon as one of the frequencies is present, and 0V for the other one. The task was intended as a first step into the domains of pattern recognition and signal processing, rather than being an

application in itself. One could imagine, however, such a circuit being used to demodulate frequency-modulated binary data received over a telephone line.

It might be thought that this task is trivially easy. So it would be, if the circuit had access to a clock or external resources such as RC time-constants by which the period of the input could be timed or filtered. It had not. Evolution was required to produce a configuration of the array of 100 logic cells to discriminate between input periods *five orders of magnitude* longer than the input $\Rightarrow$ output propagation time of each cell (which is just a few nanoseconds). No clock, and no off-chip components could be used: a continuous-time recurrent arrangement of the 100 cells had to be found which could perform the task entirely on-chip. Many people thought this would not be possible.

The evolutionary algorithm was basically a conventional generational Genetic Algorithm (GA) [3]. The population of size 50 was initialised by generating fifty random strings of 1800 bits each. After evaluation of each individual on the real FPGA, the next generation was formed by first copying over the single fittest individual unchanged (elitism); the remaining 49 members were derived from parents chosen through linear rank-based selection, in which the fittest individual of the current generation had an expectation of twice as many offspring as the median-ranked individual. The probability of single-point crossover was 0.7, and the per-bit mutation probability was set such that the expected number of mutations per genotype was 2.7. This mutation rate was arrived at in accordance with the Species Adaptation Genetic Algorithm (SAGA) theory of Harvey [4], along with a little experimentation.

The GA was run on a normal desktop PC interfaced to some simple in-house electronics[2] as shown in Fig. 2. To evaluate the fitness of an individual, the hardware-reset signal of the FPGA was first momentarily asserted to make certain that any internal conditions arising from previous evaluations were removed. Then the 1800 bits of the genotype were used to configure the 10×10 corner of the FPGA as described in the previous section, and the FPGA was enabled. At this stage, there now exists on the chip a genetically specified circuit behaving in real-time according to semiconductor physics.

---

[2] TECHNICAL ELECTRONICS NOTES: The FPGA and its interface to the PC, the tone generator, and the analogue integrator all reside comfortably on a single full-length card plugging into the AT (ISA) Bus of the PC. The analogue integrator was of the basic op-amp/resistor/capacitor type, with a MOSFET to reset it to zero [7]. A MC68HC11A0 micro-controller operated this reset signal (and that of the FPGA), and performed 8-bit A/D conversion on the integrator output. A final accuracy of 16 bits in the integrator reading was obtained by summing (in software) the result of integration over 256 sub-intervals, with an A/D conversion followed by a resetting of the analogue integrator performed after each sub-interval. The same micro-controller was responsible for the generation of the tone.

Locations in the configuration memory of the FPGA and in the dual-port RAM used by the the micro-controller could be read and written by the PC via some registers mapped into the AT-Bus I/O space. The XC6216 requires some small but non-trivial circuitry to allow this — schematics are available from the author.
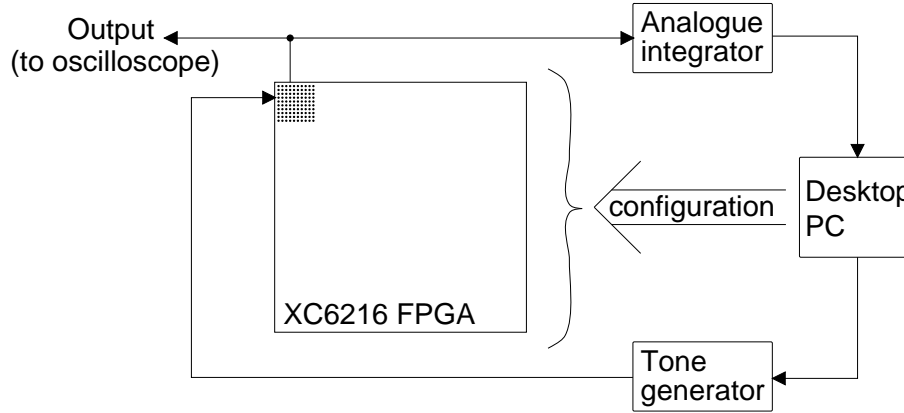
Fig. 2. The experimental arrangement.

The fitness of this physically instantiated circuit was then automatically evaluated as follows. The tone generator drove the circuit's input with five 500ms bursts of the 1kHz square-wave, and five of the 10kHz wave. These ten test tones were shuffled into a random order, which was changed every time. There was no gap between the test tones. The analogue integrator was reset to zero at the beginning of each test tone, and then it integrated the voltage of the circuit's output pin over the 500ms duration of the tone. Let the integrator reading at the end of test tone number t be denoted i_t (t=1,2,... 10). Let S_1 be the set of five 1kHz test tones, and S_{10} the set of five 10kHz test tones. Then the individual's fitness was calculated as:

$$\text{fitness} = \frac{1}{10}\left|\left(k_1 \sum_{t \in S_1} i_t\right) - \left(k_2 \sum_{t \in S_{10}} i_t\right)\right| \quad \text{where} \begin{cases} k_1 = 1/30730.746 \\ k_2 = 1/30527.973 \end{cases} \quad (1)$$

This fitness function demands the maximising of the difference between the average output voltage when a 1kHz input is present and the average output voltage when the 10kHz input is present. The calibration constants k_1 and k_2 were empirically determined, such that circuits simply connecting their output directly to the input would receive zero fitness. Otherwise, with $k_1 = k_2 = 1.0$, small frequency-sensitive effects in the integration of the square-waves were found to make these useless circuits an inescapable local optimum.

It is important that the evaluation method — here embodied in the analogue integrator and the fitness function Eqn. 1 — facilitates an evolutionary pathway of very small incremental improvements. Earlier experiments, where the evaluation method only paid attention to whether the output voltage was above or below the logic threshold, met with failure. It should be recognised that to evolve non-trivial behaviours, the development of an appropriate evaluation technique can also be a non-trivial task.

4 Results

Throughout the experiment, an oscilloscope was directly attached to the output pin of the FPGA (see Fig. 2), so that the behaviour of the evolving circuits could be visually inspected. Fig. 3 shows photographs of the oscilloscope screen, illustrating the improving behaviour of the best individual in the population at various times over the course of evolution.

The individual in the initial random population of 50 that happened to get the highest score produced a constant +5V output at all times, irrespective of the input. It received a fitness of slightly above zero just because of noise. Thus, there was no individual in the initial population that demonstrated any ability whatsoever to perform the task.

After 220 generations, the best circuit was basically copying the input to the output. However, on what would have been the high part of the square wave, a high frequency component was also present, visible as a blurred thickening of the line in the photograph. This high-frequency component exceeds the maximum rate at which the FPGA can make logic transitions, so the output makes small oscillations about a voltage *slightly below* the normal logic-high output voltage for the high part of the square wave. After another 100 generations, the behaviour was much the same, with the addition of occasional glitches to 0V when the output would otherwise have been high.

Once 650 generations had elapsed, definite progress had been made. For the 1kHz input, the output stayed high (with a small component of the input wave still present) only occasionally pulsing to a low voltage. For the 10kHz input, the input was still basically being copied to the output. By generation 1100, this behaviour had been refined, so that the output stayed almost perfectly at +5V only when the 1kHz input was present.

By generation 1400, the neat behaviour for the 1kHz input had been abandoned, but now the output was mostly high for the 1kHz input, and mostly low for the 10kHz input... with very strange looking waveforms. This behaviour was then gradually improved. Notice the waveforms at generation 2550 — they would seem utterly absurd to a digital designer. Even though this is a digital FPGA, and we are evolving a recurrent network of logic gates, the gates are not being used to 'do' logic. Logic gates are in fact high-gain arrangements of a few transistors, so that the transistors are usually saturated — corresponding to logic **0** and **1**. Evolution does not 'know' that this was the intention of the designers of the FPGA, so just uses whatever behaviour these high-gain groups of transistors happen to exhibit when connected in arbitrary ways (many of which a digital designer must avoid in order to make digital logic a valid model of the system's behaviour). This is not a digital system, but a continuous-time, continuous valued dynamical system made from a recurrent arrangement of high-gain groups of transistors — hence the unusual waveforms.
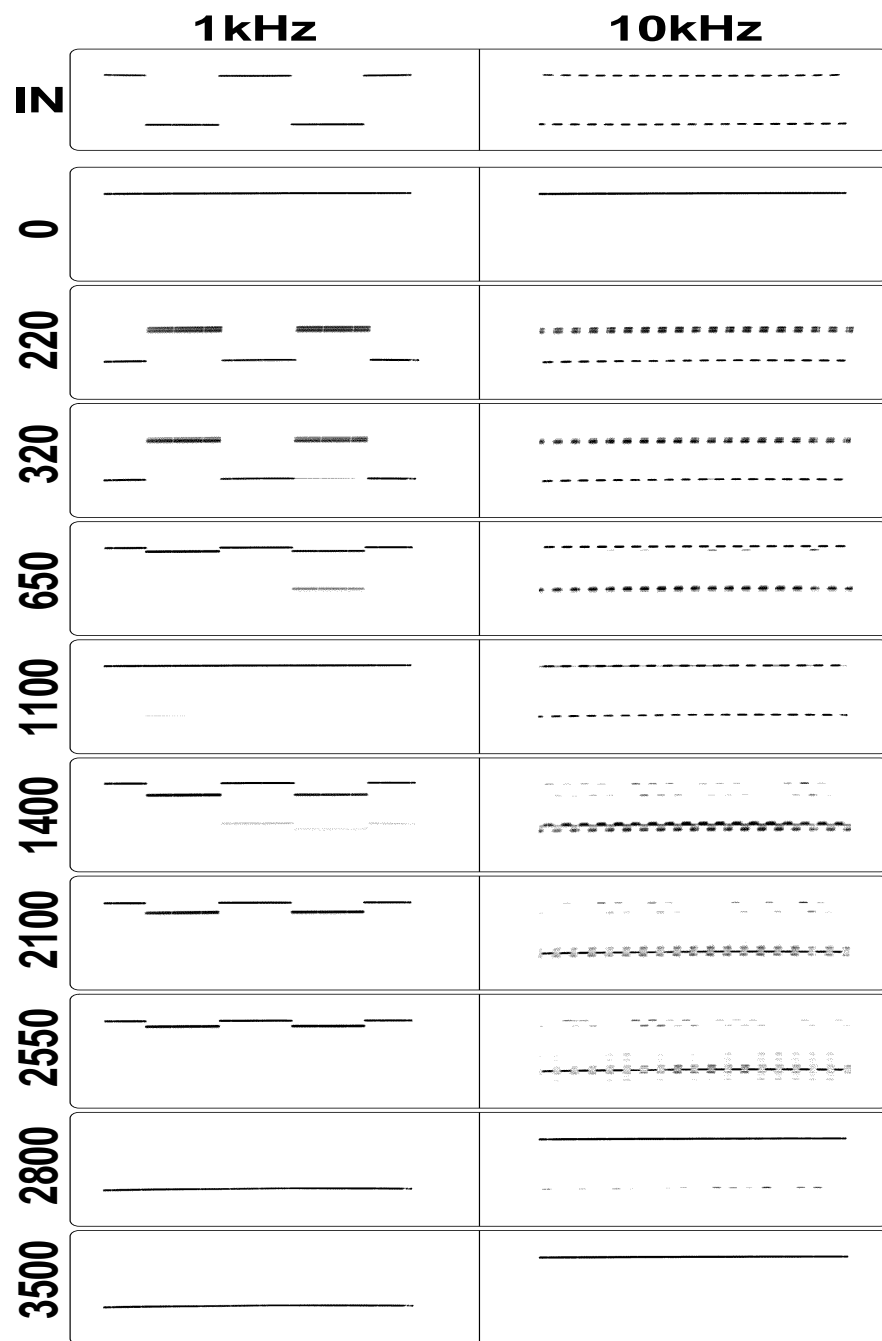
Fig. 3. Photographs of the oscilloscope screen. **Top:** the 1kHz and 10kHz input waveforms. **Below:** the corresponding output of the best individual in the population after the number of generations marked down the side.

By generation 2800, the only defect in the behaviour was rapid glitching present on the output for the 10kHz input. Here, the output polarity has changed over: it is now low for the 1kHz input and high for 10kHz. This change would have no impact on fitness because of the absolute value signs in the fitness function (Eqn. 1); in general it is a good idea to allow evolution to solve the problem in as many ways as possible — the more solutions there are, the easier they are to find.

In the final photograph at generation 3500, we see the perfect desired behaviour. In fact, there were infrequent unwanted spikes in the output (not visible in the photo); these were finally eliminated at around generation 4100. The GA was run for a further 1000 generations without any observable change in the behaviour of the best individual. The final circuit (which I will arbitrarily take to be the best individual of generation 5000) appears to be perfect when observed by eye on the oscilloscope. If the input is changed from 1kHz to 10kHz (or vice-versa), then the output changes cleanly between a steady +5V and a steady 0V without any perceptible delay.

Graphs of maximum and mean fitness, and of genetic convergence, are given in Fig. 4. These graphs suggest that some interesting population dynamics took place, especially at around generation 2660. The experiment is analysed in depth from an evolution-theoretic perspective in a companion paper [5], so I will not dwell on it here. Crucial to any attempt to understand the evolutionary process which took place is the observation that the population had formed a genetically converged 'species' *before* fitness began to increase: this is contrary to conventional GA thinking, but at the heart of Harvey's Species Adaptation Genetic Algorithm (SAGA) [4] conceptual framework.
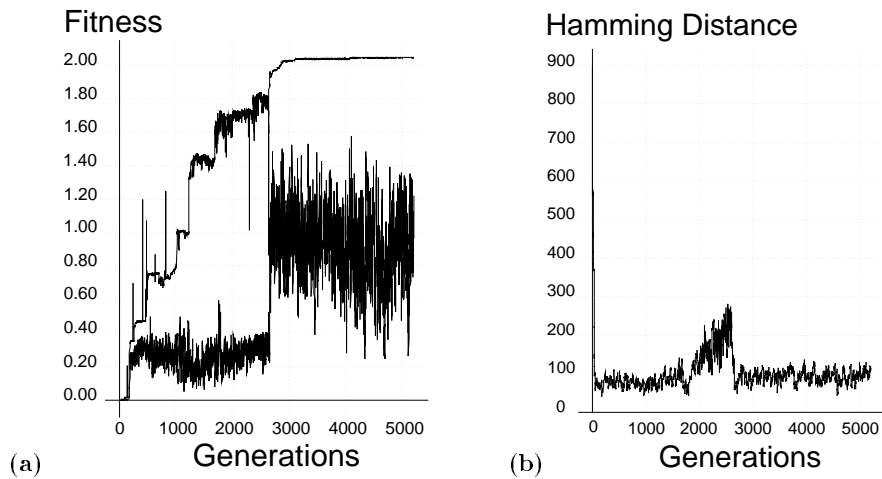

Fig. 4. (a) Maximum and mean fitnesses of the population at each generation. (b) Genetic convergence, measured as the mean Hamming distance between the genotypes of pairs of individuals, averaged over all possible pairs.

The entire experiment took 2–3 weeks. This time was dominated by the five seconds taken to evaluate each individual, with a small contribution from the process of calculating and saving data to aid later analysis. The times taken for the application of selection, the genetic operators, and to configure the FPGA were all negligible in comparison. It is not known whether the experiment would have succeeded if the individuals had been evaluated for shorter periods of time — fitness evaluations should be just accurate enough that the small incremental improvements in performance that facilitate evolution are not swamped by noise. An exciting aspect of hardware evolution is that very high-speed tasks can be tackled, for instance in the pattern recognition or signal processing domains, where fitness evaluation — and hence evolution — can be very rapid. The recognition of audio tones, as in this experiment, is a long duration task in comparison to many of these, because it is reasonable to expect that the individuals will need to be evaluated for many periods of the (slow) input waveforms, especially in the early stages of evolution. The author was engaged in a different project while the experiment was running, so it consumed no *human* time.

5 Analysis

The final circuit is shown in Fig. 5; observe the many feedback paths. The lack of modularity in the topology is unsurprising, because there was no bias in the genetic encoding scheme in favour of this.

Parts of the circuit that could not possibly affect the output can be pruned away. This was done by tracing all possible paths through the circuit (by way of wires and function units) that eventually connect to the output. In doing so, it was assumed that all of a function unit's inputs could affect the function unit output, even when the actual function performed meant that this should not theoretically be the case. This assumption was made because it is not known exactly how function units connected in continuous-time feedback loops actually *do* behave. In Fig. 6, cells and wires are only drawn if there is a connected path by which they could possibly affect the output, which leaves only about half of them.

To ascertain fully which parts were actually contributing to the behaviour, a search was conducted to find the largest set of cells that could have their function unit outputs simultaneously clamped to constant values (**0** or **1**) without affecting the behaviour. To clamp a cell, the configuration was altered so that the function output of that cell was sourced by the flip-flop inside its function unit (a feature of the chip which has not been mentioned until now, and which was not used during evolution): the contents of these flip-flops can be written by the PC and can be protected against any further changes. A program was written to randomly select a cell, clamp it to a random value, perform a fitness evaluation, and to return the cell to its un-clamped configuration if performance was degraded, otherwise to leave the clamp in place. This procedure was iterated, gradually building up a maximal set of cells that can be clamped without altering fitness.
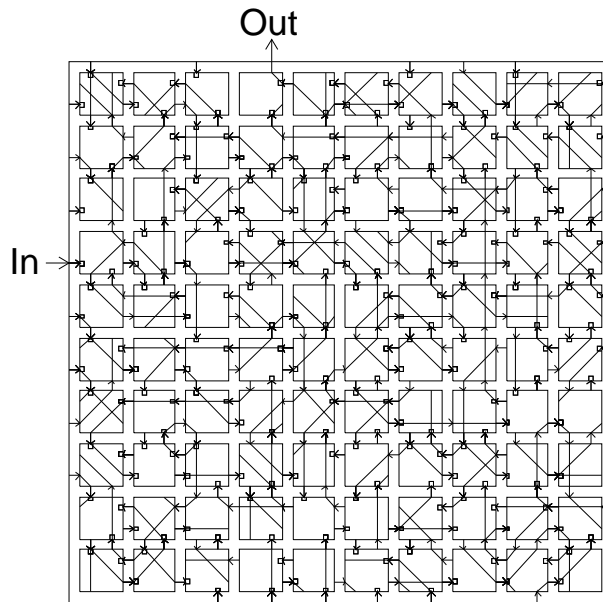
Fig. 5. The final evolved circuit. The 10×10 array of cells is shown, along with all connections that eventually connect an output to an input. Connections driven by a cell's function output are represented by arrows originating from the cell boundary. Connections into a cell which are selected as inputs to its function unit have a small square drawn on them. The actual setting of each function unit is not indicated in this diagram.

In the above automatic search procedure, the fitness evaluations were more rigorous (longer) than those carried out during evolution, so that very small deteriorations in fitness would be detected (remember there is always some noise during the evaluations). However, there was still a problem: clamping some of the cells in the extreme top-left corner produced such a tiny decrement in fitness that the evaluations did not detect it, but yet by the time *all* of these cells of small influence had been clamped, the effect on fitness was quite noticeable. In these cases manual intervention was used (informed by several runs of the automatic method), with evaluations happening by watching the oscilloscope screen for several minutes to check for any infrequent spikes that might have been caused by the newly introduced clamp.

Fig. 7 shows the functional part of the circuit that remains when the largest possible set of cells has been clamped without affecting the behaviour. The cells shaded gray cannot be clamped without degrading performance, *even though there is no connected path by which they could influence the output* — they were not present on the pruned diagram of Fig. 6. They must be influencing the rest of the circuit by some means other than the normal cell-to-cell wires: this probably takes the form of a very localised interaction with immediately neighbouring
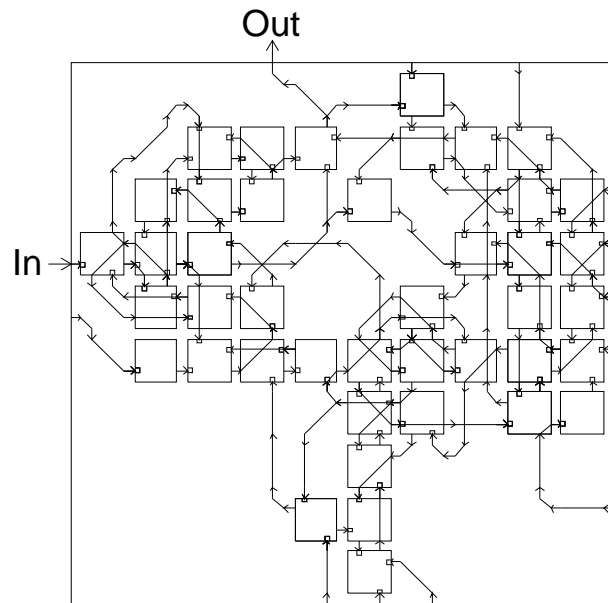
Fig. 6. The pruned circuit diagram: cells and wires are only drawn if there is a connected path through which they could possibly affect the output.
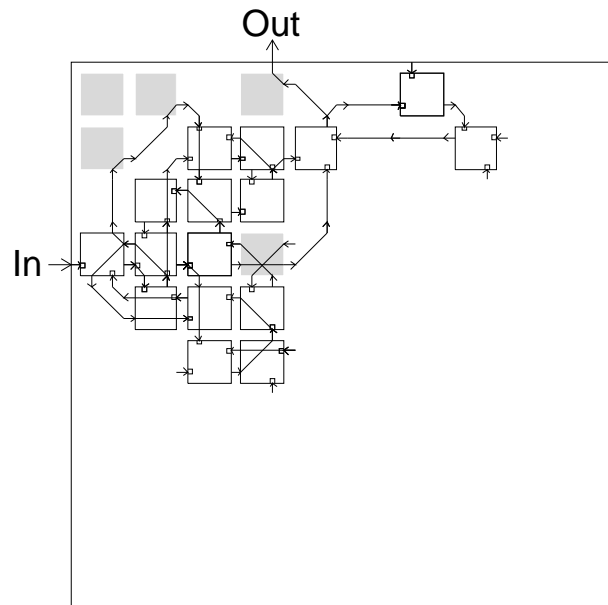

Fig. 7. The functional part of the circuit. Cells not drawn here can be clamped to constant values without affecting the circuit's behaviour — see main text.

components. Possible mechanisms include interaction through the power-supply wiring, or electromagnetic coupling. Clamping one of the gray cells in the top-left corner has only a small impact on behaviour, introducing either unwanted pulses into the output, or a small time delay before the output changes state when the input frequency is changed. However, clamping the function unit of the bottom-right gray cell, which also has two active connections routed through it, degrades operation severely even though that function output is not selected as an input to any of the NEWS neighbours: it doesn't go anywhere.

This circuit is discriminating between inputs of period 1ms and 0.1ms using only 32 cells, each with a propagation delay of less than 5ns, and with no off-chip components whatsoever: a surprising feat. Evolution has been free to explore the full repertoire of behaviours available from the silicon resources provided, even being able to exploit the subtle interactions between adjacent components that are not directly connected. The input/output behaviour of the circuit is a digital one, because that is what maximising the fitness function required, but the complex analogue waveforms seen at the output during the intermediate stages of evolution betray the rich continuous-time continuous-value dynamics that are likely to be internally present.

In [12] it was shown that in GAs like the one used here, there can be a tendency for circuits to evolve to be relatively unaffected by genetic mutations, on average. (This effect was first noticed in a different context [2, 8], and only occurs significantly in engineering GAs under particular — but common — conditions.) Depending on the genetic encoding scheme, this can have a variety of consequences for the phenotype, including graceful degradation in the presence of certain hardware faults. For our circuit evolved here, however, increasing the proportion of the possible mutations that do not reduce fitness may result in *decreasing* the number of cells implicated in generating the behaviour. So it may be no accident that the functional core of cells seen in Fig. 7 is small.

So far, we have only considered the response of the circuit to the two frequencies it was evolved to discriminate. How does it behave when other frequencies of square wave are applied to the input? Fig. 8 shows the average output voltage (measured using the analogue integrator over a period of 5 seconds) for input frequencies from 31.25kHz to 0.625kHz. When the case temperature of the FPGA is 31.2°C (as it was, ±5°C, during evolution), then for input frequencies ≥ 4.5kHz the output stays at a steady +5V, and for frequencies ≤ 1.6kHz at a steady 0V. Thus, the test frequencies (marked F1 and F2 in the figure) are correctly discriminated with a considerable margin for error. As the frequency is reduced from 4.5kHz, the output begins to rapidly pulse low for a small fraction of the time; as the frequency is reduced further the output spends more time at 0V and less time at +5V, until finally resting at a steady 0V as the frequency reaches 1.6kHz. These properties might be considered 'generalisation.'

Fig. 8 also shows the circuit's behaviour when hot or cold. The high temperature was achieved by placing a 60W light-bulb near the chip, the low temperature by opening all of the laboratory windows on a cool breezy evening. Varying the temperature moves the frequency response curve to the left or right, so once
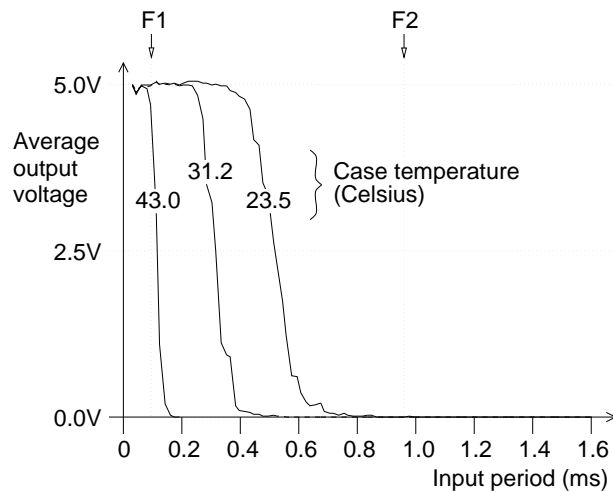
Fig. 8. The frequency response of the final circuit, measured at three different temperatures. F1 and F2 are the two frequencies that the circuit was evolved to discriminate; in fact, for ease of implementation, they happen to be of period 0.096 ms (10.416 kHz) and 0.960 ms (1.042 kHz) respectively, rather than exactly 10 kHz and 1 kHz as mentioned in the main text.

the margin for error is exhausted the circuit no longer behaves perfectly to discriminate between F1 and F2. In the examples given here, at 43.0°C the output is not steady at +5V for F1, but is pulsing to 0V for a small fraction of the time. Conversely, at 23.5°C the output is not a steady 0V for F2, but is pulsing to +5V for a small fraction of the time. This is not surprising: the only time reference that the system has is the natural dynamical behaviour of the components, and properties such as resistance, capacitance and propagation delays are temperature dependent. The circuit operates perfectly over the 10°C range of temperatures that the population was exposed to during evolution, and no more could reasonably be expected of it. We'll return to the issue of evolving temperature stability in the discussion that follows.

6   Discussion

The idea of enabling sophisticated behaviour to arise from an unusually small number of electronic components by allowing them to interact more freely than is customary dates back at least as far as Grey Walter's electromechanical 'tortoises' in 1949 (when the active components were thermionic valves and relays)[6]. More recently, Mead's philosophy for analogue neural VLSI has been to exploit the behaviours that semiconductor structures naturally exhibit, rather than choosing a set of functions and *then* trying to implement them in hardware [9]. In 'Pulse-stream' neural networks, the use of continuous-time dynamics has been demonstrated to release new power from a digital substrate [10].

The core principle that these ideas approach is to look for an efficient composition of electronic components selected from a set of *physical* (not abstract) resources, such that their coupled natural behaviours collectively give rise to the required overall system behaviour. In this paper, we have seen evolution do exactly that. A 'primordial soup' of reconfigurable electronic components has been manipulated according to the overall behaviour it exhibits, and on no other criterion, with no constraints imposed upon the structure or its dynamical behaviour other than those inherent in the resources provided.

For a human to design such a system on paper would require the set of coupled differential equations describing the detailed electronic and electromagnetic interactions of every piece of metal, oxide, doped silicon, etc., in the system to be considered at all stages of the design process. Because this is not practical, the structure and dynamical behaviour of the system must be constrained to make design tractable. The basic strategy is to: (1) Break the system into smaller parts that can be understood individually. (2) Restrict the interactions between these parts so *that* can be understood. (3) Apply 1 and 2 hierarchically, allowing design at increasing levels of abstraction.

Thus, conventional design always requires constraints to be applied to the circuit's spatial structure and/or dynamical behaviour. Evolution, working by judging the effects of variations applied to the real physical hardware, does not. That is why the circuit was evolved without the enforcement of any spatial structure, such as limitations upon recurrent connections, or the imposition of modularity, and without dynamical constraints such as a synchronising clock or handshaking between modules.[3] This sets free all of the detailed properties of the components to be used in developing the required overall behaviour. It is reasonable to claim that the evolved circuit consequently uses significantly less silicon area than would be required by a human designer faced with the same problem, but such assertions are always open to attack from genius designers.

The outstanding problem with allowing evolution a free hand to exploit the resources is that the evolving circuits can become tailored too specifically to the exact conditions prevailing during evolution. For instance, our example circuit was shown to be using subtle interactions between adjacent components on the silicon; surely if this evolved configuration were used with another, nominally identical, FPGA chip then it would no longer work? Every chip has slightly different propagation delays, capacitances, etc., and the circuit could have come crucially to rely on those of the particular chip on which it was evolved. To investigate this question, the final population at generation 5000 was used to configure a *completely different* 10×10 region of the same FPGA chip (Fig. 9).

When used to configure this new region, the individual in the population that was fittest at the old position deteriorated by $\approx 7\%$. However, there was another individual in the population which, at the new position, was within 0.1% of perfect fitness. Evolution was allowed to continue at the new position, and after only 100 generations had recovered perfect performance. When this new

---

[3] See [14, 13] for an expansion of this argument and earlier experiments specifically designed to explore the feasibility of such unconstrained evolution.
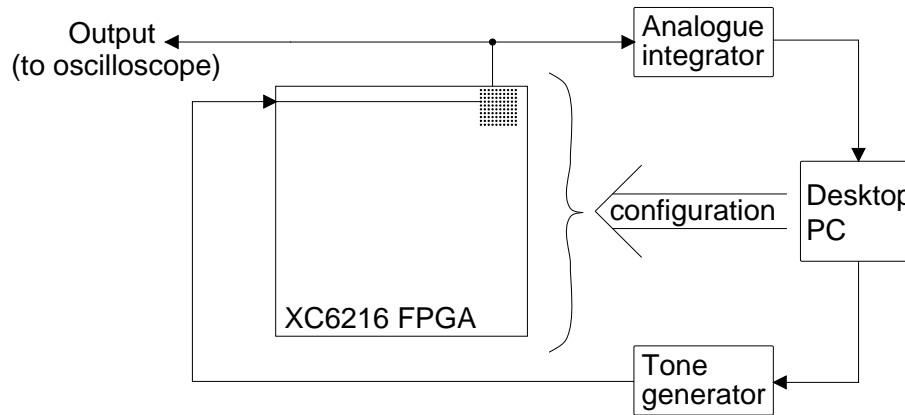
Fig. 9. Moving the circuit to a different region of the FPGA.

population was moved back to the *original* region of silicon, again the transfer reduced the fitness of the individual that used to be fittest, but there was another individual in the population that behaved perfectly there.

Recall that the circuit works perfectly over the 10°C range of temperatures to which the population was exposed during evolution. This, together with the ease with which evolution was observed to adapt the circuit to work on a new region of silicon, suggests a unified solution to the problem of evolving circuits with engineering tolerances. The plan is to have ~10 nominally identical FPGA chips, selected from separate batches (so as to be as different from each-other as possible), held at different temperatures using Peltier-effect heat pumps, and with a range of permissible power-supply voltages. To evaluate an individual's fitness, it will be tested on each of the FPGAs and given a score according to its ability to perform under all of these conditions. This will not slow down evolution because the FPGAs can operate in parallel. The hope is that evolution will produce a configuration that works at any permissible temperature and power-supply voltage and for any FPGA of that type. Success is not certain, because it will not be possible to expose the evolving circuits to every possible combination of conditions, but there is good reason to think that it can be made easier for evolution to generalise than to specialise. If the unconstrained efficient evolutionary exploitation of resources can be made an engineering practicality, the pay-offs will be great.

7   Conclusion

When an evolutionary fitness evaluation is the judging of the physical behaviour of a reconfigurable electronic device, evolution can be allowed to explore the whole space of possible configurations. Much of this space is beyond the scope of conventional design methods, so concepts of what electronic circuits can look like need to be broadened. The benefit of such a step is that evolution can then utilise

the available resources more efficiently, with richer circuit structure, dynamical behaviour, and respect for the natural physical behaviour of the medium. This has been demonstrated by a simple, but not toy, experiment — the first direct 'intrinsic' evolution of an FPGA configuration.

References

1. H. de Garis. Growing an artificial brain with a million neural net modules inside a trillion cell cellular automaton machine. In *Proc. 4th Int. Symp. on Micro Machine and Human Science*, pp211–214, 1993.
2. M. Eigen. New concepts for dealing with the evolution of nucleic acids. In *Cold Spring Harbor Symposia on Quantitative Biology*, vol. LII, 1987.
3. D.E. Goldberg. *Genetic Algorithms in Search, Optimisation & Machine Learning*. Addison Wesley, 1989.
4. I. Harvey. Species Adaptation Genetic Algorithms: A basis for a continuing SAGA. In F. J. Varela and P Bourgine, eds, *Towards a Practice of Autonomous Systems: Proc. 1st Eur. Conf. on Artificial Life*, pp346–354. MIT Press, 1992.
5. I. Harvey and A. Thompson. Through the labyrinth evolution finds a way: A silicon ridge. In this volume: *Proc. 1st Int. Conf. on Evolvable Systems: From Biology to Hardware 1996 (ICES96)*, Springer-Verlag LNCS.
6. O. Holland. Grey Walter: the pioneer of real artificial life. In *ALife V: Proc. 5th Int. Workshop Synthesis and Simulation of Living Systems*. MIT Press, 1996.
7. P. Horowitz and W. Hill. *The Art of Electronics*. Cambridge University Press, 2nd edition, 1989.
8. M.A. Huynen and P. Hogeweg. Pattern generation in molecular evolution: Exploitation of the variation in RNA landscapes. *J. Mol. Evol.*, 39:71–79, 1994.
9. C.A. Mead. *Analog VLSI and Neural Systems*. Addison Wesley, 1989.
10. A.F. Murray. Analogue neural VLSI: Issues, trends and pulses. *Artificial Neural Networks*, 2:35–43, 1992.
11. J.V. Oldfield and R.C. Dorf. *Field Programmable Gate Arrays: Reconfigurable logic for rapid prototyping and implementation of digital systems*. Wiley, 1995.
12. A. Thompson. Evolutionary techniques for fault tolerance. In *Proc. UKACC Int. Conf. on Control 1996 (CONTROL '96)*, pp693–698. IEE Conference Publication No. 427, 1996.
13. A. Thompson. Silicon evolution. In J. R. Koza et al., eds, *Proc. of Genetic Programming 1996 (GP96)*, pp444–452. MIT Press, 1996.
14. A. Thompson, I. Harvey & P. Husbands. Unconstrained evolution and hard consequences. In E. Sanchez & M. Tomassini, eds, *Towards Evolvable Hardware: The evolutionary engineering approach*, pp136–165. Springer-Verlag LNCS 1062, 1996.
15. Xilinx, Inc. XC6200 Advanced product specification V1.0, 7/96. In *The Programmable Logic Data Book*. 1996. See http://www.xilinx.com.

This article was processed using the LaTeX macro package with LLNCS style