Evolving Fault Tolerance on an Unreliable Technology Platform

Morten Hartmann[1] **Frode Eskelund**[1] **Pauline C. Haddow**[1] **Julian F. Miller**[2]

[1] Dept. of Computer and Information Science
The Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
{mortehar,eskelund,pauline}@idi.ntnu.no

[2] School of Computer Science
University of Birmingham
Birmingham, B15, UK
j.miller@cs.bham.ac.uk

Abstract

One of the key areas in which evolvable hardware has been shown to excel is in achieving robust analogue and digital electronics. In this paper this domain is investigated further by manipulation of the digital abstraction. Some of the strict requirements of digital gates are relaxed in order to increase the complexity of the functionality available to evolution in order to evolve fault tolerant designs. Results from extrinsic evolution of a 2-by-2 bit multiplier, based on CMOS technology under various noise and fault conditions, illustrate the suitability of the messy gate methodology used herein for evolution of a fault tolerant design.

1 INTRODUCTION

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.

Each gate and its connections plays a crucial role in the overall behaviour of a digital circuit. 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.

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 [OT99] and Ecole Polytechnique Fédérale de Lausanne (EPFL) [MSST00], 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 [BOST00] where principles of biological immune systems were adopted to attain fault-tolerance. Work on achieving tolerance to temperature changes includes [TL00] and [SKZ01].

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 an underlying fault-detection and repair from within the technology. The work of Haddow and van Remortel [HvR01] 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 [Aea99]. Hounsell and Arslan [HA01] have developed a fault tolerant hardware platform for the automated design of multiplier-less digital filters. Tyrrell et al [THS01] have used evolutionary strategies to achieve redundancy thus providing inherent fault tolerance in the design.

The approach described herein uses the concept of messy gates [MH01a]. The messy gate concept may be said to be a fault tolerant methodology 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 [Tho96]. Here the focus was not so much on fault tolerance but more towards achieving unique solutions to difficult problems and, as such, illustrating the power of evolution. Messy gates are able to operate in analogue voltage levels that are outside the normal digital scope. In addition, the analogue outputs of the messy gates are allowed to propagate through the circuit.

Circuit designs are evolved using messy gates as their components. As in nature, evolution is not prone to designs where every part is required for satisfactory behaviour, but rather to distributed designs where no single point is crucial. Fault tolerance emerges through the abstraction mechanism as its functionality is exploited by evolution.

Earlier work of Miller and Hartmann [MH01a, MH01b] on the messy gates approach may be said to be a proof of concept. 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 has been extended to a technology specific model. This paper presents the new model, simulator and the results of experimentation.

The model for the messy gates and the simulator that was developed are described in section 2. In section 3, the evolutionary algorithm is explained. Section 4 describes the experiments and section 5 gives a discussion of the results. In section 6 we present some ideas for future work and section 7 presents our conclusions.

2 MESSY GATES AND THE SIMULATOR ENVIRONMENT

Introduced in [MH01a] and elaborated in [MH01b], messy gates is an approach which removes some of the digital abstraction and investigates the impact on evolution of circuits using these gates to achieve fault tolerance. While earlier work is based on an abstraction level not close to a specific hardware technology, this paper introduces a model of messy gates which is very close to one technology, specifically Complementary Metal Oxide Semiconductor (CMOS). The model is parameterised and can be tuned to different semiconductor technologies.

The simulator is based on observations made during analogue simulation of digital gates in Simulation Program for Integrated Circuits Emphasis (SPICE). Each gate was modelled at the transistor level including various error sources. Figure 1 shows the model of a NOR gate which was provided to the SPICE simulator. As shown, the model incorporates three noise generators ($FGEN1$ through $FGEN3$), several capacitances ($C1$ through $C3$), current leaks ($R1$ and $R2$), and output load ($LOAD$). NAND, NOR, NOT, MUX and NMUX (multiplexor with one input inverted) gates were simulated at analogue 5V CMOS transistor level whilst being exposed to different configurations of capacitances, resistances and noise. For the purpose of the experiments herein, only the MUX and NMUX gates were used.

A sigmoid function was used to approximate the fall and rise behaviour of the digital gates. Sigmoid functions for each gate type were individually tuned to approximate behaviour shown in SPICE simulations. For instance, a NOR gate was simulated in SPICE showing a behaviour as shown in Figure 2. Using a sigmoid function this was approximated in our model as shown in Figure 3. 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. The size of this table was limited to 64 KB, in order to allow it to fit into the first level cache of most modern processors. It should be noted that the smoothness displayed in Figure 3 is only the core part of the gate model, and more noisy behaviour like the one in Figure 2 is likely to be displayed as noise is added.

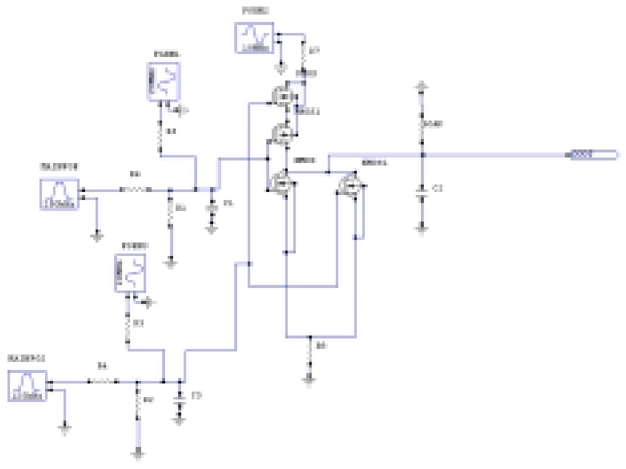

Figure 1: SPICE transistor level layout of NOR gate

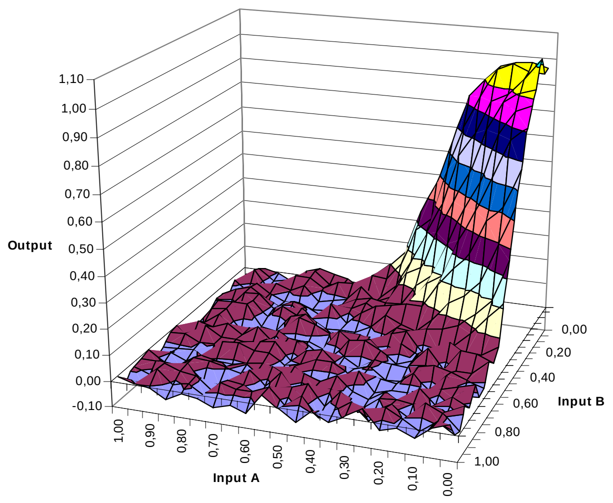

Figure 2: SPICE simulation of NOR gate

The current simulator focuses on internal faults of types stuck-at errors, floating outputs and partly random output,

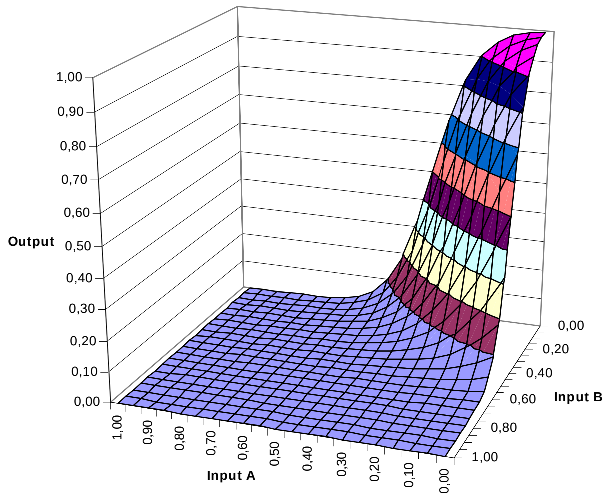

Figure 3: Model approximation of NOR gate

in addition to supporting induced signal noise. Input or output stuck-at errors cover the cases of short-circuit to power or ground and certain cases of inter-signal short-circuits. Floating output errors covers the case 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 simulator uses a real number internal representation within the range 0 to 1, to represent the CMOS voltage range of 0 to 5 volts.

The transistor layout used in SPICE simulations allows analogue signals to propagate through gates and there is no explicit mechanism for pulling the output signal to either a completely high or low state e.g. a push-pull stage. As shown in Figure 3, the sigmoid behaviour will allow propagation of analogue signals and yet be biased towards the digital endpoints of the analogue scale (the real numbers 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.

The resulting gate model used in the simulator is illustrated in Figure 4. E_1, E_2 and E_3 generate one of the supported errors (stuck-at, floating outputs or partly random output) or let the signal propagate through without error. The probability of error is preset as a parameter, whilst the type of error is random with equal probability for each of the four possible faults. F is the sigmoid function approximated to the real behaviour of the corresponding gate in SPICE. Finally, the output noise N 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 errors and noise of this abstracted model is not directly related to the errors of the transistor model, but are present to achieve a behaviour similar to that shown by the transistor model. The simulator currently supports feed-forward networks and realization details such as routing and layout are ignored.

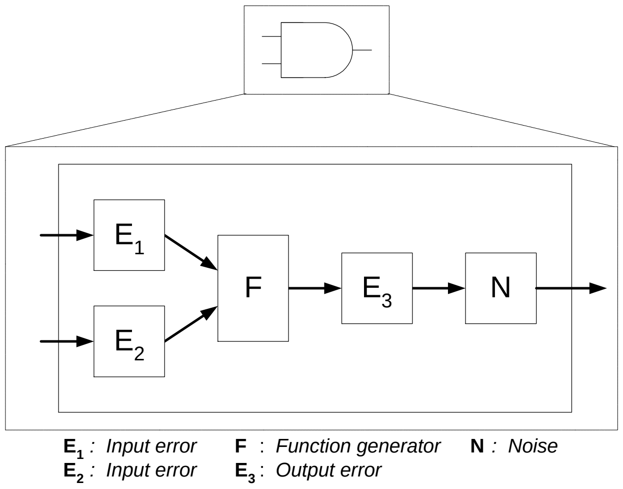

Figure 4: Generic 2-input gate

3 EVOLUTIONARY ALGORITHM

The algorithm used 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, one not from the former generation is selected. The work of Vassilev and Miller [VM00] illustrates the advantage of this approach.

The target solution is defined simply via its truth table and the number of inputs and outputs. Under fitness evaluation, every circuit is subject to the complete set of possible inputs and a selection of noise and fault vectors. The outputs generated by the circuit are compared to the target solution truth table and any analogue output values are clamped to their closest digital value for comparability.

The fitness function used is expressed in Equation 1. A circuit C (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 vectors. Another term $(G_c * G_p)$ penalises the number of gates. 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

nature	label	select	type	input A	input B
input	0				
input	1				
input	2				
input	3				
gate	4	0	MUX	0	3
gate	5	2	MUX	2	0
gate	6	5	NMUX	0	1
gate	7	2	MUX	2	1
gate (o)	8	7	MUX	7	4
gate (o)	9	8	MUX	5	6
gate (o)	10	4	MUX	7	6
gate (o)	11	3	MUX	4	1

Figure 5: Example genotype of a 4 inputs, 4 outputs circuit

output vectors in the target truth table minus 1 e.g. $2^4 - 1$ in the case of the 2-by-2 bit multiplier. Thus, the range of the fitness of the 2-by-2 bit multipliers described in section 4 is 0 to 15.

$$F = Max - \left(\frac{\sum_{n=1}^{TPI} diff(C_m, T)}{TPI} + G_c * G_p \right) \quad (1)$$

F	Fitness of individual
Max	Maximum obtainable fitness
TPI	Test pr. individual
$diff()$	Number of incorrect output vectors
C_m	Circuit in environment m
T	Target truth table
G_c	Number of gates used
G_p	Penalty pr. gate used

An example of the genotype used to represent a circuit is shown in Figure 5. The connections of this specific genotype can be seen in Figure 7. Connections refer to labels of either the inputs of the circuit (0 to 3 in the example) or to the output of one of the gates in the circuit (4 to 11 in the example). The last gates in the genotype representation are considered to be connected to the external outputs of the circuit (8 to 11 in the example). Only feed-forward connections are allowed. The genotype uses MUX and NMUX (multiplexor with one input inverted) and as such, each gate has three inputs (select, input A and input B).

Mutations are done at gate level. If a gate is mutated, one of its inputs is remapped or its type is changed to a random type within the predefined set of gate types. The gate types available to evolution are predefined. The simulator is not limited to the gates described in section 2, but each new gate requires handcrafted tuning of parameters.

4 PERFORMED EXPERIMENTS

The focus of these experiments is to discover the influence of noise and gate failures on evolution of messy circuits. 2-by-2 bit multipliers were evolved using the algorithm described in section 3 and the model and simulator environment defined in section 2. Only gates of type MUX and multiplexors with one input inverted, NMUX, were made available to evolution to maintain comparability with [MH01a, MH01b].

Five sets of experiments were carried out labeled A to E in Table 1. The noise percentage signifies the amount of noise relative to full signal strength that was superimposed at each gate output. Noise is implemented as a random value within this range. Error probability is the chance of any given gate failing i.e. being subject to one of the errors explained in section 2.

All experiments used a gate mutation probability of 15%, population size of 30 individuals and a maximum gate count of 9. Termination of each run occurred when an individual avoided bit errors at the outputs and its size was less than 10 gates.

A noteworthy fact is that each circuit is evaluated ten times under different noise and error conditions. Each of these evaluations gives rise to a fitness value. Fitness for this circuit (individual) is an average of these fitness values. This means that an individual needs to be tolerant to more than one specific configuration of faults and noise and exhibit an overall tolerance to the environmental settings in order for it to survive the selection process.

EXPERIMENT	NOISE	ERROR PROB.
A	10%	0%
B	30%	0%
C	0%	10%
D	0%	30%
E	35%	10%

Table 1: The experimental set of noise and error probabilities

5 RESULTS AND EVALUATIONS

The evolutionary system successfully evolved circuits able to perform a 2-by-2 bit multiplication in all the specified environments. Figure 6 illustrates how the average of the most fit individual for experiments B and E grow towards 100% fitness (15). As expected, the rougher environment in experiment E made it harder to obtain better fitness when compared to experiment B.

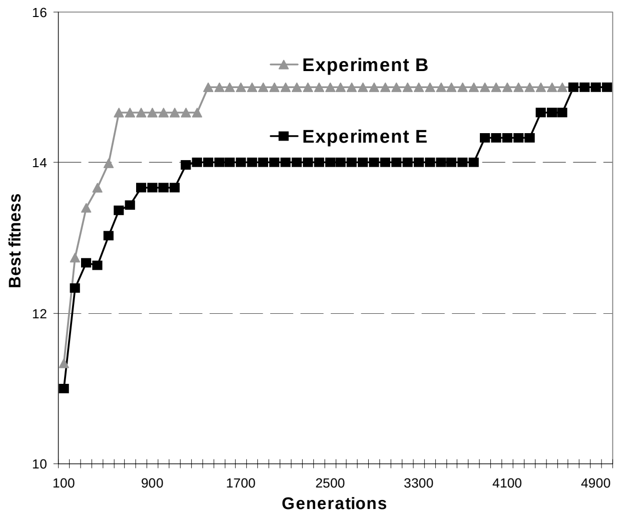

Figure 6: Best fitness individuals averaged over several runs

An example of an evolved multiplier from experiment A is illustrated in Figure 7. The genotype of this specific multiplier is the one shown earlier in Figure 5.

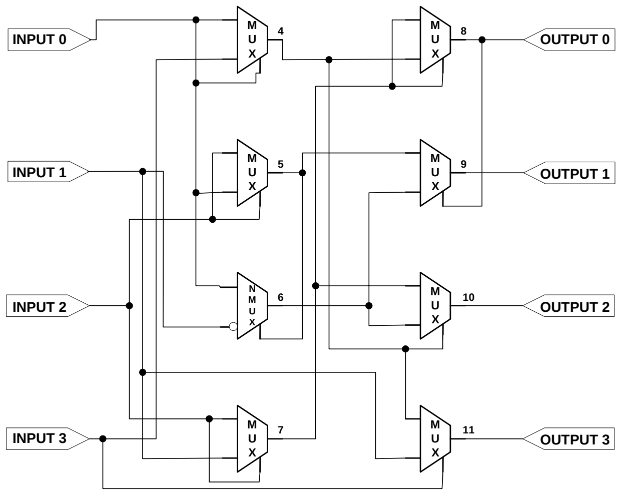

Figure 7: Example of evolved multiplier

As seen in Figure 6 and 8 the number of generations required before a correct individual was evolved increased as noise and error probability got more severe. Experiments A, B and C took on average below 2000 generations. Experiment E required slightly more computational labour, around 3000 generations. This may be said to be due to the combination of severe noise and substantial error probability. Experiment D clearly separates from the other experiments. The gate error probability presented a difficult design task. However, the fact that evolution was able to create such a fault tolerant architecture is quite an achievement. The computational time required is not really that severe, on average the performance is about 100 generations per second.

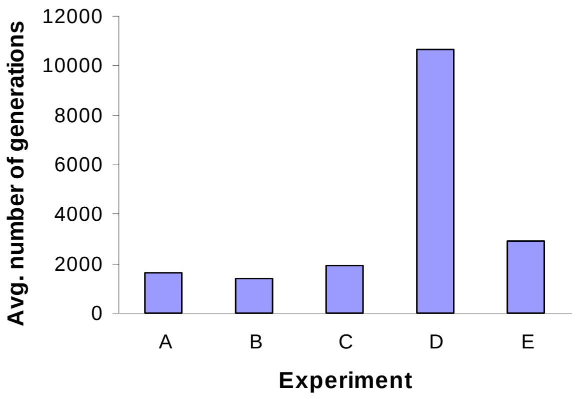

Figure 8: Generations used before termination

In the experiments performed here evolution was not given much room to play with in terms of gates. The maximum number of gates was set to nine. Due to this maximum only a slight variation in the number of gates was seen, as shown in Figure 9. Also, during the experiments, G_p in equation 1 was so low (0.001) that the output correctness always had highest priority. In experiment D the average number of gates used is below the general average. The reason for this is that evolution avoids extra gates, as each gate increases the chance of gate failure. In experiment E, evolution may be assumed to perform similarly where the rough environment in terms of both noise and gate errors forces a bias towards small circuits.

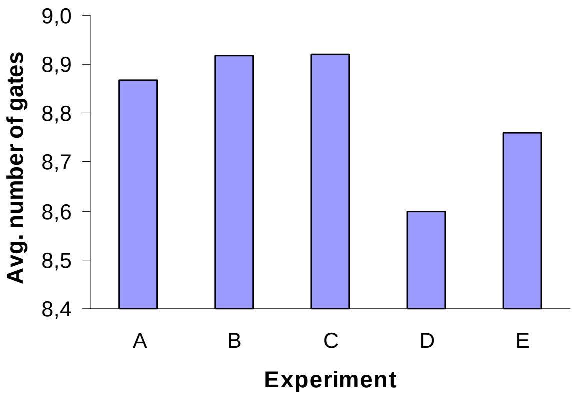

Figure 9: Gates used in final solutions

6 FUTURE WORK

Several extensions of the work described are planned, primarily exploiting the speed of the simulator in order to run extensive numbers of experiments. Such experiments would include allowing evolution to use more gates to increase the genetic variation, evolving with more common type of logic gates like NAND and NOR, evolving circuits with different functionality and exploring the suitability of different evolutionary algorithms. Extensive investigation of the evolved circuits true fault tolerance compared to other approaches would also be advantageous. In addition, verification of the circuits on both digital and analogue levels is important.

A feature of the developed simulator is that its states and signals are fully observable. Combining this with the increased freedom of the semi digital environment may yield results that exploit complex intrinsic silicon functionality not available in a traditional evolutionary digital design environment. Such exploitation of intrinsic features have shown stunning results e.g. in [Tho95, Tho96], but the features of such circuits have been hard or impossible to observe and understand [Lay98]. A goal of our project is to move into hardware implementation with new knowledge on how to exploit the intrinsic properties of the underlying technology.

7 CONCLUSIONS

In this work a simulator has been developed that takes the messy gates approach much closer to real electronic hardware. In this simulated environment evolution is able to perform circuit design tasks that present a serious challenge. Within certain limits, noise influence and fault probability does not even seem to affect evolution. In fact, external influences that human engineers view as destructive and problematic may be exploited by evolution e.g. increasing the noise may in fact decrease the number of generations needed to evolve a complete circuit. When faults and noise influence are increased to severe levels, evolution seems to find it harder to find solutions. Still, it does in fact manage to do so and very quickly when compared to a human designer. Just imagine the design task of creating a 2-by-2 bit multiplier using eight or nine multiplexors in an environment where almost one third of the gates fail on average.

References

[Aea99] H. Abelson et al. Amorphous Computing. Technical report, Massachusetts Institute of Technology, 1999.

[BOST00] Daryl Bradley, Cesar Ortega-Sanchez, and Andy 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*, pp 215–224. IEEE Computer Society, 2000.

[HA01] Ben I. Hounsell and Tughrul 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*, pp 127–135. IEEE Computer Society, 2001.

[HvR01] Pauline C. Haddow and Piet 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*, pp 232–239. IEEE Computer Society, 2001.

[Lay98] P. Layzell. A New Research Tool for Intrinsic Hardware Evolution. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pp 47–56. Springer, 1998.

[MH01a] 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*, pp 116–123. IEEE Computer Society, 2001.

[MH01b] J. Miller and M. Hartmann. Untidy Evolution: Evolving Messy Gates for Fault-Tolerance. In Y. Liu et al, editor, *Evolvable Systems: From Biology to Hardware. Fourth Int. Conf., ICES 2001*, volume 2210 of *Lecture Notes in Computer Science*, pp 14–25. Springer, 2001.

[MSST00] Daniel Mange, Moshe Sipper, André Stauffer, and Gianluca Tempesti. Toward Self-repairing and self-replicating hardware : the embryonics approach. In *The 2nd NASA/DoD Workshop on Evolvable Hardware*, pp 205–214, 2000.

[OT99] Cesar Ortega and Andy Tyrrell. Reliability Analysis in Self-Repairing Embryonic Systems. In A. Stoica, D. Keymeulen, and J. Lohn, editors, *Proc. The First NASA/DoD Workshop* *on Evolvable Hardware, EH 1999*, pp 120–128. IEEE Computer Society, 1999.

[SKZ01] 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*, pp 93–97. IEEE Computer Society, 2001.

[Tho95] A. Thompson. Evolving Electronic Robot Controllers that Exploit Hardware Resources. In *The 3rd European Conference on Artificial life (ECAL95)*, 1995.

[Tho96] A. Thompson. An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. In *1st International Conference on Evolvable Systems, ICES*, Lecture Notes in Computer Science, pp 390–405. Springer, 1996.

[THS01] 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*, pp 98–106. IEEE Computer Society, 2001.

[TL00] Adrian Thompson and Paul Layzell. Evolution of Robustness in an Electronics Design. In Julian F. Miller, Adrian Thompson, Peter Thomson, and Terence C. Fogarty, editors, *Evolvable Systems: From Biology to Hardware. Third Int. Conf., ICES 2000*, volume 1801 of *Lecture Notes in Computer Science*, pp 218–228. Springer, 2000.

[VM00] V. K. Vassilev and J. F. Miller. The advantages of landscape neutrality in digital circuit evolution. In Julian F. Miller, Adrian Thompson, Peter Thomson, and Terence C. Fogarty, editors, *Evolvable Systems: From Biology to Hardware. Third Int. Conf., ICES 2000*, volume 1801 of *Lecture Notes in Computer Science*, pp 252–263. Springer, 2000.