Evolving Robust Digital Designs

Morten Hartmann, Pauline Haddow and Frode Eskelund
Dept. of Computer and Information Science
The Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
{mortehar,pauline,eskelund}@idi.ntnu.no

Abstract

Robust electronics is a challenge that the evolvable hardware field is addressing. This paper focusses on tolerance of faults where faults may arise through defects in the technology or unforeseen events. By relaxing the digital abstraction we enable evolution to find robust circuit architectures, exploiting non digital signal variations where necessary. Using a flexible technology and environment simulator, multipliers and adders are extrinsically evolved in noisy environments where gates may fail. The robustness of these evolved circuits are further tested in various noise and gate failure environments.

1 Introduction

Fault tolerance may be expected to gain more and more importance in the future. However, extremely harsh and changing environments, like outer space, force us to think about this issue today.

In nature, evolution is not prone to designs where every individual part's functionality is crucial for satisfactory behaviour. In digital design, on the other hand, each gate and its connections plays a crucial role in the overall circuit behaviour. This work explores this feature of nature by freeing gates from the perfect digital operation requirement and allowing evolution to create "almost digital" designs by interconnecting these imperfect digital gates. This concept was introduced by Miller and Hartmann [8] and is termed messy gates. Messy gates are able to operate without the digital thresholds, their non-perfect digital outputs being allowed to propagate through the circuit. Circuits evolved using messy gate components have evolved architectures that are tolerant to faults.

Earlier work of Miller and Hartmann [8, 9] on the messy gates approach may be said to be a proof of concept approach. That is, the model did not take into account any particular technology but more investigated the possibility of exploiting non-perfect digital gates to achieve fault tolerance. This work showed promising results and, as such, the model was extended to a technology specific model [5]. The simulator was parameterised enabling tuning to different semiconductor technologies. The simulator was also extended to gate level simulations, enabling faults to be applied at the gate level rather than component level (multiplexor). However, all experiments presented were based on multiplexor components for comparison with earlier work.

The main goals of the work presented herein are to verify features of our model, apply faults and noise to the gate level simulations, expand the application component to not only a multiplier but also an adder, conduct more extensive noise and fault testing and address the single point of failure issue (see section 5).

The motivation for this work is presented in section 2. Section 3 describes our gate model for 5V Complementary Metal Oxide Semiconductor (CMOS). Features of the evolutionary process applied to the messy gates are presented in section 4. Section 5 describes experiments and achieved results. Finally, section 6 concludes the paper and section 7 contains some directions for future work.

2 Motivation

Silicon is not a truly reliable technology. However, by using a digital abstraction we obtain a more robust platform against signal variation and noise whilst sacrificing much intrinsic complexity. That is, above and below the digital thresholds we are not concerned with signal variations.

Unforeseen events can easily prevent proper operation of a regular digital design. Why is this? The digital abstraction assumes that the technology will always operate within the specifications laid down by the abstraction mechanism and since silicon is not a truly reliable technology, deviations may be expected.

The work herein is concerned both with faults arising from defects imposed at the fabrication stage (or naturally present in whatever technology one might be investigating)

as well as faults that may occur during operation, whether static or dynamic in nature.

A number of noteworthy efforts have already been conducted within fault detection and repair based on the principles of biological development. In the embryology work conducted at York [2] and Ecole Polytechnique Fédérale de Lausanne (EPFL) [7], experiments have been conducted using FPGAs with extended Configurable Logic Blocks (CLBs) to contain a complete genotype of the circuit. Through repeated cell divisions a circuit develops from a single cell into a full-grown phenotype. An interesting approach was taken in [3] where principles of biological immune systems were adopted to attain fault tolerance. Work on achieving tolerance to temperature changes includes [12] and [10].

Inherent fault tolerance is present if the design is able to continue its operation undisturbed by fault inducing events, without the need for explicit mechanisms for fault detection and recovery. This may be achieved by robust ways of computation or by underlying fault detection and repair from within the technology. The work of Haddow and van Remortel [4] considered possibilities for achieving fault detection and repair from within the technology as well as more fault tolerant ways of distributing digital designs onto the technology based on the amorphous computing concept [1]. Hounsell and Arslan [6] have developed a fault tolerant hardware platform for the automated design of multiplierless digital filters. Tyrrell et al [13] have used evolutionary strategies to achieve redundancy thus providing inherent fault tolerance in the design.

The messy gate concept may be said to be a fault tolerant rather than fault detection and repair methodology. This tolerance is a tolerance to a less reliable technology. It may not only tolerate less than perfect gates but in fact uses evolution to exploit this *messiness* i.e. non perfect digital signals. Other work which exploited features of the technology includes the work of Thompson [11]. Here the focus was not so much on fault tolerance but more towards achieving unique solutions to difficult problems and, as such, illustrated the power of evolution.

The goal of using evolution together with messy gates is to evolve architectures which may incorporate some intricate redundancy where the importance of individual gates is reduced but that the priorities of different pathways play some crucial role. In this way evolution can exploit the messiness available to achieve architectures which may not achieve correct functionality when implemented on traditional technologies. However, our long term goal is to establish design methods for achieving robustness for future technologies where such non perfect digital gates exist and analogue signals may be allowed to propagate through the design.

3 Gate Model

The work presented in [5] introduced a model of messy gates based on CMOS technology. The model is now expanded to a more flexible parameterised model enabling specialisation towards different semiconductor technologies.

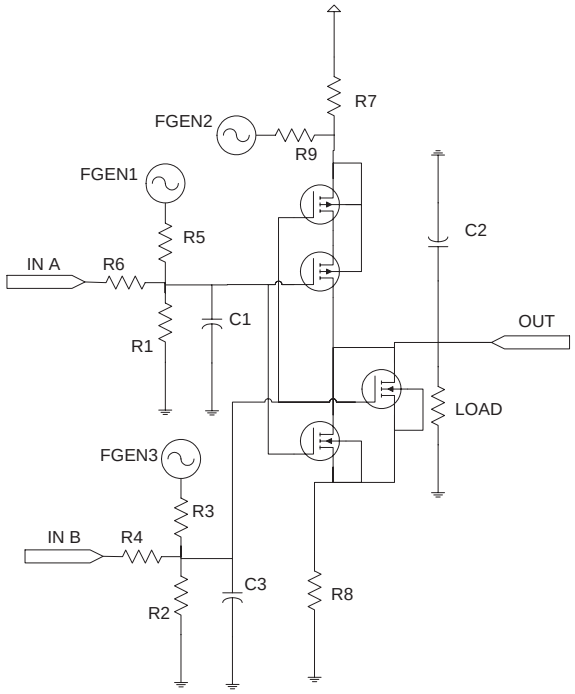

Figure 1. SPICE Transistor Level Schematic of NOR Gate

To parameterise our model for a given technology, each gate was first modelled at the transistor level including various error sources and simulated using the Simulation Program for Integrated Circuits Emphasis (SPICE). Figure 1 provides an example of our transistor level gate model for a NOR gate. As shown, the model incorporates three noise generators ($FGEN1$ through $FGEN3$), several capacitances ($C1$ through $C3$), current leaks ($R1$ and $R2$) and output load ($LOAD$). During simulation the gate was exposed to different configurations of capacitances, resistances, noise and faults. NAND, NOR and NOT gates were all simulated at analogue 5V CMOS.

The second stage was to find a way to represent the activation function of these gates in our model based on 5V CMOS. A sigmoid function in the range 0 to 1 was proposed. Different sloping sigmoid functions were generated, as shown in figure 2. The results from the SPICE NOT gate simulations indicated that slope A=15 was closest to the rise

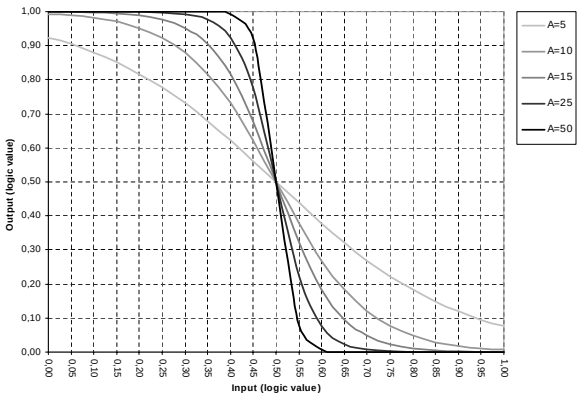

Figure 2. Sigmoid Functions for a NOT Gate

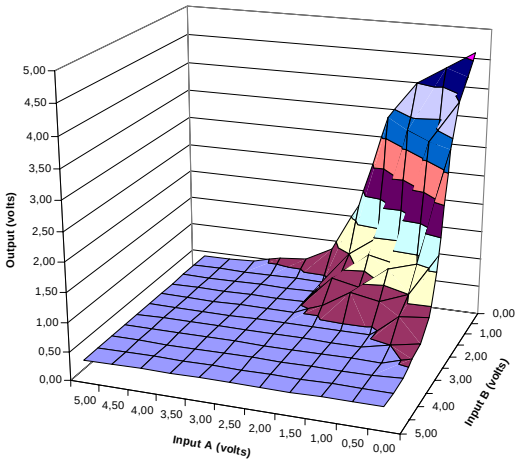

Figure 3. SPICE Simulated NOR Gate

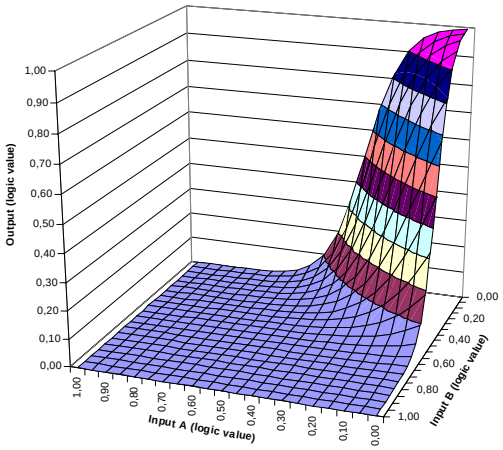

Figure 4. Modelled NOR Gate Behaviour

and fall behaviour of the NOT gate.

This sigmoid function was then used to tune other gate behaviour to an approximation of the rise and fall behaviour of the given gate in 5V CMOS. SPICE simulation of our transistor level gate model of a NOR gate, illustrated in figure 1, resulted in the rise and fall behaviour shown in figure 3. The process of tuning this behaviour to our representation may be described as follows. The OR of pairs of analogue input values (between 0 and 1) were used to generate input values to the sigmoid function i.e. x-axis values. From the sigmoid function, an appropriate output value could be found for the gate in 5V CMOS i.e. the value on the y-axis. These values were then used to generate an approximated NOR gate activation function as illustrated in figure 4. The OR of these values was taken rather than the NOR since the sigmoid function was an inverted function based on a NOT gate. To achieve an OR gate, we combined the models for NOR and NOT and similarly to achieve an AND gate we combined the models for NAND and NOT.

Approximations were subject to some limitations due to the fact that a look-up table replaced a full sigmoid function in order to speed up simulations. It should be noted that the smoothness displayed in figure 4 is only the core part of the gate model and more noisy behaviour like the one in figure 3 is likely to be displayed as noise is added. That is, the noise generated in the transistor level simulation contributed to the tuning of the parameters when generating the approximated sigmoid function. This approximation smoothes out the fluctuations of the noise present at the transistor level, and such fluctuations must be regenerated at the more abstract simulation level.

The model parameters are contained in two separate files: a Technology Specification file and an Evolutionary Algorithm (EA)/Simulator Configuration file. The Technology Specification file defines the technology to be used through its activation function table (sigmoid function approximation) as well as providing a list of gates available to the evolution process — see table 1. Each gate is defined in terms of four parameters. The EA/Simulator Configuration file, shown in table 2, contains parameters for the evolutionary algorithm and other simulator parameters.

The current simulator focusses on internal faults of type stuck-at errors, floating outputs and partly random output in addition to supporting induced signal noise. Input or output stuck-at errors cover the case of short-circuit to power or ground, or certain cases of inter-signal short-circuits. 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. It should be noted that the simulator uses a real number

Parameters	Range	Description
GATES	[1,∞]	Number of gate-types defined in this file
ACT_FUNC_ENTRIES	[1,∞]	Number of entries in the activation function table
ACT_FUNC_DATA	Floats	The activation function table
GATE	N/A	Start gate definition
NAME	Text	Name of gate
FUNCTION	Text	Function of gate
NUM_INPUTS	[0,∞]	Number of inputs in the gate
GATE	N/A	Start gate definition
NAME	Text	Name of gate
.	.	.
.	.	.
.	.	.

Table 1. Technology Specification File

internal representation from 0 to 1 to represent the CMOS voltage range.

As shown in figure 4, the sigmoid behaviour will allow propagation of analogue signals and yet be biased towards the digital endpoints of the analogue scale (0 and 1 in simulation). The model provides evolution with the possibility to exploit this non-digital feature to achieve more robustness in evolved designs.

Our gate model is illustrated in figure 5. E_1, E_2 and E_3 generate one of the supported errors or let the signal propagate through without error. The probability of error is preset as a parameter in the EA/Simulator Configuration file, whilst the type of error is random with equal probability for each of the three possible faults.

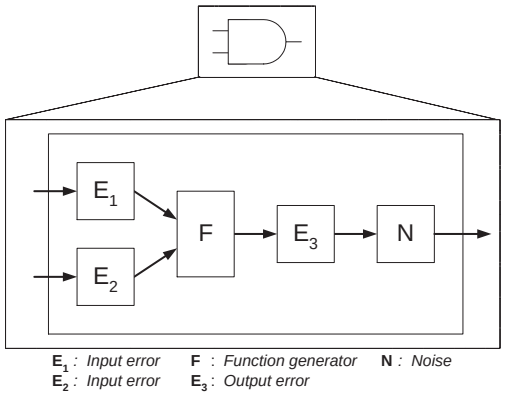

Figure 5. Generic 2-input Gate

F is the model approximation of the real behaviour of the corresponding gate in 5V CMOS. The output noise N, is noise that is superimposed on the 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. The simulator currently supports feed-forward networks and realization details such as routing and layout are ignored.

Parameter name	Range	Description
GATES	[1,∞]	Defines maximum number of gates to be used per individual
GENERATIONS	[1,∞]	Defines maximum number of generations to be calculated
INDIVIDUALS	[1,∞]	Number of individuals in each generation
REPORT_GAP	[1,∞]	Number of generations between each report
MUTATION_RATE	[0,100]	Percent possibility of a gate mutating
FITNESS_LIMIT	[0,∞]	End simulation when a certain fitness is reached.
ALGORITHM	[0,1]	Choose genetic algorithm to use
SEED	[0,∞]	Random seed to use
NOISE	[0,1]	Percent of noise in the circuit
FAILRATE	[0,100]	Percent possibility of a gate failing
ERROR_RUNS	[0,∞]	Number of runs per individual per generation
INPUTS	[1,∞]	Number of inputs in the circuit
OUTPUTS	[1,∞]	Number of outputs from the circuit
DATA_COUNT	[1,∞]	Number of data entries in the truth table
DATA	Binary	Defines the truth table

Table 2. EA/Simulator Configuration File

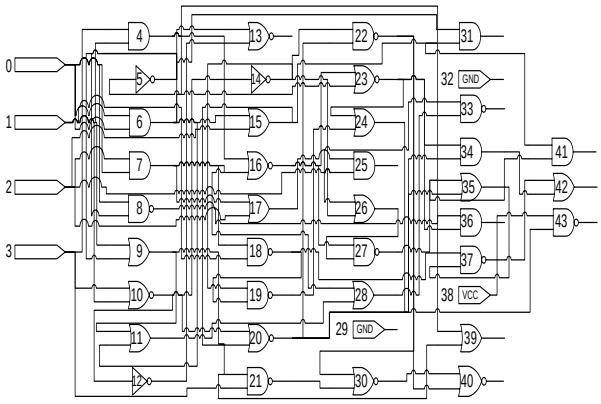

Figure 6. Schematic of Genotype

4 Circuit Evolution

Figure 6 illustrates an example circuit, an individual, which may be represented in our genotype form as shown in figure 7. Connections refer to labels of either the inputs of the circuit (0 to 3) or to 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). Only feed-forward connections are allowed. The genotype uses two-input NOR, NAND,

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

Figure 7. Example Genotype of a 4-inputs, 3-outputs Circuit

OR and AND gates as well as NOT, VCC and GND and, as such, each element has zero, one or two inputs and one output.

The algorithm used for the experiments herein is a (1+λ) evolutionary strategy with neutral genetic drift. That is, a generation consists of the best individual from the former generation and mutations of it. Neutral drift is obtained when in the case of equal fitness amongst the best individuals, a random one is selected, thereby allowing changes in the genetic material while fitness stays at a maximum. The algorithm does not use crossover. The work of Vassilev and Miller [14] illustrates the advantage of this approach when evolving digital circuits.

Mutations are applied at the gate level. If a gate is mutated, one of its inputs is remapped or its type is changed to a random type from the set of gate types. It should be noted that the set of gate types may be expanded to include other new gate types (other than those given in section 3) by tuning the gate behaviour, as described.

The target solution for a given circuit is defined by its truth table and the number of inputs and outputs. Under fitness evaluation, every circuit is subjected to a selection of noise and fault vectors and evaluated using a complete set of possible inputs. All analogue output values of the circuit are clamped to their closest digital value for comparability. The outputs may then be compared to the target solution truth table.

The fitness function is expressed in Equation 1. A circuit C (individual) is tested against the target truth table (T) a number of times (TPI = 10) 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. Finally, these terms are subtracted from the maximum obtainable fitness, Max, to yield a fitness score in the range 0 to the total number of output bits in the target truth table. Thus, the range of the fitness of the 2x2 bit multipliers and adders described in section 5 is 0 to 64 and 0 to 48 respectively.

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

F	Fitness of individual
Max	Maximum obtainable fitness
TPI	Test per individual
$diff()$	Number of incorrect output bits
C_m	Circuit in environment m
T	Target truth table

5 Experiments and Results

The goal of these experiments is to further investigate noise robustness and fault tolerance using a technology specific messy gate model. In addition, experiments were conducted to verify features of the simulator itself. Two application circuits were chosen, a 2x2 bit multiplier and a 2x2 bit adder. The multiplier provides us with a more parallel design and the adder a more serial design. The individuals consisted of 40 gates (thereby providing much space for neutrality, mentioned in section 4), resulting in a genotype size of 44 when evolving 2x2 bit multipliers or adders (4 inputs). The population size was set to 20 ($\lambda = 19$) and the mutation rate was 15%.

Evolution was provided with the following gate type set: NAND, NOR, OR, AND and NOT, as well as the possibility to connect to VCC and GND (logic 1 and 0). During fitness evaluation, noise and gate faults are applied (and randomly generated) at each timestep in the eventdriven simulator. Each experiment is repeated resulting in different winning individuals which have been evaluated under different noise and gate fault conditions. Fitness for a given experiment is an average of the best fitness of all the runs.

For experiments conducted in a noisy environment, the noise percentage shown signifies the amount of noise relative to full signal strength that was applied to each gate output in a given circuit. Noise is implemented as a random value within the range of possible signal values obtained from the noise percentage. Noise was increased in steps of 5%, ranging from 5% to 50%.

The gate failure probability is the chance of any given gate failing i.e. being subject to one of the errors explained in section 3. To investigate gate failure, evolution was performed in environments where the failure probability rate was gradually increased in steps of 5%, from 5% to 15%, in

the case of the multiplier and in steps of 2.5% from 2.5% to 7.5% in the case of the adder.

An important issue in gate failure experiments is the existence of single points prone to critical failure. This problem has been considered in the work presented in [8, 9, 5] although not addressed in the experiments. It may be assumed that it is impossible for evolution to find solutions around failures occurring in output gates, since such a failure would generate errors directly influencing fitness evaluation seemingly regardless of the design of the circuit (within the current simulator setup). Therefore it was decided that it would be more correct to apply gate failures to the remainder of the circuit where evolution can evolve solutions around these failures. As such, valuable simulation time is not wasted trying to evolve solutions for circuits with output gate failures. A more realistic picture of the efficiency of evolution is achieved where this data focusses on the parts of the circuit that evolution can improve. As such, it should be noted that gate failure percentages given in this section are percentages based on non-output gates, termed unprotected gates in the remainder of the section.

5.1 Verification of Evolved Circuits

The experiments reported in this article are all conducted in a simulator environment. It is of course desirable to conduct such experiments in a "real environment" i.e. on a given technology. However, controlling noise and gate failure on such an environment is not easy to achieve. For instance, on todays regular CMOS technology, one would be forced to generate fresh silicon with defects for every evaluation. Reconfigurable solutions would not necessarily be more realistic than simulations. However, new technologies could allow such tuning of defects and fail rates. Herein, however, it was decided to verify the workings of the simulator and the evolved circuits by running a number of evolved circuits on a SPICE simulator.

The 2x2 bit multiplier and the 2x2 bit adder were evolved under perfect conditions i.e. no noise or gate failures, and the evolved circuits were then simulated in SPICE. The current verification has been conducted as a digital verification and the circuits were found to function correctly i.e. their truth tables were as expected.

5.2 Robustness to Noise

The noise robustness of the evolved circuits was investigated by evolving 2x2 bit multipliers and 2x2 bit adders with increasing amounts of noise. In all cases up to 50% noise for the multiplier and 45% noise for the adder, a fully functioning circuit was evolved. The relationship between the noise and the number of generations required to produce the fully functional circuit is shown in figures 8 and 9 respectively. The results show no trend towards increased difficulty in evolvability for increased noise. Rather, a more fluctuating number of generations may be seen as noise increases.

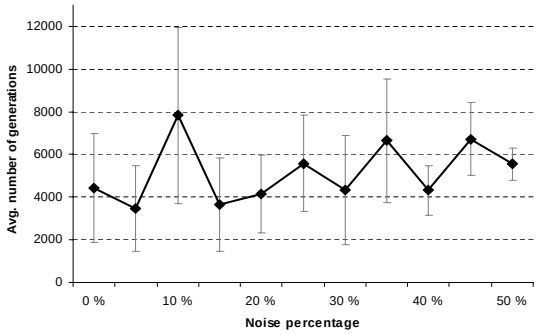

Figure 8. Multiplier: Increasing Noise

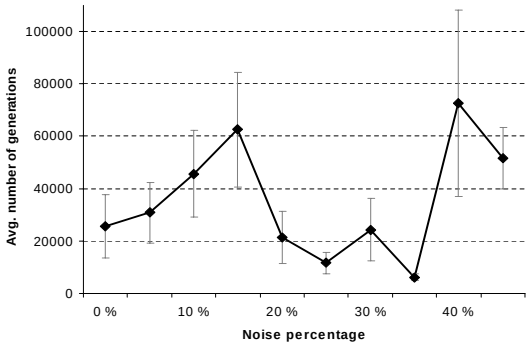

Figure 9. Adder: Increasing Noise

These results indicate that even as much as 50% noise does not create a particularly difficult problem for evolution to solve. Two reasons for this may be suggested. The first is based on the modelled behaviour of a given gate. Considering the NOR gate, as shown in figure 4, it may be seen that the digital gates pull signals towards full or zero signal strength, thereby reducing the propagation of small fluctuations. This effect dampens the influence of noise in the circuit. This may be said to be a typical condition for digital design i.e. the tolerance of a certain amount of noise. However, this does not account for the effects of large noise percentages. The second reason is that evolution can in fact exploit this "messiness" i.e. noise. Certain noise fluctuations may be able to force evolution away from local peaks in the fitness landscape, thus helping the evolution process to reach full fitness. This may explain the dip in the number of generations at 20 to 35%. That is, up to a given noise level, sufficient noise may aid the evolution process.

As may be observed in figure 9, the evolution of adders appears to be about ten times as hard as the evolution of multipliers. There might be several reasons for this, but the use of XOR functionality in adders is perhaps the most crucial. Conventional adder designs require XOR gates or equivalent constructs and since XOR gates are not a part of the set of gates made available to evolution, equivalent constructs must be generated by evolution. On the other hand, the gates required for a more traditional approach to a multiplier design are available to evolution.

An important fact of the evolutionary system is its probabilistic nature. Since noise is applied as a random number within a specified range, it is possible that an individual might encounter a low noise environment on average, even though a circuit is tested a number of times. Perhaps then, the final circuit of one evolutionary run might have gained its perfect fitness through a strike of luck rather then a robust circuit design. To investigate the impact of the probabilistic nature of the evolutionary system all evolved circuits of each noise step were subject to extensive testing after evolution was completed.

Each evolved circuit (individual) was tested 1000 times with varying noise environments (with fluctuations within the noise ranges specified for that circuit). For each experiment set i.e. 1000 tests, the average of the 1000 fitness results was calculated. This process was repeated for each of the winning individuals of the original experiments, giving the average fitness values.

The resulting performance, for multipliers, is shown in figure 10. The diamonds illustrate the average of all average fitness values for the given noise environment. The maximum and minimum values indicate the maximum and minimum of the average fitness values for the given noise environment.

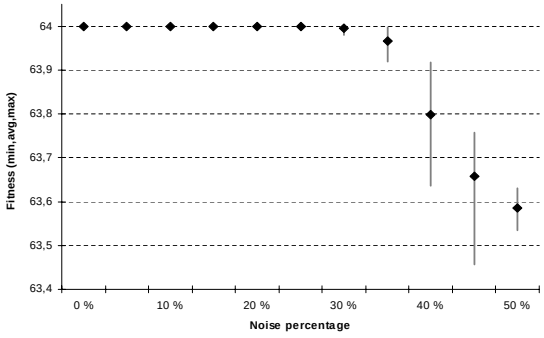

Figure 10. MUL: Verification of Performance in Noisy Environments

The results illustrate that for up to 25% noise, all evolved circuits proved to be immune to noise. In 30 to 35% noise environment, some evolved circuits performed perfectly on average whereas others performed slightly less than perfect but still very acceptable i.e. over 63.9 fitness. In the 40 to 50% noise environment, although our original results in figure 8 illustrate the number of generations towards a fully functional circuit, this thorough testing has shown that none of the best individuals from the original experiments achieved 100% functionality after rigorous testing. However, a minimum of over 63.4 fitness illustrates that the circuits evolved are still very robust.

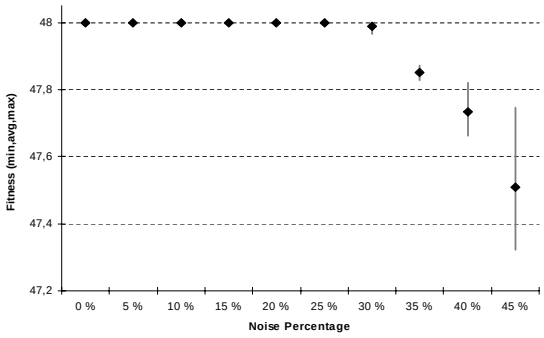

Figure 11. ADD: Verification of Performance in Noisy Environments

The results for the adder, as shown in figure 11 are similar. The adders exhibited immunity noise for up to 25% noise, and near immunity for 30% noise. For 35 to 45% noise, a minimum of over 47.3 was achieved.

5.3 Gate Failure Tolerance

Gate failure experiments were conducted for both the 2x2 bit multiplier and the 2x2 bit adder. The results are shown in figures 12 and 13 respectively.

For the case of the multiplier, it is apparent from figure 12 that when the failure probability increases, it becomes harder to evolve a fully functional circuit. However, with 5 and 10% gate failure it was still relatively easy to evolve solutions. The dip at 15% in fact gives a false impression and the reason for a different marking being used on the graph. These results were an average of only 3 experimental runs rather than 5 shown at 0, 5, and 10%. The remaining 2 runs took over 20,000 generations but didn't reach completion at the time of writing. As such, one can expect a significant increase in the average result for 15% based on all 5 runs, as illustrated by the dotted line. This also indicates a need for further experiments for each gate failure stage. Further experiments at 20 and 25%, not illustrated, indicated a further increase in the number of generations required.

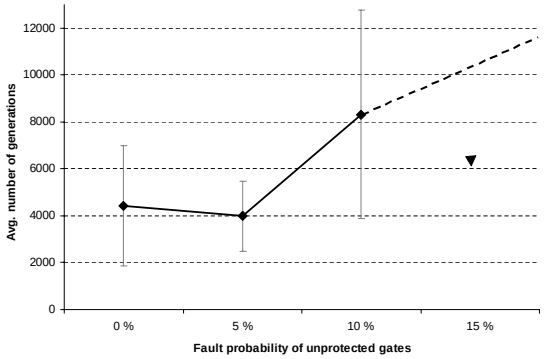

Figure 12. MUL: Increasing the Fault Probability of Unprotected Gates

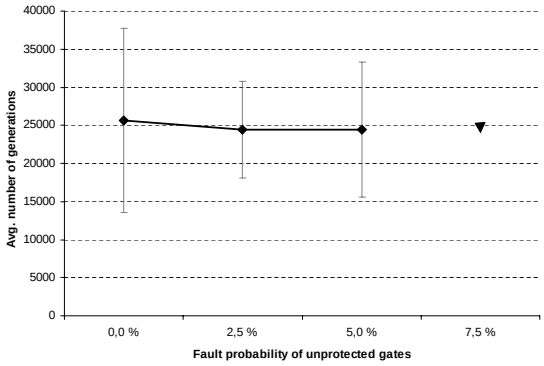

Figure 13. ADD: Increasing the Fault Probability of Unprotected Gates

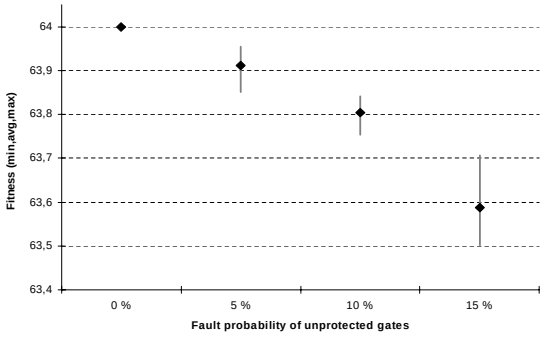

Figure 14. MUL:Verification of Performance when Unprotected Gates Fail

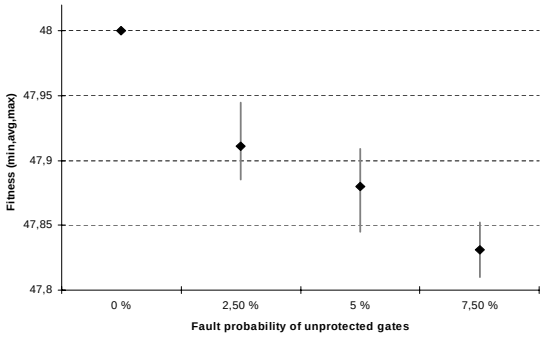

Figure 15. ADD:Verification of Performance when Unprotected Gates Fail

Gate failures are, as for noise, also provided as probabilities to the simulator. However, one can expected a greater impact on evolution since the difference between a gate failing and not failing is more noticeable than the difference between some noise and more noise. Even though the required number of generations shows an increasing trend, it is still a significant achievement for evolution to be able to generate circuits that produce correct output bits at these gate failure rates. There is however some luck involved, since every circuit is only exposed to a limited selection of gate failure test sets i.e. five test sets for each circuit.

For the case of the adder, results are only available for up to 7,5% gate failures, as shown in figure 13. Up to this level no significant increase in the number of generations is seen. However, this is in keeping with the multiplier results and, as such, an increase in generations for increasing gate failures may be expected. One surprising result, however, is that the degree of difficulty of evolving adders with gate failures relative to the difficulty of evolving multipliers with gate failures is only about four times as opposed to ten times highlighted by the noise results.

The verification experiments on gate failure are shown in figures 14 and 15 respectively . As might be expected, the evolved circuits do not appear to be immune to faults. However, one might argue that they show a very strong tolerance to gate failure. For instance, the vertical line at 5% fault probability in figure 14 tells us that one evolved multiplier circuit is able to maintain an average fitness around 63.95 and on average all circuits yield a fitness of about 63.9. The performance gradually decreases as gate failure increases but in a fairly graceful manner. In fact, in the case of 15% gate failure, a minimum average fitness of 63.5 is still maintained. This means that evolution achieves a circuit where we can expect less than one bit error out of 64 when 15% of the unprotected gates fail. A similar graceful decline may be seen in the adders verification results presented in figure 15.

5.4 Combining Faulty and Noisy Environments

Combining faults and noise in the same environment yields a hard circuit design task. Experiments were conducted where both gate faults and noise were gradually increased so as to investigate the manner in which the combination of the two influence evolution i.e. the required number of generations. The number of generations used to evolve the 2x2 bit multiplier are plotted in figure 16 for varying noise and gate failure conditions. Although the results given only provide gate failures up to 2.5, 5 and 15% it may be seen that the combination of noise and gate failures relative to gate failures alone (see figure 12) is an easier task for evolution. Again this confirms our assumption that noise helps evolution to achieve robustness in the light of gate failures.

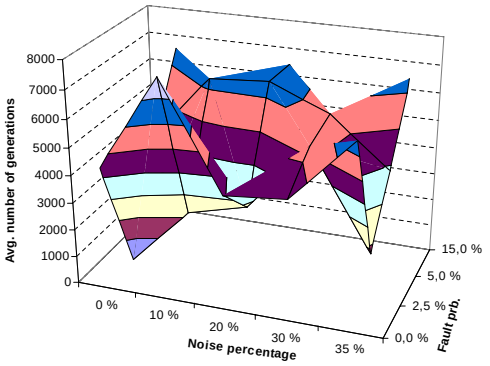

Figure 16. Evolving 2x2 MUL in Combinations of Noisy and Faulty Environments

6 Conclusion

Many aspects of evolution in environments with gate failures and noise level increases have been investigated. The results show that noise does not present a significant issue to evolution. In fact, it seems that evolution makes a virtue of the applied noise. The circuits evolved seem to be immune to significant noise levels (25%). In the case of environments where gates fail, the random failures seem to be more of an issue to evolution than in the case of applied noise. This was expected since gate failures are significantly more critical than applied noise. However, especially in the case of the multiplier, evolution was capable of generating circuits even though the chance of gates failing was severe i.e. 15%.

When the evolved circuits under noisy environments were subject to extensive testing, they proved to be very resilient to faults. However, none of the evolved circuits were immune to gate failures. Interestingly, the combination of noise with gate failures seem to simplify the problem of evolving solutions to gate failures, at least in the case of the multiplier.

Error and noise values are recomputed at each timestep of the simulation and evaluation of a circuit is conducted only after a circuit has been stable for two timesteps. As such, the results presented for gate failures may be quite optimistic since many faults may disappear before the circuit output is compared to the target truth table.

7 Future Work

Scalability is off course an important issue that should be investigated. The evolved robust circuits investigated herein could act as optimised building blocks in a larger system designed by conventional or other means. However, scalability may be better achieved by a scalable methodology. Therefore, we need to investigate if the messy gate concept is in fact scalable to larger designs i.e. can large robust designs be evolved? We have run experiments on 1x1 and 2x2 bit multipliers and adders and are in the process of running 3x3 bit multipliers.

Verification through analogue spice is also desirable. This would allow us to test circuits evolved under noisy conditions, to check correct functionality.

The single point of failure problem, mentioned in section 5, is an issue which needs further attention. It should be possible to enable evolution to use some form of redundancy at the output so that evolution of a fully functional circuit is possible in the light of output gate failures.

One of our goals is to move towards intrinsic evolution. We are currently investigating different hardware solutions which might offer possibilities in this direction.

An issue that is being addressed related to the gate failure results, discussed in section 6 is at what frequency to apply gate faults during evolution.

8 Acknowledgements

The authors would like to thank Julian F. Miller for his ideas on evolution of messy gates and his valuable inputs on digital circuit evolution.

References

[1] H. Abelson et al. Amorphous computing. Technical report, Massachusetts Institute of Technology, 1999.

[2] D. Bradley, C. Ortega-Sanchez, and A. M. Tyrrell. Embryonics + immunotronics: A bio-inspired approach to fault-tolerance. In J. Lohn, A. Stoica, D. Keymeulen, and

S. Colombano, editors, *Proc. The Second NASA/DoD Workshop on Evolvable Hardware, EH 2000*, pages 215–224. IEEE Computer Society, 2000.

[3] D. Bradley and A. Tyrrell. The architecture for a hardware immune system. In D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, editors, *Proc. The Third NASA/DoD Workshop on Evolvable Hardware, EH 2001*, pages 193–200. IEEE Computer Society, 2001.

[4] P. C. Haddow and P. van Remortel. From here to there: Future robust ehw technologies for large digital designs. In D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, editors, *Proc. The Third NASA/DoD Workshop on Evolvable Hardware, EH 2001*, pages 232–239. IEEE Computer Society, 2001.

[5] M. Hartmann, F. Eskelund, P. Haddow, and J. F. Miller. Evolving fault tolerance on an unreliable technology platform. In *To appear in Proceedings of the 4th Genetic and Evolutionary Computation Conference (GECCO'02)*, 2002.

[6] B. I. Hounsell and T. Arslan. Evolutionary design and adaptation of digital filters within an embedded fault tolerant hardware platform. In D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, editors, *Proc. The Third NASA/DoD Workshop on Evolvable Hardware, EH 2001*, pages 127–135. IEEE Computer Society, 2001.

[7] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward self-repairing and self-replicating hardware : the embryonics approach. In *The 2nd NASA/DoD Workshop on Evolvable Hardware*, pages 205–214, 2000.

[8] J. Miller and M. Hartmann. Evolving messy gates for fault-tolerance: Some preliminary findings. In D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, editors, *Proc. The Third NASA/DoD Workshop on Evolvable Hardware, EH 2001*, pages 116–123. IEEE Computer Society, 2001.

[9] J. Miller and M. Hartmann. Untidy evolution: Evolving messy gates for fault-tolerance. In Y. Liu, K. Tanaka, M. Iwata, T. Higuchi, and M. Yasunaga, editors, *Evolvable Systems: From Biology to Hardware. Fourth Int. Conf., ICES 2001*, volume 2210 of *Lecture Notes in Computer Science*, pages 14–25. Springer, 2001.

[10] A. Stoica, D. Keymeulen, and R. Zebulum. Evolvable hardware solutions for extreme temperature electronics. In D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, editors, *Proc. The Third NASA/DoD Workshop on Evolvable Hardware, EH 2001*, pages 93–97. IEEE Computer Society, 2001.

[11] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *1st International Conference on Evolvable Systems, ICES*, Lecture Notes in Computer Science, pages 390–405. Springer, 1996.

[12] A. Thompson and P. Layzell. Evolution of robustness in an electronics design. In J. F. Miller, A. Thompson, P. Thomson, and T. C. Fogarty, editors, *Evolvable Systems: From Biology to Hardware. Third Int. Conf., ICES 2000*, volume 1801 of *Lecture Notes in Computer Science*, pages 218–228. Springer, 2000.

[13] A. M. Tyrrell, G. Hollingworth, and S. L. Smith. Evolutionary strategies and intrinsic fault tolerance. In D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, editors, *Proc. The Third NASA/DoD Workshop on Evolvable Hardware, EH 2001*, pages 98–106. IEEE Computer Society, 2001.

[14] V. K. Vassilev and J. F. Miller. The advantages of landscape neutrality in digital circuit evolution. In J. F. Miller, A. Thompson, P. Thomson, and T. C. Fogarty, editors, *Evolvable Systems: From Biology to Hardware. Third Int. Conf., ICES 2000*, volume 1801 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.