Evolution of fault-tolerant and noise-robust digital designs

M. Hartmann and P.C. Haddow

Abstract: Artificial evolution has been shown to generate remarkable systems of exciting novelty. It is able to automatically generate digital circuit designs and even circuits that are robust to noise and faults. Extensive experiments have been carried out and are presented here to identify more clearly to what extent artificial evolution is able to generate robust designs. The evolved circuits are thoroughly tested whilst being exposed to noise and faults in a simulated environment and the results of their performance are presented. The evolved multiplier and adder circuits show a graceful degradation as noise and failrate are increased. The functionality of all circuits is measured in a simulated environment that to some extent takes into account analogue electronic properties. Also included is a short overview of some recent work illustrating the robustness and tolerance of bio-inspired hardware systems.

1 Introduction

Designing and manufacturing electronic chips is a complex task, growing more so as the industry keeps up with what is commonly referred to as Moore's law. The 'red brick wall', a metaphor used to illustrate the point where current technology cannot be pushed further, seems to be repeatedly pushed slightly forward in time [1]. If we at some point stand face-to-face with this brick wall, a technology transition or alteration will hopefully help us pass it.

Whether we push current technology or transfer to new ones, there is still the task of doing the actual design. The increasing density of modern chips and demand for more complex designs presents a challenging task. Massive resources are spent world-wide on all the different phases in the complete design cycle. In addition, we face a design technology gap as engineers and tools are unable to make efficient use of all the resources available on state-of-the-art chips.

Another issue of electronic design is the need for fault and noise tolerance. The high density of chips increases the possibility of failing components and the complexity of designs increases the probability of human errors. The acceptance for faults is diminishing as the market demands increasingly more reliable systems. The need for fault-tolerant designs and management of noise are stated amongst the long-term (2008 through 2016) grand challenges in [2].

The above may be summarised as two key demands: novel automated design and fault and noise tolerance. Recently, a growing research field has started exploring new solutions to these problems, the field of bio-inspired hardware design. The main motivation within bio-inspired hardware design comes from the observation of natural systems. Biological organisms such as humans are in many ways extremely complex, yet nature has managed to evolve creatures that utilise their physical, chemical, electrical and biological properties in intricate complex dynamical ways. In addition, biological organisms are tolerant to faults on many levels in that they keep on functioning even though cells or sometimes even entire limbs fail.

The focus of the paper is a subfield of bio-inspired hardware systems known as evolvable hardware (EHW). EHW can be viewed as covering the phylogenetic [3] part of biologically inspired systems, i.e. temporal evolution, popularly termed Darwinism. In this paper, small electronic circuits are extrinsically evolved, i.e. they are evolved in a computer-simulated environment.

2 Background

Most digital systems are reliable as long as all events that occur are expected and within specifications. However, once something unforeseen happens, i.e. signal variations deviate outside the specified voltage range for a logic 1 or 0 or external events deviate from those specified to be tolerated by the system, a digital system is extremely vulnerable. For instance, a single unexpected logic inversion can halt an entire system. The issue with engineering-approaches to the problem of noise robustness and fault tolerance is the fact that they will always be limited by the view and insight of the engineer.

Several important contributions to fault tolerance, fault detection and fault repair within the field of bio-inspired hardware exist. For instance, embryology-inspired work conducted at York [4] and L'Ecole Polytechnique Fédérale de Lausanne (EPFL) [5] experiments with multi-layered hardware organisms where each cell contains the complete genotype of the circuit. Through repeated cell divisions, a circuit develops from a single cell into a full-grown phenotype. Principles from biological immune systems have been used to achieve fault detection and repair [6]. Fault tolerance and error detection for robots are explored in [7, 8] using embryonic arrays and artificial immune systems.

IEE Proceedings online no. 20040014

doi: 10.1049/ip-cdt:20040014

Paper first received 12th May and in revised form 10th October 2003

The authors are with the Norwegian University of Science and Technology, Department of Computer and Information Science, Sem Sælands vei 7-9, 7491 Trondheim, Norway

Garvie and Thompson are evolving hardware with built-in self-test behaviour [9]. As early as 1995, Thompson evolved fault-tolerant electronic control systems [10]. In [11] a genetic representation is constructed that allows a spacecraft controller to exploit faults. At NASA evolution of field-programmable transistor arrays (FPTA) is used to obtain fault tolerance [12]. Evolutionary fault-recovery techniques on the FPTA have also been investigated [13]. In addition, circuits have been evolved to be tolerant to temperature changes [14].

The work herein originated from [15], where MUX gates were simulated in an abstract environment, but without simulating the logical behaviour of any specific technology. 2-bit multipliers were evolved in increasingly noisy environments and then tested when subjected to noise or stuck-at faults. The messiness i.e. the noise of the environment and the seemingly confusing architectures exploiting it, seemed to increase fault tolerance to stuck-at faults and even produce some surprisingly compact 6-gate solutions. This fault tolerance emerged implicitly, that is, without faults being applied during evolution in order to prepare the circuits for such events. In [16] it was found that stronger fault tolerance could be obtained by explicitly evolving the multipliers for fault tolerance. The fitness function was taking into account the actual fault tolerance during evolution. The increased fault tolerance came at a higher computational cost. It was also discovered that evolution would tend to generate small circuits. It was assumed that this was an effective way of reducing the amount of noise owing to the fact that noise was applied on a per gate basis. This effect will be discussed in Section 5.

The work was continued [17] using a new parameterised technology simulator. The experiments demonstrated the ability of evolution to generate fault tolerant multipliers in a simulation of CMOS technology. Transferring to a more realistic simulation abandoned some of the messiness, as noise propagation is suppressed owing to the sigmoid behaviour of CMOS gates. However, no explicit stages were added to the gates that would further enforce the signals to keep within the digital thresholds. The experiments were extended to cover 2-bit adders in [18]. The MUX gates were abandoned, and replaced by the less complex AND, NAND, OR, NOR and NOT gates, as well as allowing connections to VCC and GND (logic 1 and 0 respectively). In addition, the completed evolved circuits were thoroughly tested to verify the performance with regard to fault tolerance and noise robustness.

The experiments described in this paper are based on a rewritten improved simulator. The simulator described in [18] was discovered to have a stability issue, and could get stuck in time-consuming loops waiting for two time-steps to produce the same output in a gate (which would not always be the case when noise or faults were applied). Since the time domain was not a part of the simulations anyway, it was completely abandoned to simplify the rewritten simulator. The experiments herein have been conducted on a parallel cluster, resulting in experiments covering a wider range of noise and faults. This makes the experimental data more statistically significant and clarifies the capabilities of evolution to cope with different amounts of noise and faults. It was decided to use a more sophisticated genetic algorithm (GA) since the previously used GA was suspected to have problems with high levels of noise and faults.

3 Simulation

The current simulator is a rewritten version of the one described in [18]. Our feed-forward analogue simulator

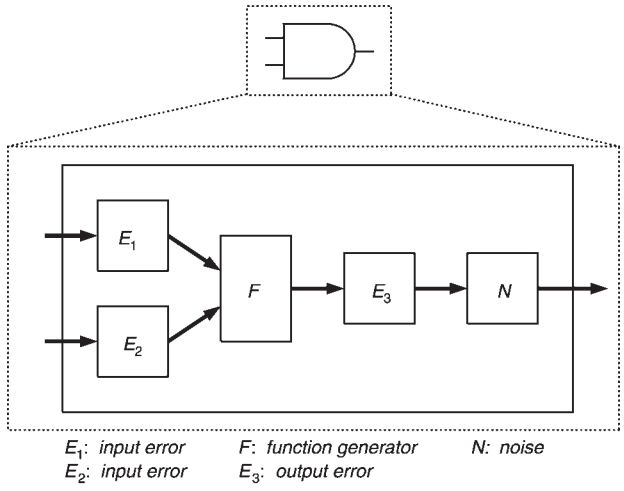

Fig. 1 *Model of our two-input gate*

allows simple, yet for our purposes sufficiently realistic, modelling of a digital circuit including analogue noise. With regard to timing, the simulator is completely combinatorial. Thus, one single simulation represents only one specific moment in time. The advantage of this approach is that many experiments can be run without requiring huge amounts of computational resources, which would be the case with a state-of-the-art simulator (e.g. SPICE). Additionally, it allows tuning to different technologies.

Currently, a sigmoid approximation to 5 V CMOS technology is the core of our gate model. The output of a gate is calculated as a function of the sigmoid approximation. This function corresponds to F in Fig. 1, a depiction of the gate model. E_1, E_2 and E_3 can generate one of the supported errors or let the signal propagate through without error. The probability of error is preset for each experiment, while the type of error is random with equal probability for each of the three possible faults. The possible faults are of type stuck-at errors, floating output and partly random output. In addition, there is support for inducing signal noise at each gate output. Input or output stuck-at errors cover the case of short-circuit to power or ground, or the cases of inter-signal short-circuits that behave as stuck-at errors when observed in a combinatorial and timeless domain. Floating output errors cover cases where the output is completely random, while partly random output covers the case where the output is correct for one logical value while random for the other logical value, e.g. logical 1 is represented as 1 whilst logical 0 is represented as a random number from 0 to 1.

The output noise N in Fig. 1 is noise that is superimposed on the output signal to approximate errors that are not explicitly a part of the model, e.g. thermal noise, radiation, power supply noise, component variance and cross talk. Noise has no time-related effects, but is simply implemented as random numbers within a specified range (in per cent of the complete signal range).

Interested readers are encouraged to look up further details on the simulator in [18].

4 Circuit evolution

Earlier work [15–18] has been based on a simple yet effective evolutionary strategy (ES), namely $(1 + \lambda)$. This brute force approach has been shown to be efficient in digital circuit evolution when combined with neutral genetic drift [19]. However, experimentation has indicated that the ES experiences trouble when evolution has to cope with

what	label	type	input A	input B
input	0			
input	1			
input	2			
input	3			
gate	4	AND	3	1
gate	5	NOT	4	
gate	6	AND	0	0
gate	7	AND	2	0
.				
.				
.				
gate	38	VCC		
gate	39	OR	4	9
gate	40	NOR	30	22
gate, output	41	AND	15	27
gate, output	42	OR	37	34
gate, output	43	NAND	38	24

Fig. 2 *Example genotype*

much noise and many faults. The unforgiving rejection of all but the best individual does not suit the random affliction of faults and noise.

The experiments herein are all based on evolution using a tournament selection genetic algorithm (GA). Tournament selection allows easy adjustment of the selection pressure and is only sensitive to the relative difference between fitness values. In addition, it is often used in conjunction with noisy fitness functions [20]. As will become clear during the remainder of this paper, the same individual or circuit may not receive the same fitness score if tested several times. The reason for this is the probabilistic nature of the simulations with regard to faults and noise. Because of this the evaluation of fitness is regarded as noisy.

Our genotype is a netlist. An example of the representation of a circuit is shown in Fig. 2. The corresponding circuit is depicted in Fig. 3. Connections refer to labels of either the inputs of the circuit (0 to 3) or the output of one of the gates in the circuit (4 to 43). The last gates in the genotype representation are considered to be connected to the external outputs of the circuit (41 to 43). Allowed elements are NOR, NAND, OR, AND, NOT, connection to VCC and connection to GND. Mutations are applied at the gate level. If a gate is mutated, either one of two incidents can occur with an equal chance. In the first incident, one of the gate inputs is randomly selected. That input is connected to the output of a random gate in the circuit prior to the mutated gate in the netlist, thus ensuring that the circuit stays strictly combinatorial. In the second incident, the type of the mutated gate is changed to a random type among the set of allowed elements.

The goal of evolution is to generate a circuit that successfully produces the correct mapping between input and output vectors. The target behaviour is specified by a truth table. Under fitness evaluation, a circuit is subject to several noise and/or fault vectors while testing the complete set of possible input vectors. The analogue output values of the circuit are rounded to their closest logical value and then compared to the target truth table.

The fitness function is expressed in (1). A circuit C (an individual) is tested against the target truth table (T) a number of times (TPI) under different environments. Noise and fault probabilities are used to generate the different environments m for each test. The average of all tests is computed to yield a penalty for the number of incorrect output bits. Thus, the fitness is effectively the negative of the average hamming distance between the measured function of the evolved circuit and the target function truth table.

$$F = -\left(\frac{\sum_{n=1}^{TPI} diff(C_m, T)}{TPI}\right) \tag{1}$$

where F is the fitness of the individual, TPI is the number of tests per individual, $diff()$ is the number of incorrect output bits, C_m is the circuit in environment m and T is the target truth table.

In order to get a picture of performance of the evolved circuits one should keep in mind that 2-bit multipliers have 64 output bits in their truth table, while 2-bit adders have 48.

5 Experiments and results

Two target combinatorial circuits were chosen: 2-bit multipliers and 2-bit adders, the former being relatively more parallel than the latter. Circuits were limited to 40 gates. Since our circuits evolve to sizes less than 20 gates, this leaves more than 50% of the genome to be genes without direct phenotypic influence. This results in a genotype size of 44 when evolving 2-bit multipliers or adders (4 inputs). The population size was set to 20. Mutation rate was 5% (per gate) and the crossover rate was set to 20%. Elitism was deployed, so that the best individual of a generation was transferred untouched to the next generation. This elitism selects a random individual among the best ones, thus ensuring neutrality (which was shown to be important in [19]). The tournament selection GA was set to use a group size of 3 and a selection probability of 70%.

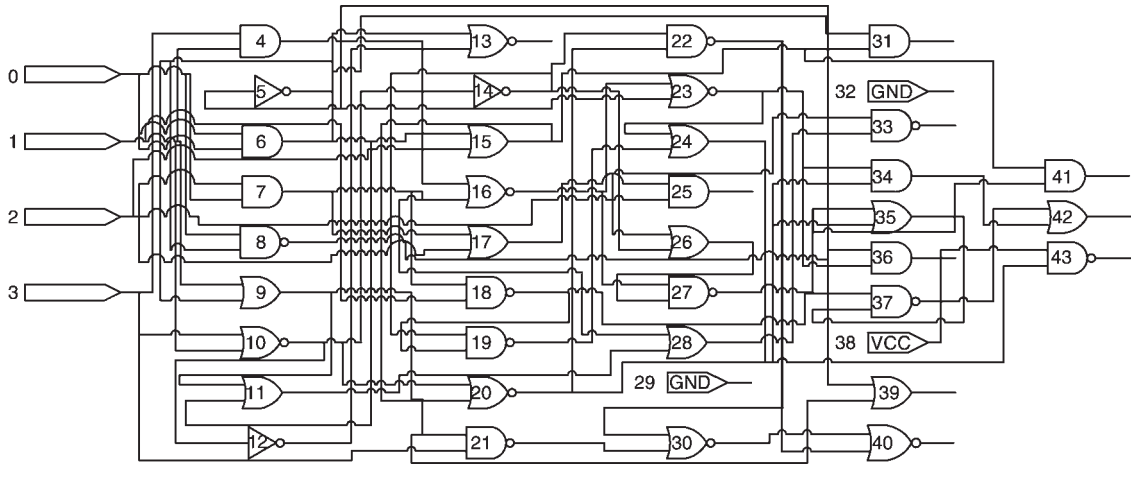

Fig. 3 *Schematic of example circuit*

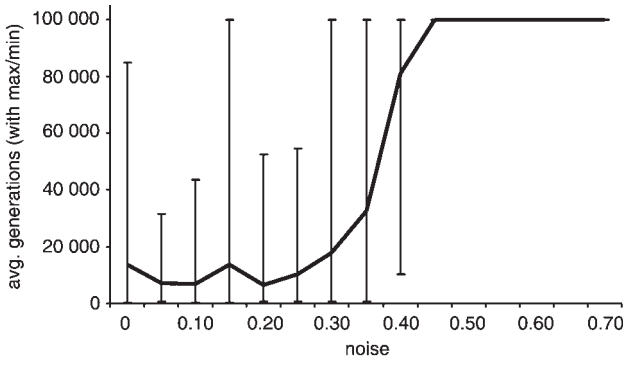

Fig. 4 *Increasing noise: multipliers (generations)*

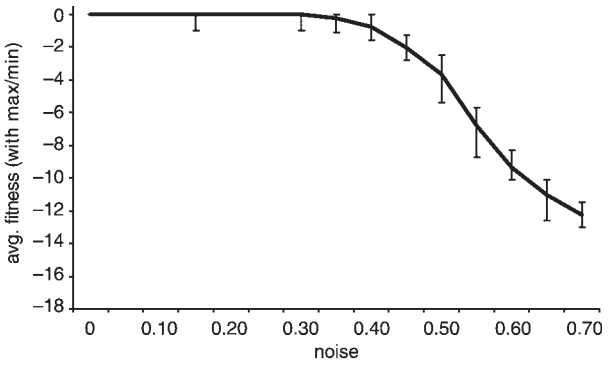

Fig. 6 *Increasing noise: multipliers (final fitness)*

This means that to select a parent, three individuals are randomly selected and compared. There is a probability of 70% that the one with the best fitness is chosen as a parent. Otherwise, a parent is chosen randomly from the group of three. Finally, the number of generations was limited to 100 000. Such a limitation was necessary to stop evolution from running for extremely long times when noise and/or failrate is high.

Obviously, if one of the gates connected to the circuit output fails the output is bound to be distorted and be incorrect unless it by chance happens to produce the correct logic value. Thus, evolution has no way to influence the impact of such failure. As argued in [18], the output gates are, therefore, protected against faults, but not noise, in the following experiments.

For every variation of noise percentage and failrate 20 separate evolutionary runs were performed so that the average behaviour over several runs could be observed. Owing to the high number of runs required in the span over all variations of noise and failrate the experiments were conducted on a 40-node Beowulf-type parallel cluster [21] using OpenPBS [22]. Every circuit was also tested ten times ($TPI = 10$) in these experiments to even out the randomness of applied noise and faults. Thus, the number of tests run for each value of noise and/or failrate was 10 tests per individual $\times$ 20 circuits evolved $=$ 200.

5.1 Robustness to noise

Evolution was carried out with multipliers and adders when applying up to 70% noise. Figures 4 and 5 show the relationship between noise and the required number of generations for multipliers and adders respectively. The vertical bars show the two most extreme individuals, those that for their amount of noise, require the most and the least number of generations to evolve (among the 20 separate runs that were conducted for each value of noise). The noise values shown along the x-axis represent all noise values for which experiments were conducted.

As was suspected from earlier experiments, it is clear that evolution can cope with low values of noise. The sigmoid behaviour of the gates may explain some of this effect, as small variations in the signals will be dampened. Still an increase in the number of required generations would be the intuitive thing to expect. However, there is a tendency for a drop in the number of generations required to complete evolution up to 15%–20% noise. The added noise may actually aid evolution in exploring the search space more efficiently. A general observation is that adders are harder to evolve than multipliers. As pointed out in [18], the lack of an XOR gate may explain this.

Another important observation is the great variation in the number of generations required to evolve the circuits. Some individuals are evolved extremely quickly, while others take a long time or never complete, even with the same amount of noise. This is the case until the noise is so dominant that evolution always runs for 100 000 generations. Such a variation is obviously rooted in the probabilistic nature of GAs, as random chance plays a big part in the efficiency of evolution.

Since experiments are halted if perfect fitness is not achieved within 100 000 generations, the average fitness of the circuits at the end of evolution is expected to fall as noise increases. This can be observed in Figs. 6 and 7, where the average fitness of completed circuits is depicted for increasing amounts of noise. Again, the vertical bars represent the extremes, in this case the worst and the best individuals. Certain multipliers achieve perfect fitness up to 40% noise, while for adders some individuals complete with perfect fitness with up to 35% noise.

Because of the random affliction of the applied noise, some circuits could be more lucky than others even though they are tested multiple times in an effort to even out the

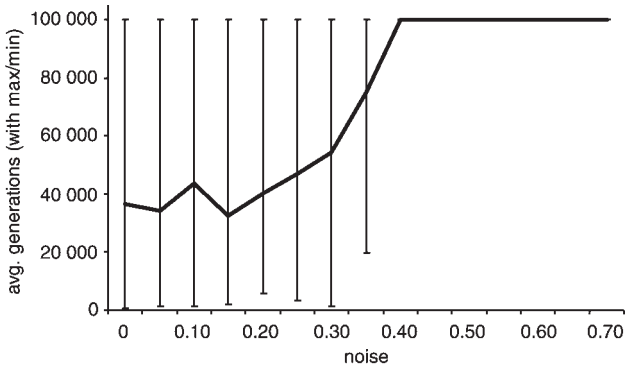

Fig. 5 *Increasing noise: adders (generations)*

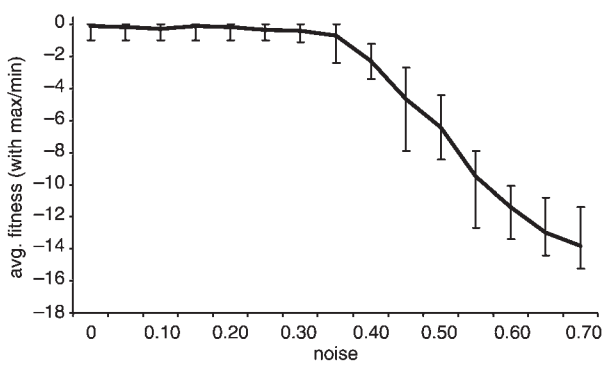

Fig. 7 *Increasing noise: adders (final fitness)*

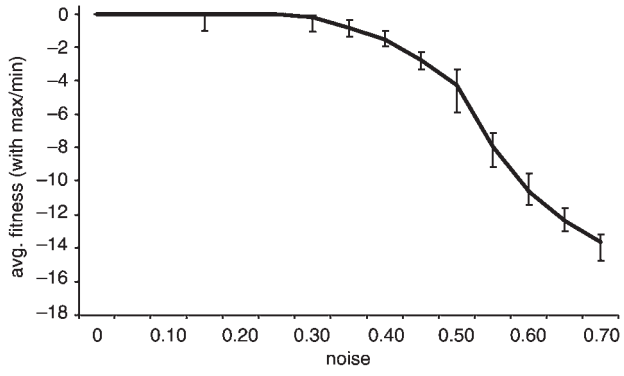

Fig. 8 *Increasing noise: multipliers (actual performance)*

impact of noise. To investigate the impact of the probabilistic nature of the evolutionary system all evolved circuits were subjected to extensive testing after evolution was completed. Each evolved circuit was tested 1000 times with random noise within the noise range specified for that circuit. The results are shown in Figs. 8 and 9. It is obvious that by doing thorough testing, one can observe that the circuits may be lucky during the 10 tests performed during evolution. When tested extensively afterwards, some multipliers work perfectly up to 25% noise, while there are adders that can function correctly with up to 20% noise.

A final observation that can be made from the experiments is a trend appearing in the number of gates each circuit uses. Even though each individual consists of 40 gates, the number of gates actually active (contributing to the output) in each circuit varies. The variation in the average number of gates for increasing noise is shown in Figs. 10 and 11, with vertical bars representing the largest and smallest circuits in each case. It is obvious, as was also observed in [15], that evolution prefers small circuits in noisy environments. Since noise is applied per gate, randomness is kept to a minimum by using few gates. In extreme cases of noise, it is obviously more effective to use fewer gates than necessary for complete functionality than struggling for full functionality but being overwhelmed by noise. The size of the circuits seems to be in the range of 16 to 18 gates within the range of noise where circuits prove to be perfectly functional during the extensive (1000) tests.

5.2 Tolerance to gate failures

Evolution of multipliers and adders was also conducted when random gate failures were allowed to strike all but the output gates of the circuits. The failrate was increased in steps from 0% up to 14%. Figures 12 and 13 show the relationship between the failrate and the required number of generations.

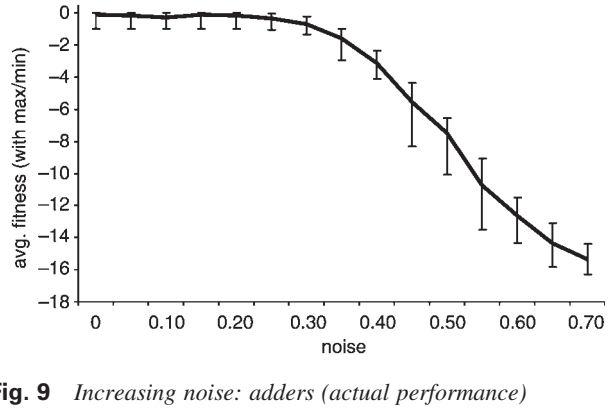

Fig. 9 *Increasing noise: adders (actual performance)*

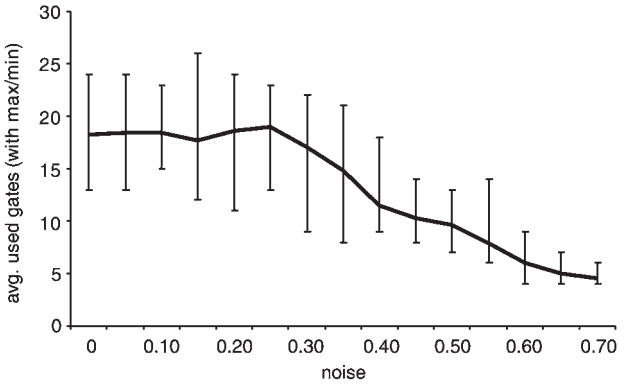

Fig. 10 *Increasing noise: multipliers (number of gates)*

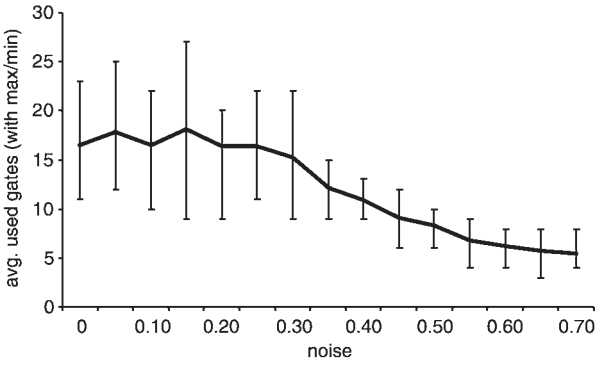

Fig. 11 *Increasing noise: adders (number of gates)*

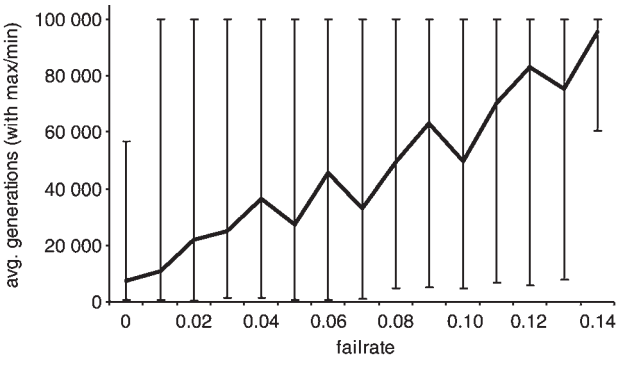

Fig. 12 *Increasing failrate: multipliers (generations)*

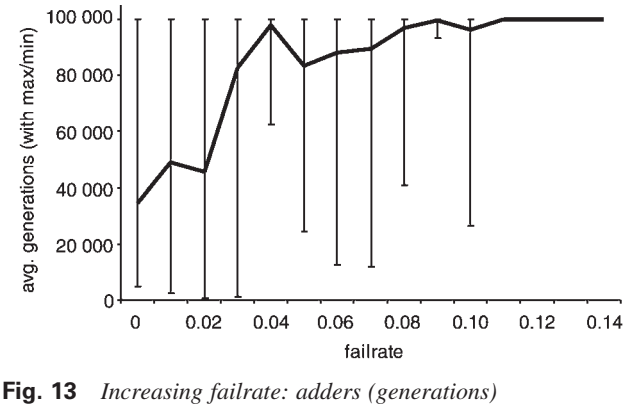

Fig. 13 *Increasing failrate: adders (generations)*

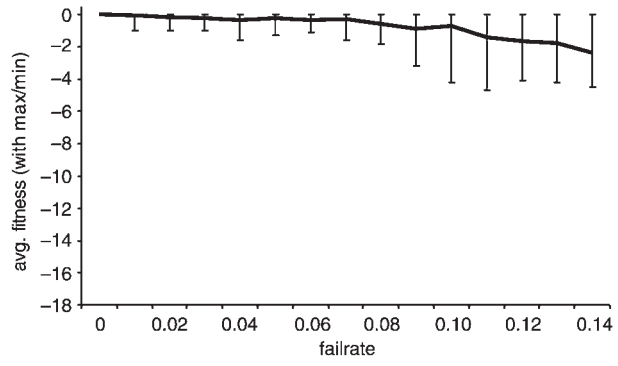

Fig. 14 *Increasing failrate: multipliers (final fitness)*

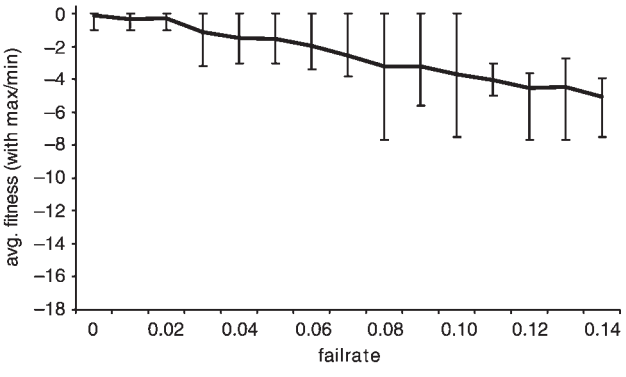

Fig. 15 *Increasing failrate: adders (final fitness)*

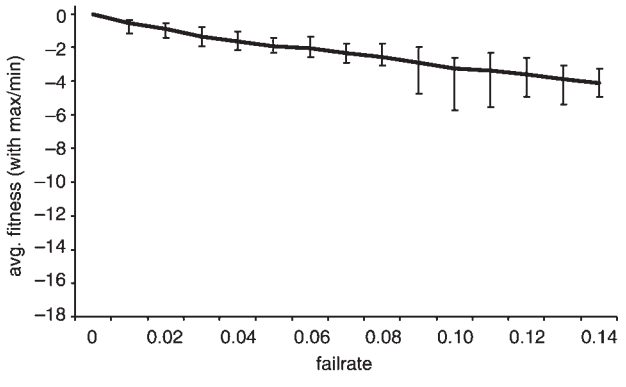

Fig. 16 *Increasing failrate: multipliers (actual performance)*

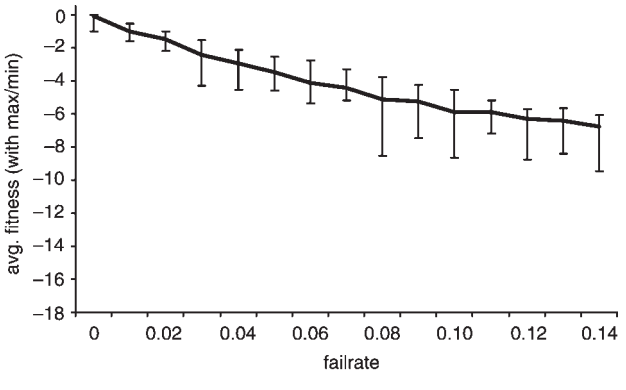

Fig. 17 *Increasing failrate: adders (actual performance)*

As opposed to the observations made when evolving circuits in noisy environments, the gate failures do not seem to aid the evolutionary process. The phenomenon of gate failure aiding evolution could to some degree be observed in earlier experiments, and the use of a tournament selection GA may explain why a similar phenomenon does not seem to occur in these experiments. The fitness achieved as evolution completed can be observed in Figs. 14 and 15. The best multipliers and adders achieved perfect fitness with up to 14% and 10% failrate, respectively.

A verification of the circuits evolved for fault tolerance was performed, similar to that performed with the circuits evolved for noise robustness. The results are shown in Figs. 16 and 17. Eventhough the circuits evolved prove very resilient to faults, none of them is able to work 100% correctly when gates fail. This cannot be expected, since during some of the 1000 tests very many gates may actually fail. Theoretically speaking, there is indeed a chance that all gates (except the protected output gates) fail. Keeping this in mind it is mind-boggling that evolution is able to generate multipliers that on average produce only two incorrect output bits for all possible input vectors when there is a 6% chance for each gate to fail. Similarly, one multiplier actually produces only 2.34 incorrect bits on average when there is an 11% chance for every gate to fail.

As with experiments with variations in noise, a trend in the number of used gates can be seen when varying the probability of gate failure. The gate usage is shown in Figs. 18 and 19. It is obvious that evolution is forced to reduce the number of gates much earlier than in the case where noise is being applied.

5.3 Combinations of noise and gate failures

Some of the above results indicate that the application of noise may in some situations actually improve the efficiency

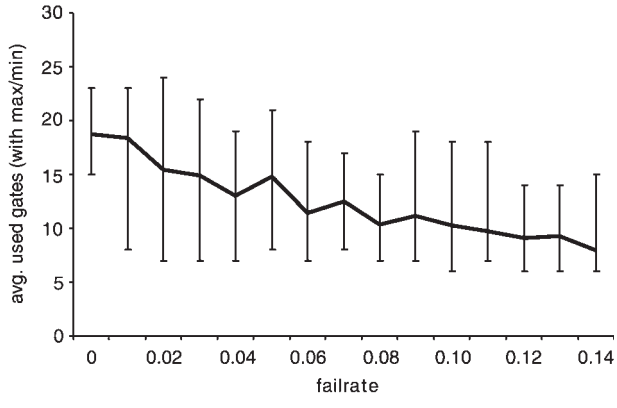

Fig. 18 *Increasing failrate: multipliers (number of gates)*

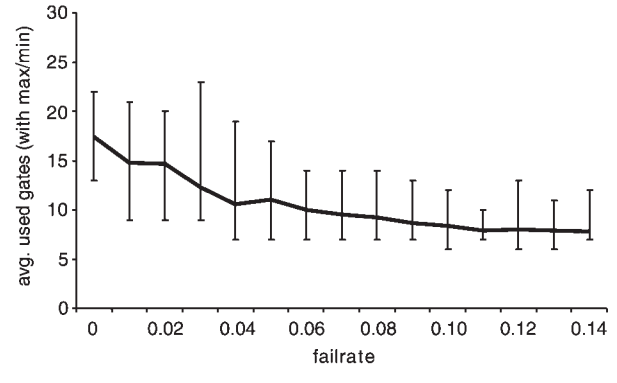

Fig. 19 *Increasing failrate: adders (number of gates)*

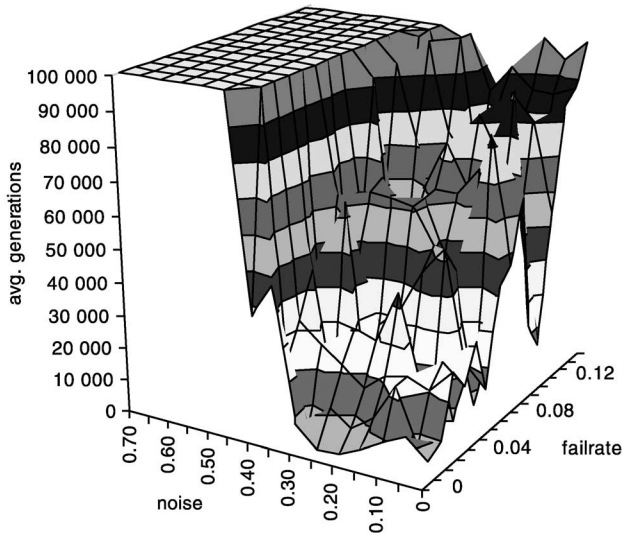

Fig. 20 *Increasing noise and failrate: multipliers (generations)*

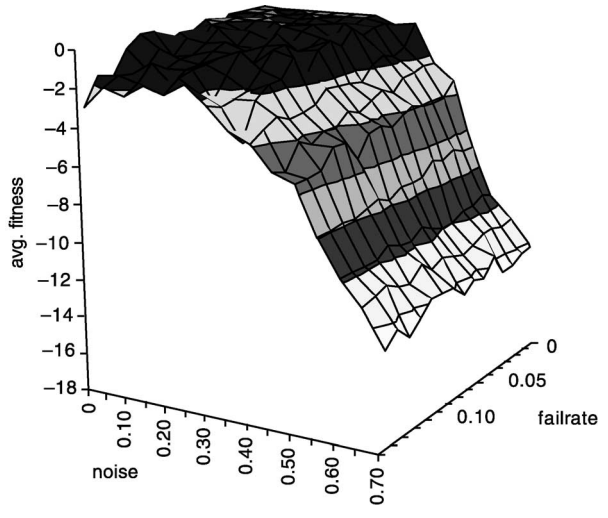

Fig. 21 *Increasing noise and failrate: multipliers (final fitness)*

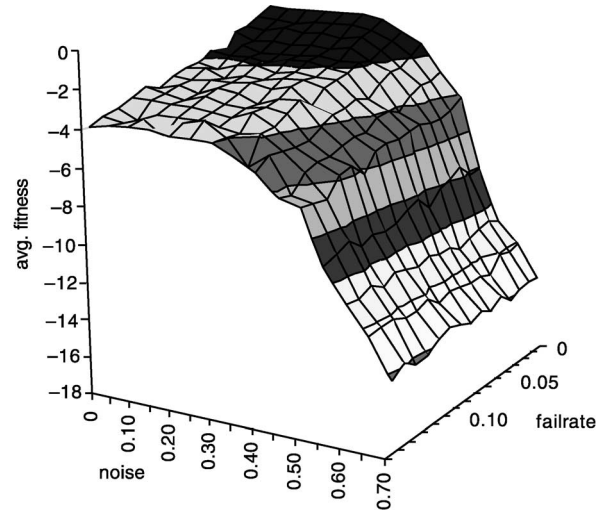

Fig. 22 *Increasing noise and failrate: multipliers (actual performance)*

of evolution. Perhaps this could be exploited by evolution when evolving fault-tolerant circuits, as was assumed to be observed in [18]. A limited set of experiments was conducted for multipliers with varying amounts of noise and gate failures. Due to the massive numbers of experiments required to span all possible variations of noise and failrate, only five evolutionary runs were conducted for each case (as opposed to 20 experiments when only considering variations in either noise or failrate).

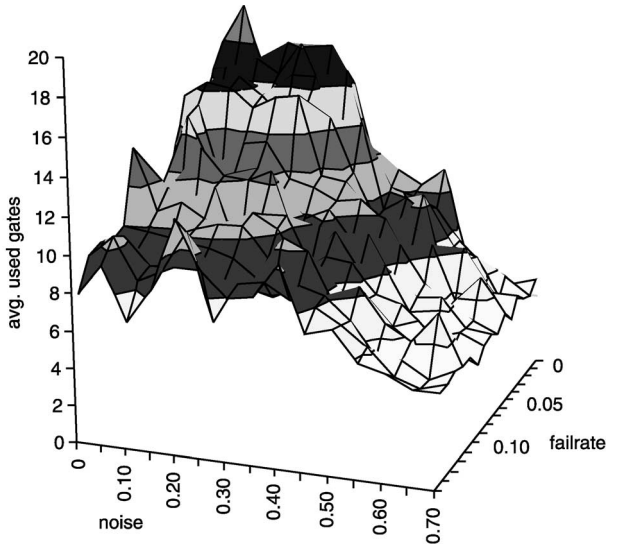

Fig. 23 *Increasing noise and failrate: multipliers (number of gates)*

The average number of generations required can be observed in Fig. 20. The extreme individuals that were displayed earlier are not shown here for reasons of visibility. Some improved efficiency can be observed as combining noise with gate failures yields fewer numbers of generations when compared to experiments with only gate failures. For instance, an area around about 35% noise and 5% failrate stands out as very low with regard to the required number of generations.

The fitness achieved at the end of evolution is shown in Fig. 21, while the fitness achieved when doing thorough testing afterwards is depicted in Fig. 22. In both cases, one can observe that the impact of the failrate is strongest when noise levels are low, but seems to diminish as noise levels increase. The average number of gates used can be seen in Fig. 23.

6 Conclusions

The experiments show that for limited size circuits, evolution is capable not only of automatic generation of 100% functional circuits, but also shows potential to generate circuits that are robust to noise and/or tolerant to failing gates. Such circuits could be combined with more traditional fault-tolerance schemes for increased performance. For instance, classical redundancy techniques could use evolved structurally different but functionally equivalent (digitally speaking) circuits as the redundant elements. This would give a multilayered tolerance with the benefits of both bio-inspired novelty and engineered resilience.

The circuits could also provide valuable information if one seeks ways to implement strong fault tolerance without complete redundancy. The circuits illustrate the ability of evolution to generate novel designs with beneficial properties, completely unlike the solutions an engineer would come up with.

One important issue of evolved systems is the scalability problem. It is common to experience a relation of exponential nature between the size of the genotype and the time it takes to find a correct solution. A few runs were made using the same evolutionary system as with the

experiments herein, and a 3-bit multiplier was evolved using an average of about half a million generations (of 5 runs without noise or faults, with 233 325 as lowest and 666 197 as highest). These circuits were about 75 gates in size, with a genotype of 200 gates. Effectively, the number of required generations increased with a factor of 30, while the genotype increased with a factor of 5.

One interesting bio-inspired effort to cope with the scalability problems of GAs is artificial development. A discussion of artificial development techniques is outside the scope of this paper. The interested reader is encouraged to proceed to recent inspiring work such as [23–28].

7 References

1 Semiconductor Industry Association. *The National Roadmap for Semiconductors*, 2001, http://public.itrs.net/
2 Semiconductor Industry Association. *The National Technology Roadmap for Semiconductors*, 2002 Update, http://public.itrs.net/
3 Sanchez, E., Mange, D., Sipper, M., Tomassini, M., Perez-Uribe, A., and Stauffer, A.: 'Phylogeny, ontogeny and epigenesis: Three sources of biological inspiration for softening hardware', *Lect. Notes Comput. Sci.*, 1997, **1259**, pp. 35–54
4 Ortega, C., and Tyrrell, A.: 'Reliability analysis in self-repairing embryonic systems'. Proc. 1st NASA/DoD Workshop on Evolvable hardware (EH), 1999, pp. 120–128
5 Mange, D., Sipper, M., Stauffer, A., and Tempesti, G.: 'Toward self-repairing and self-replicating hardware: the embryonics approach'. Proc. 2nd NaSA/DoD Workshop on Evolvable Hardware (EH), Palo Alto, CA, 13–15 July 2000, pp. 205–214
6 Bradley, D., Ortega-Sanchez, C., and Tyrrell, A.M.: 'Embryonics + immunotronics: A bio-inspired approach to fault-tolerance'. Proc. 2nd NASA/DoD Workshop on Evolvable Hardware (EH), Palo Alto, CA, 13–15 July 2000, pp. 215–224
7 Jackson, A.H., Canham, R., and Tyrrell, A.M.: 'Robot fault-tolerance using an embryonic array'. Proc. 2003 NASA/DoD Conf. on Evolvable Hardware, Chicago, IL, 9–11 July 2003, pp. 91–100
8 Canham, R., Jackson, A.H., and Tyrrell, A.: 'Robot error detection using an artificial immune system'. Proc. 2003 NASA/DoD Conf. on Evolvable Hardware, Chicago, IL, 9–11 July 2003, pp. 199–207
9 Garvie, M., and Thompson, A.: 'Evolution of self-diagnosing hardware', *Lect. Notes Comput. Sci.*, 2003, **2606**, pp. 238–248
10 Thompson, A.: 'Evolving fault tolerant system'. Proc. 1st IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA), Sheffield, UK, 12–14 September 1995, pp. 524–529
11 Lohn, J., Larchev, G., and DeMara, R.: 'A genetic representation for evolutionary fault recovery virtex fpgas', *Lect. Notes Comput. Sci.*, 2003, **2606**, pp. 47–56
12 Keymeulen, D., Zebulum, R.S., Jin, Y., and Stoica, A.: 'Fault-tolerant evolvable hardware using field-programmable transistor arrays', *IEEE Trans. Reliab.*, 2000, **49**, (3), pp. 305–316
13 Zebulum, R., Keymeulen, D., Duong, V., Guo, X., Ferguson, M.I., and Stoica, A.: 'Experimental results in evolutionary fault-recovery for field programmable analog devices'. Proc. 2003 NASA/DoD Conf. on Evolvable Hardware (EH), Chicago, IL, 9–11 July 2003, pp. 182–188
14 Stoica, A., Keymeulen, D., and Zebulum, R.: 'Evolvable hardware solutions for extreme temperature electronics'. Proc. 3rd NASA/DoD Workshop on Evoluable Hardware (EH), Long Beach, CA, 12–14 July 2001, pp. 93–97
15 Miller, J., and Hartmann, M.: 'Evolving messy gates for fault-tolerance: some preliminary findings'. Proc. 3rd NASA/DoD Workshop on Evolvable Hardware (EH), Long Beach, CA, 12–14 July 2001, pp. 116–123
16 Miller, J., and Hartmann, M.: 'Untidy evolution: Evolving messy gates for fault-tolerance', *Lect. Notes Comput. Sci.*, 2001, **2210**, pp. 14–25
17 Hartmann, M., Eskelund, F., Haddow, P.C., and Miller, J.F.: 'Evolving fault tolerance on an unreliable technology platform'. Proc. Conf. on Genetic and Evolutionary Computation (GECCO), New York, 9–13 July 2002, pp. 171–177
18 Hartmann, M., Haddow, P.C., and Eskelund, F.: 'Evolving robust digital design'. Proc. NASA/DoD Conf. on Evolvable Hardware (EH), Alexandria, VA, 15–18 July 2002, pp. 36–45
19 Vassilev, V.K., and Miller, J.F.: 'The advantages of landscape neutrality in digital circuit evolution', *Lect. Notes Comput. Sci.*, 2000, **1801**, pp. 252–263
20 Miller, B.L., and Goldberg, D.E.: 'Genetic algorithms, tournament selection, and the effects of noise', *Complex Syst.*, 1995, **9**, pp. 193–212
21 Cassens, J., and Constantinescu, Z.: 'It's magic: Sourcemage gnu/linux as a high performance cluster os'. Presented at LinuxTag Conf. 10-13 July 2003, Karlsruhe, Germany, http://ClustIS.idi.ntnu.no
22 http://www.openpbs.org, accessed September 2003
23 Kumar, S., and Bentley, P.J.: 'Biologically inspired evolutionary development', *Lect. Notes Comput. Sci.*, 2003, **2606**, pp. 57–68
24 Downing, K.L.: 'Developmental models for emergent computation', *Lect. Notes Comput. Sci.*, 2003, **2606**, pp. 105–116
25 Tufte, G., and Haddow, P.: 'Building knowledge into developmental rules for circuit design', *Lect. Notes Comput. Sci.*, 2003, **2606**, pp. 69–80
26 Tempesti, G., Mange, D., Petraglio, E., Stauffer, A., and Thoma, Y.: 'Developmental processes in silicon: an engineering perspective'. Proc. NASA/DoD Conf. on Evolvable Hardware (EH), Chicago, IL, 9–11 July 2003, pp. 255–264
27 Bentley, P.J.: 'Evolving fractal proteins', *Lect. Notes Comput. Sci.*, 2003, **2606**, pp. 81–92
28 Miller, J.F., and Thomson, P.: 'A developmental method for growing graphs and circuits', *Lect. Notes Comput. Sci.*, 2003, **2606**, pp. 93–104