Evolving *messy* gates for fault tolerance: some preliminary findings

Julian F. Miller
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
j.miller@cs.bham.ac.uk

Morten Hartmann[1]
Department of Computer and Information Science
The Norwegian University of Science and Technology
7491 Trondheim, Norway
m.hartmann@idi.ntnu.no

Abstract

We investigate a preliminary model of gate-like components with added random noise. We refer to these types of components as ***messy****. The principal idea behind messy gates is that evolving circuits using messy gates may confer some beneficial properties, one being fault-tolerance. The exploitation of the physical characteristics has already been demonstrated in intrinsic evolution of electronic circuits. This provided some of the inspiration for the work reported in this paper. Here we are trying to create a* ***simulateable*** *world in which "physical characteristics" can be exploited. We are also trying to study the question: What kind of components are most useful in an evolutionary design scenario?*

1 Introduction

Natural evolution has produced the most subtle and complex bio-chemical information processing machines known (i.e. living creatures). In addition to this complexity living systems possess a remarkable degree of fault tolerance and robustness. At this point it is necessary to clarify the exact meanings of the terms: fault-tolerance and robustness. Robustness deals primarily with problems that are expected to occur and must be protected against. By contrast, fault tolerance primarily deals with problems that are unexpected. Humans can design systems to be robust but true fault tolerance is a much more difficult problem. This is particularly acute in digital electronics. Digital gates are robust from the point of view of minor changes in input voltages but systems built from them are fragile to stuck-at faults.

Another aspect of human designed systems is that they are usually built from production line components (especially if they are electronic). Living systems are built from components (i.e. cells) that vary considerably in their properties. It is now recognised that many of the advantageous properties of living systems emerge from the way in which the individual properties of the components is exploited.

Artificial neural networks (ANN) were one of the first bio-inspired circuits to be developed. It has been suggested that ANNs exhibit graceful degradation in the presence of faults [1]. However more recent work has indicated that ANNs are not intrinsically fault tolerant [6, 11].

In recent decades the use of design algorithms that employ the principles of Darwinian evolution have been applied to the design of electronic systems [2,5,12,13]. Such work has become known as Evolvable Hardware. One of the most intriguing findings in this field is that of Adrian Thompson [7]. He showed that it was possible for artificial evolution to create FGPA designs that exploited the physical characteristics of the silicon substrate to efficiently carry out a particular task. Thompson found that unconstrained artificial evolution explored very unusual ways of solving problems *precisely* because it was able to exploit the subtle and incidental physical characteristics. It can be argued that evolution has produced such complex systems because it can make use of the full, *unmodellable* richness of the physical world.

Another question raised from Thompson's work is the following: *What basic components should we be using in artificial evolution?* It does not seem very likely that the electronic components that have been created for human design should happen to be particularly useful in artificial evolution. Indeed it could be seen as a testament to the power of evolution that using them, artificial evolution could produce anything useful at all. If one is going to build practical systems using artificial evolution it appears that one has to go back to basics and try to design radically new forms of electronic components or circuits [e.g. 14] that might assist the artificial evolutionary process. This could be done in two ways. First, one could search for special

[1] This work was carried out while in the School of Computer Science, University of Birmingham

materials and then subject them to *intrinsic* artificial evolution, and second, one could try to define new kinds of components in simulation and subject them to *extrinsic* evolution. The former is obviously a good approach, however it is potentially *very expensive* and likely to be very difficult. The latter though, perhaps, is not as exciting, however it is *feasible* and could actually assist the former goal by helping us to identify what kinds of properties are important. These thoughts were the starting point for the work reported in this paper.

The paper is divided as follows: Section 2 is concerned with the definition of the new component model (which is described as **messy**). Section 3 describes the genotype representation and evolutionary algorithm that was used. Section 4 describes the experiments performed. In section 5 the experimental results are presented. In section 6 the concluding remarks are made. The paper finishes with a discription of possible future work in section 7.

2 Messy gates

Since the aim is not only to create new types of components for use in artificial evolution but also to *understand* exactly how they work, it was necessary to choose a new model that was adjustable. It was desirable to be able to change a parameter continuously so that the new component model could become the same as a familiar traditional component. A natural choice of traditional component was a digital logic gate. Accordingly, a model that was that of a digital multiplexer with additional randomness on the output was created. The new model took real valued input combined the inputs to give a real valued output and random noise was then superimposed on the output. These gates are being referred to as *messy* and as having a degree of *messiness*.

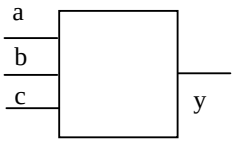

Figure 1: Model of messy MUX

The equation below describes the messy MUX:

$$y = mr + a\bar{c} + bc \qquad (1)$$

where *m* represents the constant value of messiness chosen for the entire circuit and r represents a real random number uniformly sampled from the interval [-1.0, 1.0]. The inputs to the messy MUX are *a* and *b*, and *c* is the control input. The bar over *c* refers to *1-c*. All variables are real-valued. Clearly when *m=0* and *a*, *b*, and *c* are only allowed to be 0 or 1, the digital MUX is recovered. In the experiments a messy MUX with one input inverted (input b would then become 1-b) was used as well.

3 Evolutionary algorithm and genotype

The genotype representation is the same as that used for evolving digital designs [3,4]. It is best explained with a small example. In figure 2 are shown four messy MUX (mMUX) gates. The numbers (0 -3) refer to the four primary inputs of the target function. The numbers on the inputs to the mMUX refer to the connections to the primary inputs or the outputs of other mMUX. The outputs of the mMUX are labelled (sequentially following on from the inputs). Thus the second mMUX on the left has the "a" input connected to the output of the first mMUX, the other two mMUX inputs are connected to the two primary inputs 3 and 0. In this example it is assumed that the target function has four outputs. Thus the genotype has four output connections (4 5 7 3). The numbers in bold refer to which of the two mMUX were being used (0 refers to mMUX with no inputs inverted, 1 refers to the "b" input being inverted). The numbers printed in grey refer to inactive genes or mMUX (i.e the third mMUX does not have its output connected). This paper only considers feed-forward circuits. The representation allows any mMUX to have its input connected to any other mMUX on its left.

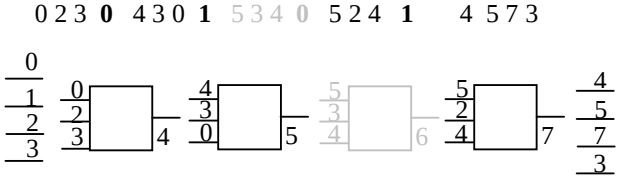

Figure 2: Example genotype and resulting circuit

The evolutionary algorithm employed was a simple form of (1+4) evolutionary strategy (ES). In this case a population of five chromosomes is randomly generated and the fittest chromosome selected. The new population is then filled with mutated versions of this. Random mutation is defined as a percentage of genes in the population that were mutated. The mutation operator respects the feed-forward nature of the circuits and also the different alphabets associated with connections and functions.

Each *circuit* has a fixed value of messiness m. However each gate has its own random value r (equation 1). This is illustrated in Figure 3.

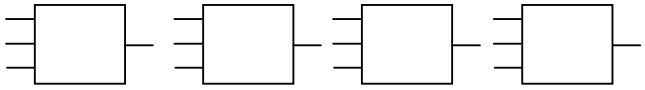

Figure 3: Circuit schematic with individual random values

The fitness of an individual is measured by testing the chromosome with all possible combinations of inputs and comparing the output values with the target Boolean truth table. For all experiments described in this paper, the target is a 2-bit multiplier. Thus, there are 4 inputs, 4 outputs and 2^4 possible input test vectors yielding a total of 64 output bits in the truth table.

Fitness is equal to the number of output bits of a circuit being equal to the corresponding bit in the target truth table. This is reffered to as bitwise correctness.

A perfectly functional circuit would have all its output bits equal to the output bits of the truth table, and thus the bitwise correctness would equal 64 in the case of the 2-bit multiplier.

In the case of messy circuits, the real valued output signals of the circuit were rounded when being compared to the target truth table.

4 Experiments

Several experiments were performed to investigate the nature of the mMUX and its influence on the evolved circuits and the evolutionary algorithm. These experiments are described below.

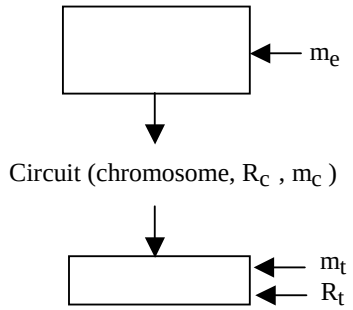

Figure 4: Experimental setup

The setup for the experiments is shown in Figure 4. The evolutionary algorithm produces a circuit consisting of a chromosome, a set of random values R_c and the messiness used (m_c=m_e). A test bench allows measuring the performance of a circuit by using different values of messiness (m_t), different sets of random values (R_t) and introducing stuck-at faults.

The first two experiments investigated whether it was feasible to evolve circuits that exhibited a natural robustness to internal random noise within some specified range.

A set of about 500 chromosomes was evolved for each of six different values of *m*. The fitness measure was based on taking an average of the bitwise correctness over 15 sets of random values r (for each gate). Evolution was halted each time the fitness was equal to 64 (all 15 chromsomes had bitwise correctness equal to 64). This set of chromosomes was used to generate the results of the first three experiments. Thus, these experiments investigates different configurations of the testbench, with the fitness of each circuit measured as an average over 50 trials. This was done both to introduce new random values R_t for every test performed, as well as testing different randomly chosen gates for stuck-at faults.

4.1 First experiment

The first experiment sought to investigate how computationally demanding it is to evolve circuits with high values of *m*, as well as the general performance of the circuits in an environment equally noisy to the one in which it was evolved (m_t=m_c). The experiment was otherwise carried out as explained earlier.

4.2 Second experiment

Thompson has demonstrated that it is possible to evolve robust circuits intrinsically by exposing them to various environments [8, 9]. In the second experiment, the same set of chromosomes were tested with the messiness of the test bench m_t, being set to increasingly higher values. This simulates the circuits running in increasingly more noisy environments, and disregards whatever value m_e used when the chromosomes were evolved. This was done to investigate the robustness of the chromosomes with regards to the amplification m_t of the internal noise, and its relation to the value of m_e used in the evolution of the different chromosomes.

4.3 Third experiment

An interesting property of a digital circuit is its fault tolerance. An experiment was carried out to measure the tolerance in the evolved circuits towards stuck-at faults.

The test bench was set up to subject the circuits to stuck-at-1 faults, by fixing a gate output to 1. Stuck-at-1 faults were selected since their impact on a multiplier is on average more severe then stuck-at-zero faults. This is due to the fact that the output part of the 2-bit multiplier truth table contains 14 zeros and 50 ones. Thus, increasing numbers of stuck-at-0 faults force the bitwise correctness towards 50,

while increasing numbers of stuck-at-1 faults force the bitwise correctness towards 14.

4.4 Fourth experiment

The last experiment sought to investigate how evolution would be capable of exploiting individual characteristics of given gates. This was done by generating random values for R_e for each gate only once for each run of the evolutionary algorithm. The algorithm would then try to utilize the properties of each individual mMUX in the circuit to solve the problem.

The set of random values was saved with each chromosome as R_c, and the test bench was configured to use the saved random values as the values to be used under test (R_t=R_c).

The fitness function was modified by adding the number of redundant nodes to the bitwise correctness (provided the bitwise correctness equalled 64). This was done to investigate if it would be possible to evolve smaller circuits for higher values of m.

5 Results

This section presents the results obtained through the experiments carried out in section 4. The results are briefly discussed within the limits of the scope of this paper.

5.1 First experiment

Evolving chromosomes with bitwise correctness equal to 64 was more computationally expensive for large values of *m*. Purely digital circuits (*m*=0) took an average of 2000 generations to evolve. On the other hand, it required an average of more than 30000 generations to evolve circuits with a messiness value of *m*=0.25.

The seemingly exponential growth of computational labour is illustrated by the graph in Figure 5.

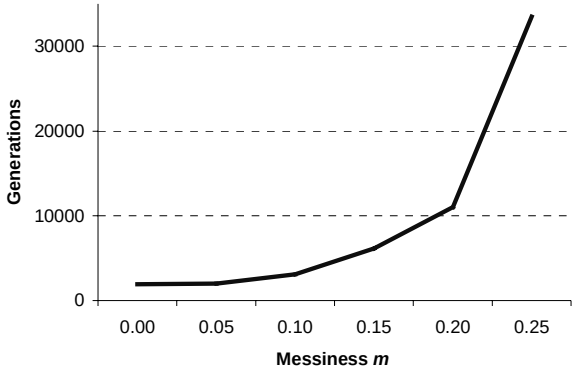

Figure 5: Growth of required evolutionary effort

Even though all chromosomes are evaluated 15 times to distribute the random values creating noise at the gate outputs, the same chromosome may not obtain perfect fitness if a new set of random values are introduced. In such a way, the number of evaluations during evolution can be considered a choice between computational effort and robustness of the evolved circuit (with regards to random values introduced).

The resulting average fitness is shown in Figure 6.

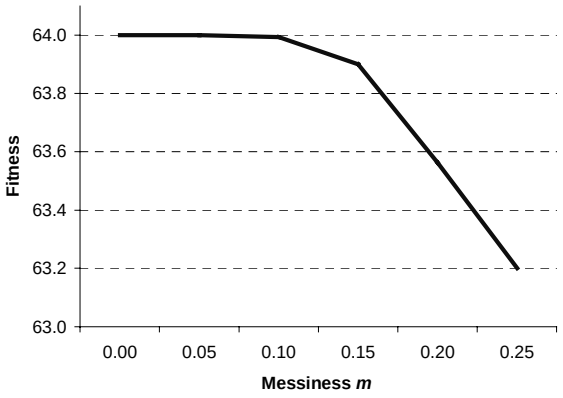

Figure 6: Average fitness measured through tests

Even though this figure shows a negative trend, it implies that messy circuits are pretty robust to variations in the internal noise within the range m_e used when the particular circuit was evolved, as the drop in fitness was quite small.

Finally, this experiment revealed another property of evolving messy circuits. Figure 7 illustrates how chromosomes evolved with high values of *m* tend to produce smaller circuits then those produced when messiness is low. This is probably due to the fact that evolution finds a way to reduce the amount of noise the circuit is exposed to. Each new gate means a new noisy value to cope with.

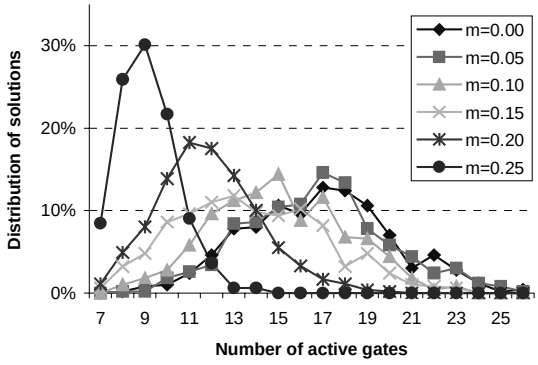

Figure 7: The number of gates tend to be small for larger values of *m*.

5.2 Second experiment

When the evolved circuits were tested in increasingly more noisy environments, a clear trend showed increased robustness for circuits evolved with higher messiness (m_e). This trend is shown in Figure 8. The graph shows the deviation of the fitness to the average fitness over all evolved circuits, when exposed to increasing noise in the environment.

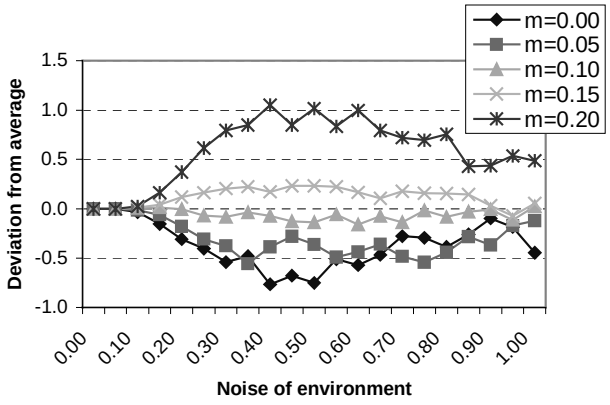

Figure 8: Average fitness when exposed to more noisy environments.

5.3 Third experiment

It was difficult to see any clear trend on whether messy circuits were more tolerant then pure digital ones when they used a small number of gates. It was desirable to investigate whether larger circuits with high messiness would be more fault tolerant. To reliably obtain large circuits (using the maximum number of gates =30) a term had to be added to the fitness that favoured larger circuits. This meant that circuits without a bitwise correctness of 64 had to be accepted. However it was observed that these circuits proved to be largely more fault tolerant to the stuck-at-1 faults and showed a more graceful degradation when compared to the zero messiness case. This trend is shown in the graph in Figure 9. Note that the test bench in this case introduced the faults in the same environment that each chromosome was originally evolved (same value of *m*). So the circuits had to cope with intrinsic randomness associated with the messiness m_c, in addition to the introduced faults.

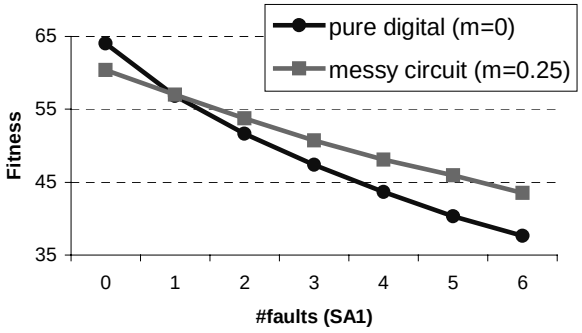

Figure 9: Difference in degradation

An interesting aspect of this graceful degradation is the fact that this tolerance is implicit, since these circuits were never evolved explicitly with fault tolerance to stuck-at faults as a part of the EA. A comparison of the fault tolerance of the evolved digital case (m=0) and conventional circuits was not carried out. Certain evolutionary systems may create circuits with some natural mutation tolerance [10], but this is probably not the case in this system due to its steep hill climbing nature.

5.4 Fourth experiment

The fourth experiment revealed that evolving circuits with permanent sets of values R_c for each chromosome, was slightly more computationally demanding for higher values of m. This property is plotted in Figure 10.

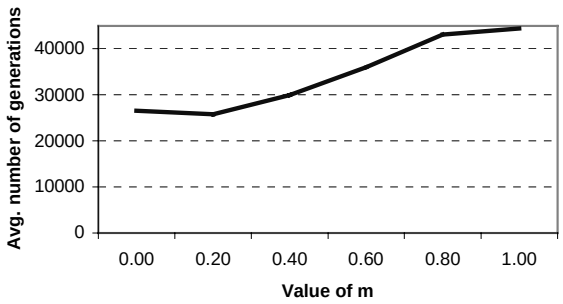

Figure 10: Higher values of m required a slightly larger amount of computational effort to evolve.

Figure 11 shows the percentage of evolved circuits having a particular number of gates for various messiness values. The low values of m tended to yield circuits with 7 gates, higher values of m had a wider distribution of evolved circuits.

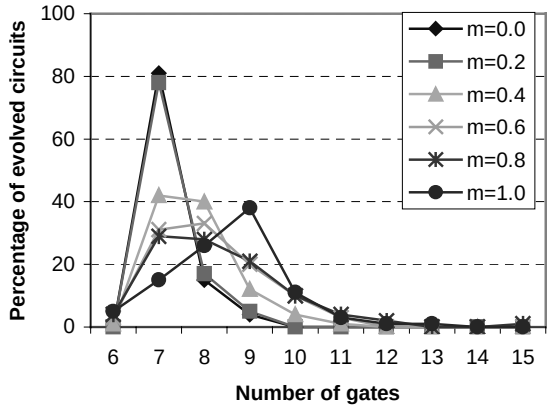

Figure 11: Distribution of solutions with regards to the number of gates in the evolved circuits

Figure 12 shows that for values of m greater then 0.2, certain circuits were found that used only 6 gates. This is interesting, since it appears the minimum number of gates (of the two types of multiplexers used) that can be used to solve the problem in a pure digital manner is 7 [4]. An example of a circuit of this type is shown in Figure 13. A and B are the two 2-bit numbers being multiplied. A circle at an input illustrates an inverted value. The shown circuit was evolved with m equal to 0.4. It demonstrates the ability of a blind evolutionary process to exploit all the "physical characteristics" of the components. This is reminiscent of Thompson's findings in [7] where he found that robust evolved clocked digital circuits exploited glitches, even though these are shunned in conventional human design.

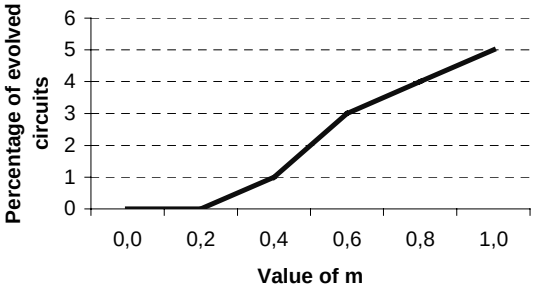

Figure 12: Graph of percentage of evolved circuits with only 6 gates

6 Conclusions

In this paper a new model of a gate-like component with added random noise was proposed. The experiments carried indicated that such components are beneficial in several ways. Firstly the new gate-like models naturally offered a robustness to noise. Secondly circuits that were evolved using these messy components exhibited implicit fault tolerance to stuck-at-faults. Finally experiments indicated that in creating a simulateable world, "physical" characteristics (intrinsic random values) could be exploited to create surprisingly efficient designs (the six mMUX 2-bit multiplier). One advantage of such a simulation is that it is possible to inspect every detail of the "physics". In addition all the designs are replicable and the functionality of the evolved circuit can be verified mathematically. This is much harder to do when evolving circuits intrinsically.

It is not easy to imagine a human design processes that could exploit such random differences. Artificial evolution however is quite adept at exploiting such things.

7 Future work

This paper is a preliminary study into an area that to the knowledge of the authors is relatively unexplored. The model of messiness discussed here is really just a starting point. It is likely that more complex models of components would be much more suitable for use in an evolutionary design process.

Further work needs to be done to explore how the implicit fault tolerance of messy circuits relates to circuits whose fitness explicitly takes into account their fault tolerance.

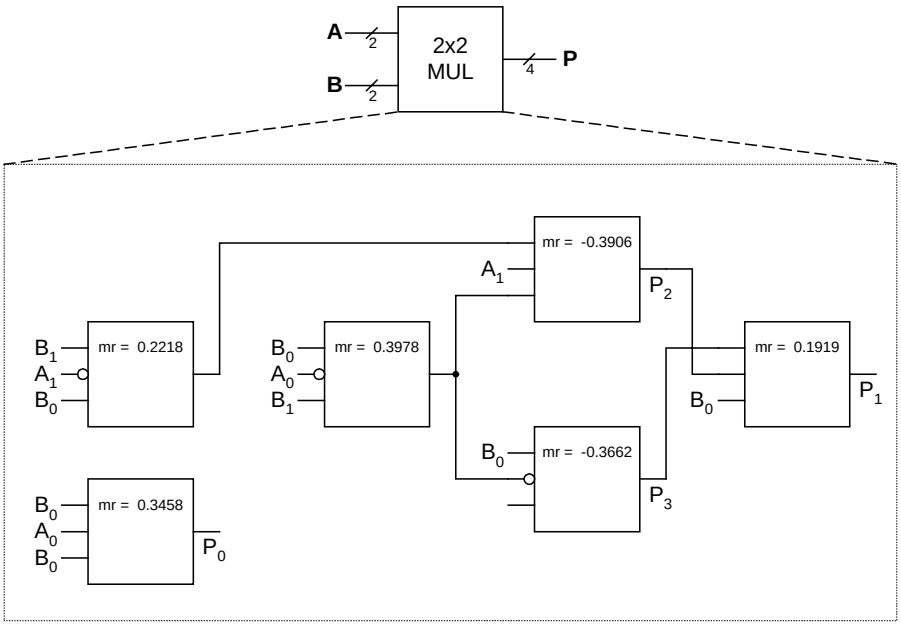

Figure 13: Example of small evolved circuit (m=0.4)

An aspect for future work is also the investigation of which new models of components are best suited for digital problems and which for analogue.

Finally, an interesting study could be done into the nature of the circuits produced that were smaller then those possible to design with pure digital MUX. There seems to be a relation between the set of random values given to a circuit and how small evolution manages to make the resulting circuit. Searching for general principles within the values of these random numbers as well as the nature of how evolution exploits them could reveal very interesting results.

References

[1] L.A. Belfore and B.W. Johnson. "The fault-tolerance of neural networks", The International Journal of Neural Networks Research and Applications 1, pp. 24-41 (Jan 1989).

[2] T. Higuchi and M. Iwata (eds.). Proceedings of the 1st International Conference on Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science (ICES96), Springer-Verlag, Berlin, vol. 1259, 1996.

[3] J. F. Miller and P. Thomson. "Cartesian Genetic Programming", in R. Poli, W. Banzhaf, W.B. Langdon, J. F. Miller, P. Nordin, T. C. Fogarty (eds.), Third European Conference on Genetic Programming Edinburgh 2000 (EuroGP2000), Lecture Notes in Computer Science, vol. 1802, pp. 121-132, Springer-Verlag, Heidelberg, 2000.

[4] J. F. Miller, D. Job, V.K. Vassilev. "Principles in the Evolutionary Design of Digital Circuits - Part I", in W. Banzhaf (ed.), Genetic Programming and Evolvable Machines, Vol. 1, No. 1/2, Kluwer Academic Publishers, Netherlands, pp. 7 - 35, 2000.

[5] J. F. Miller, A. Thompson, P. Thomson, T. C. Fogarty (eds.). Proceedings of the 3rd International Conference on Evolvable Systems: From Biology to Hardware (ICES 2000), Lecture Notes in Computer Science, vol. 1801, Springer-Verlag, Berlin, 2000.

[6] J. Nijhuis, B. Hofflinger, A. Schaik and L. Spaanenburg. "Limits to Fault-Tolerance of a Feedforward Neural Network with Learning", Digest FTCS, pp. 228-235, June 1990.

[7] A. Thompson. "An evolved circuit, intrinsic in silicon, entwined with physics", in T. Higuch, M. Iwata, W. Liu (eds.), Proceedings of The 1st International Conference on Evolvable Systems: From Biology to Hardware (ICES96), Lecture Notes in Computer Science, vol. 1259, Springer-Verlag, Heidelberg, pp. 390 - 405, 1997

[8] A. Thompson. "On the Automatic design of Robust Electronics through Artificial Evolution", in M. Sipper, D. Mange, A. Pérez-Uribe (eds.), Proceedings of The 2nd International Conference on Evolvable Systems: From Biology to Hardware (ICES96), Lecture Notes in Computer

Science, vol. 1478, Springer-Verlag, Heidelberg, pp. 13- 24, 1998

[9] A. Thompson and P. Layzell. "Evolution on Robustness in an Electronics Design", in J. Miller, A. Thompson, P. Thomson, T.C. Fogarty (eds.), Proceedings of The 3rd International Conference on Evolvable Systems: From Biology to Hardware (ICES 2000), Lecture Notes in Computer Science, vol. 1801, Springer-Verlag, Heidelberg, pp. 218 - 228, 2000.

[10] A. Thompson. "Hardware Evolution - Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution", Springer-Verlag, London, 1998.

[11] B.E. Segee and M. J. Carter. "Fault Tolerance of Pruned Multilayer Networks", Digest IJCNN, pp. II-447-452, 1991.

[12] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe and A. Stauffer, "A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems", IEEE Transactions on Evolutionary Computation vol. 1(1) pp.83-97, 1997.

[13] M. Sipper, D. Mange and A. Pérez-Uribe (eds.). Proceedings of the 2nd International Conference on Evolvable Systems: From Biology to Hardware (ICES98), Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 1478, 1998.

[14] A. Stocia, D. Keymeulen, R. Zebulum, A. Thakoor, T. Daud, G. Klimeck, Y. Jin, R. Tawel and V. Duong. "Evolution of analogue circuits on Field Programmable Transistor Arrays", in J. Lohn, A. Stocia, D. Keymeulen and S. Colombano, Proceedings of The 2nd NASA/DoD Workshop on Evolvable Hardware (EH-2000), 2000, CA IEEE Computer Society Press.