Addressing the Metric Challenge: Evolved versus Traditional Fault Tolerant Circuits

Pauline C Haddow, Morten Hartmann and Asbjoern Djupdal
Complex, Reconfigurable, Adaptive and Bio-inspired Hardware (CRAB)Lab
Dept. of Computer and Information Science
Norwegian University of Science and Technology
(pauline,mortehar,djupdal)@idi.ntnu.no

Abstract

The field of Evolvable Hardware, applying artificial evolution to the design of digital and analogue hardware is around ten years old. However, the field is far from reaching main stream electronics, although some few examples exist. One cause may be that the problems that are addressed in the field are, in general but not always, relatively simple designs which may be regarded as "toy problems", this work being no exception.

Interest in the possibilities inherent in evolved designs is growing, as may be seen from the inclusion of evolvable hardware as a topic in a number of more traditional electronics conferences. However, how good are the designs that are evolved? How can they be compared to their traditional counterparts? Suitable metrics are needed which enable comparison between these two fields of design and that can provide an accurate and fair evaluation of the given design technique. In this work the issue of fault tolerance is addressed together with the design metric reliability.

1 Introduction

The need for fault tolerance is an important issue of modern electronic design. High density chips increase the possibility of failing components and the complexity of design increases the probability of human errors. Space exploration in unknown and dynamic environments places a greater pressure on fault tolerance in terms of the faults themselves and the cost of repair. The need for fault tolerant designs is stated amongst the long term grand challenges of the International Technology Roadmap for Semiconductors, ITRS (2005) [5].

The introduction of reconfigurable technology has, in some ways, reduced the need for fault tolerance as the reconfigurable nature of these devices enables runtime correction, assuming a fault detection mechanism is available. However, when one considers space applications, run-time correction is not so easy. Reconfiguration requires uploading the new configuration when access windows permit access to the device. For small satellites, a 10 minute access window may be available 2/3 times a day with transfer rates of perhaps only 10 to 19kbits/sec [16]. With, for example, Xilinx's virtex II's 10Mbit configuration stream, even with the possibility of partial reconfiguration and techniques such as data compression and splitting and fusing of configuration data, fault tolerance still needs to be addressed to reduce the need to reconfigure.

The field of evolvable hardware, EHW, where artificial evolution is applied to the design of electronic circuits, is a promising field in the area of fault tolerant design for dynamic environments. The field of fault-tolerance in EHW was pioneered by Thompson, who in 1995 evolved fault-tolerant electronic control systems [14]. More recent work on evolved fault-tolerance includes that of The Intelligent Systems Research Group at The University of York [15, 3], which in cooperation with L'Ecole Polytechnique Fédérale de Lausanne (EPFL) [10] also investigates other bio-inspired methods for achieving fault-tolerance [6]. Stefatos and Arslan [12]proposed a two-layered fault-tolerant VLSI architecture where the second layer provides for detection and correction of the first layer. At NASA, field-programmable transistor arrays (FPTA) are evolved to be fault-tolerant [7] and fault-recovering [17]. Genetic representations are also investigated with regards to fault recovery on FPGAs [9]. Branke has investigated the workings of evolutionary systems in dynamic environments [1, 2]. Zhang et al, are investigating competitive and concensus-based evolution as an approach to handling faults [18]. A large variety of static circuit topologies were extrinsically evolved to be fault-tolerant and/or robust to noise by Hartmann and Haddow [4].

However, how good are the results being produced? One major problem is that not only the area of fault tolerance but the field of evolvable hardware itself is still in its infancy, creating relatively simple designs and evaluating these designs using internal metrics e.g. number of generations, which have no meaning at all in terms of traditional designs. The problems associated with the evolution of less simple designs are well known in the field and there is much work addressing these issues. In this paper the focus is to address the challenge of finding ways to evaluate evolved designs such that they may be fairly compared to traditional designs evaluated by traditional metrics, and vice versa. To avoid the challenge of evolving larger designs, these initial investigations are based on evaluations of a relatively simple design — a multiplier, and the limitations that such a decision brings to the results are presented.

In this paper, the fault tolerance metric reliability is addressed, using two variations of this metric. The first R_{trad}, as used in traditional design evaluation, is a measure of the probability that a system will not fail under specified conditions. This reliability measure is, of course, a measure of the probability that a circuit is 100% functional and says nothing about how dis-functional the circuit is when it is not 100% functional. The second reliability measure applied R_{ehw}, most often applied as the fitness measure in design by evolution — see section 2, represents how correct the solution is, on average, over a spectrum of faults. All experiments are conducted on simulations of a simple electronic circuit — a 2 bit multiplier, and the traditional designs evaluated are based on a non-redundant multiplier and a multiplier design with triple modular redundancy (TMR).

The paper provides an introduction to Evolvable hardware and, in particular, evolving fault tolerant designs in section 2. Section 3 explains how circuits designed using traditional techniques are incorporated in our experiments. The representation used in evolving circuits is presented in section 4. Details about how circuits were evolved and tested, and how faults are applied are given in section 5. The experiments conducted, their results and analysis of these results are given in section 6. Finally, section 7 draws some conclusions from this work.

2 Evolvable Hardware

The principles of artificial evolution are based on Darwin's theory of evolution by natural selection. His theory was first adapted to the field of artificial evolution in 1975. The main features of natural evolution consisting of reproduction by cloning, mutation and crossover through the exchange of and alteration of genetic material are included in today's evolutionary algorithms. However, an artificial selection mechanism was introduced, which unlike that in nature, steers artificial evolution towards a given goal i.e. a solution to a given problem.

The application of evolutionary techniques to hardware design is termed evolvable hardware (EHW) [13], the main goal being to replace traditional design methods with evolutionary techniques for given hardware applications. These applications are either not achievable using traditional methods or would benefit from an evolutionary approach. A number of algorithms have been developed for evolutionary design and many have been applied to evolvable hardware. To highlight some of the main concepts behind the concept of an evolutionary algorithm, a genetic algorithm is described in section 2.1. It should be noted, however, that the evolutionary algorithm applied in this work is termed Cartesian Genetic Programming [11].

2.1 Genetic Algorithms

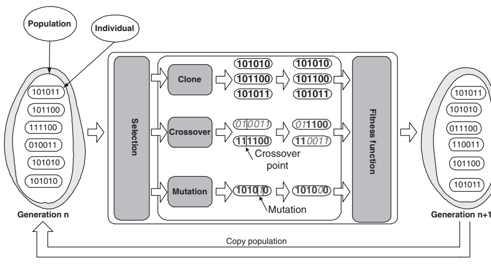

Figure 1. A Genetic Algorithm

The evolutionary process is a dynamic process where at a given point in time we have what is termed in biology a generation: a population of individuals for the given species. Biologically, a generation change is not an event but a constantly unfolding process. However, in artificial evolution, we introduce new generations through the application of genetic operators to selected individuals within the current population to form a new population of individuals i.e. a new generation. Unlike biological evolution, in artificial evolution we have a concrete goal, that is to create a correct solution i.e. a functionally correct circuit which solves a given problem. The goal is specified in terms of the fitness measure. This measure may define the functionality of the circuit i.e. how correct a given circuit solution is. However, it may also take into account other features such as area usage and power consumption and weight these different factors. It is the purpose of the selection mechanism to steer evolution towards the goal described by the fitness measure.

Figure 1 illustrates the process of a genetic algorithm. The set of individuals — circuit solutions, make up the population in the current generation. Individuals are selected and genetic operators such as cloning (copying), crossover (swapping information between individuals) or mutation (inverting a bit) may be applied before fitness is calculated and the individuals are placed in the new population. When the new population is full then the next generation can begin. The process continues until an individual in the population achieves 100% fitness or the simulator is stopped after a certain number of generations.

2.2 Evolving Fault-tolerant Designs

Biological organisms are, in comparison to electronic systems, extremely tolerant to sudden variations and changes such as failing elements. While the failure of a single transistor in a standard processor often has critical consequences, biological systems are constantly subject to similar or often much more severe failures, but usually continue to operate unaffected. Biologically inspired fault-tolerance tries to transfer such properties of biological systems to engineered systems by identifying and using key features in biology that contribute to those properties.

In the process of evolving a design, the mutation operator of the evolutionary algorithm, which applies change to individuals based on a mutation probability, may be said to act as noise in the evolution process. In some cases, this may give rise to a certain amount of implicit fault tolerance in the evolved design. As such, even with 0% faults applied, a certain amount of fault tolerance may be present in the evolved design. In this work, evolved designs are both implicitly and explicitly designed for fault tolerance.

Explicit fault tolerance is herein achieved by testing individuals during evolution with various fault scenarios and evaluating how well the individual functions under these scenarios. These results, R_{ehw}, are used to provide a fitness measure for the individual. The goal is that this fitness measure, combined with the selection method applied, are suitably defined so as to drive evolution towards finding good fault tolerant solutions.

R_{ehw} provides evolution with information, not only on the correct solutions but on how good the non correct solutions are. As such, the R_{ehw} measure provides more fine grained information to the evolutionary process about the quality of different solutions than that which would be achievable with R_{trad}.

3 Introducing Traditional Redundance

The most common form of redundancy in traditional design is that of Triple Modular Redundancy(TMR). Logic is encased (virtually) in a module and replicated three times and a majority voter is applied to these modules. The voter outputs a vector of bits where each bit is the bit produced by the majority of the modules. TMR ensures correct output as long as faults occur in only one of the modules at a time and none of the faults occur in the majority voter. The voter is thus a critical part of a TMR circuit. However, the voter may be replicated but at the cost of tripled voter logic. Further, at some point in the remainder of the circuit, these output signals will need to be voted to a single output.

To incorporate traditional fault tolerance, TMR was applied to the traditional multiplier circuit applied herein. It was chosen to apply TMR to the multiplier and not to the voter in order to avoid the drastic increase in the number of gates in the TMR circuit and thus avoiding the tripled output problem.

4 Evolved circuit Representation

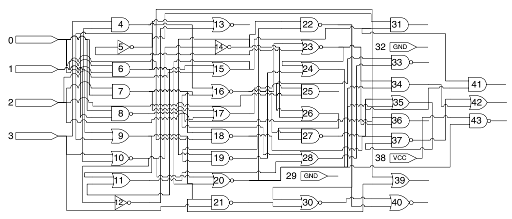

Figure 2. Example Circuit

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 3. Symbolic Netlist of the Example Circuit

Both evolved and traditional circuits were expressed in the simulator using a symbolic netlist representation. Figure 3 provides a symbolic description of the circuit illustrated in figure 2. Each row represents a gate, with its type and from where to connect its two inputs. Connections can only be made to lower numbered gates (or the circuit inputs). This simple scheme assures a pure feed-forward network and thus combinatorial circuits. The last gates in the netlist are connected to the circuit output.

5 Experimental Setup

All experiments were conducted on simulations of 2-bit multiplier designs. In earlier evolved experiments [4] faults were applied as stuck-at-faults at the inputs or at the outputs. As such, a given stuck-at-fault may or may not create a logical fault. In the work herein, faults were applied as an inverting of the output i.e. a worst case scenario, and applied on a fault rate per gate basis.

For the evolved circuits, a population of 20 randomly generated individuals was given to the evolution process. The selection method applied was tournament selection with elitism [8], crossover was applied at a rate of 0.2 and mutation at a rate of 0.05. In these experiments, mutation was applied at the gate level. A mutated gate either has one of its input connections rewired or its type changed. The evolution of each solution i.e. each evolved circuit, was allowed to continue for a maximum of 100,000 generations and the maximum size of an individual (the genome) was 80 gates.

The goal of evolving a fault tolerant circuit solution can be interpreted as the evolution of a circuit that successfully produces the correct mapping between input and output vectors despite faults. The target behavior of the multiplier was specified by a truth table. Each individual of the evolutionary process was evaluated by applying the complete set of input vectors and calculating the hamming distance between the output of the circuit to the target truth-table. The fitness was then the average of the hamming distance in 20 fault scenarios, with faults being applied randomly for each scenario according to the per gate probability of failure for the given experiment. The fitness result was normalised between 0 and 1, 0 being no correct outputs and 1 being all correct outputs.

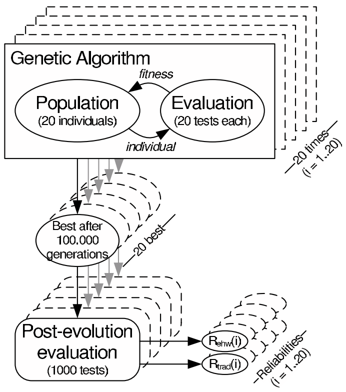

Figure 4. Experiment setup

Figure 4 illustrates the experimental setup. As shown, the evolutionary process ("Genetic Algorithm") was repeated 20 times to achieve 20 evolved circuit solutions and thus 20 best individuals for the given fault rate. Each best solution was then tested with 1000 test scenarios. The results of each of the 1000 tests were used to calculate R_{ehw} and R_{trad} for each of the 20 best individuals. The results plotted on the graphs for a given fault rate represent the best reliability result from the reliability of the 20 best evolved circuits after the post evolution testing. It should be noted that there are two ways in which the 1000 test scenarios are applied. The first (a) is to apply the 1000 tests at the same fault rate as the fault rate applied when the individuals were evolved. The second (b) is to apply a different fault rate during the 1000 tests. The first case is termed, herein, *increasing fault rate* whereas the second is termed *fixed fault rate*. Fault rates investigated in this work range from 0.0 to 0.2.

One important consideration when applying faults as faults per gate is that the bigger the circuit i.e. more gates present, the larger the challenge one faces with respect to reliability. As such, a traditionally designed multiplier using a minimum number of gates was included in the experiments. A 9 gate multiplier was hand-designed using Karnaugh map and manual optimisation of joint logic between separate outputs.

6 Experiments and Results

The goal of these experiments was to understand and compare the effect of using the two metrics : reliability traditional R_{trad} and reliability evolvable R_{ehw}, with respect to digital circuits — in particular a 2-bit multiplier.

6.1 Reliability of Evolved Circuits

In earlier work on evolving reliable circuits [4], R_{ehw} was the reliability metric applied. Faults were applied both at the inputs and output to a gate providing the possibility that a fault would not necessarily result in a gate failure. Fault scenarios applied during evolution were based on the fault rate per gate under investigation. The best individuals from 20 evolutionary runs were then each tested 1000 times using the same fault rate as applied during evolution. R_{ehw} was calculated for each of the 20 results and the average as well as the best and worst results were plotted. Output gates were protected from faults, as such faults could not be remedied by evolution.

These experiments are repeated, herein, but with a worst case fault scenario — inverting the output to cause a gate failure, an applying the "increasing fault rate scenario". The results are displayed in figure 5 as R_{ehw}. As stated, R_{ehw} represents the best result of the 20 individuals. The best result was chosen, rather than average, to illustrate what evolution is capable of rather than what it achieves on average. As shown, evolution achieves circuits with over 70% reliability even with 20% faults in a worst-case fault scenario.

These figures may, of course, be said to be misleading, in terms of a more traditional view of reliability. As such, it was important to evaluate these circuits in terms of R_{trad}. R_{trad} was calculated, similar to R_{ehw}, as the best result for the 20 individuals under the 1000 post evolution fault scenarios — see R_{trad} in figure 5. As can be seen from the results, at a fault rate of 0.04 and greater, evolution does not achieve any circuits that are 100% functional.

The break at 0.04 would indicate that at this point the task of achieving 100% correctness becomes too hard for evolution. However, as indicated by R_{ehw}, evolution manages to retain a reasonable sub-optimal solution, degrading gracefully from around 90% R_{ehw} at 0.04 to around 75% at 0.2.

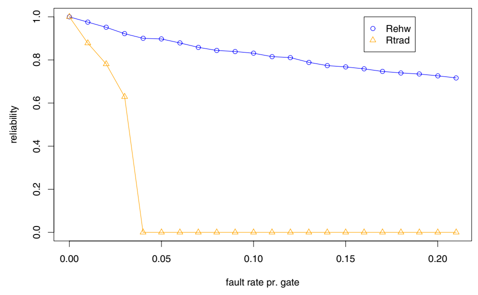

Figure 5. Circuits Evolved with Increasing Fault Rates

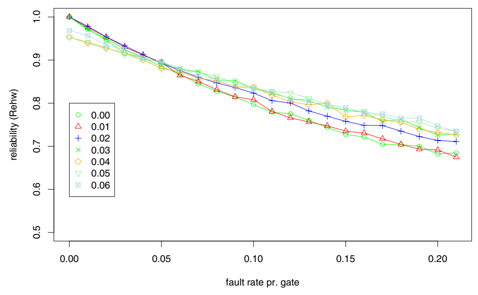

Figure 6. Rehw: Circuits Evolved with Fixed Fault Rates (0 to 0.06)

Looking again at R_{trad} in figure 5 raised the question: could the better results achieved at fault rates from 0 to 0.03 be exploited? How reliable were these circuits at higher fault rates than those fault rates that the circuits were exposed to under evolution? What would happen if a circuit evolved to tolerate 0.01 faults were tested for tolerance to a range of faults? To investigate either side of the breakpoint in the graph, the results from evolving circuits at 0.0 to 0.06 were given a new set of post evolution testing. For each of these fault rates, the 20 best individuals were tested against the whole range of fault rates i.e. exposed to 1000 fault scenarios for each fault rate. This fixed rate fault testing resulted in 7 sets of results for all fault rates. Both R_{ehw} and

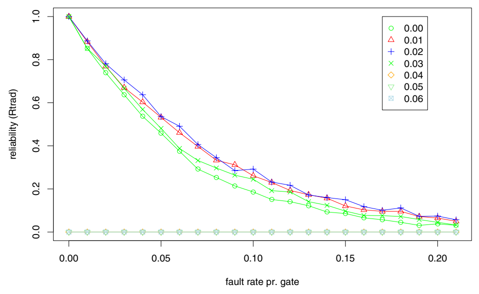

Figure 7. Rtrad: Circuits Evolved with Fixed Fault Rates (0 to 0.06)

R_{trad} were measured, as illustrated in figures 6 and 7 respectively. For the case of R_{ehw}, as shown, circuits evolved with lower fault rates perform well below fault rates of 0.04. On the other hand, circuits evolved with higher fault rates perform well above fault rates of 0.04. It should be noted that the scale on the y-axis has been increased, compared to other graphs so that the results are more visible. There is a substantial improvement in R_{trad} when fixed fault rates of 0.0 to 0.04 are applied. One may assume that fault rates of 0.05 and 0.06 provide too difficult a task for evolution and, even at a fault rate of 0.00, the best evolved circuits for 0.05 and 0.06 are not able to produce any 100% functional circuits from the 1000 fault scenarios. The best results from fixed fault rates for both R_{ehw} and R_{trad} are displayed in figure 8.

Comparing the fixed fault rate solution ("Best of 0.00-0.06 fault rate") with that of increasing fault rates ("Evolved with given fault rate") in figure 9 one can see that there is little difference in R_{ehw}. However, R_{trad} is substantially improved — see figure 10. Even at a fault rate of 0.2 some 100% functional solutions may be found.

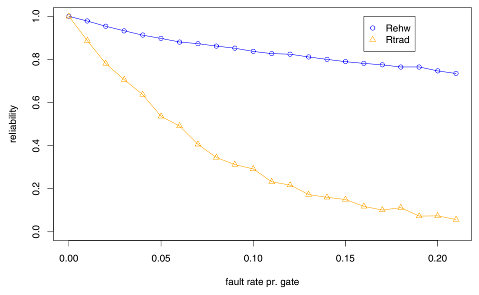

Figure 8. Circuits Evolved with Fixed Fault Rates (0 to 0.06)

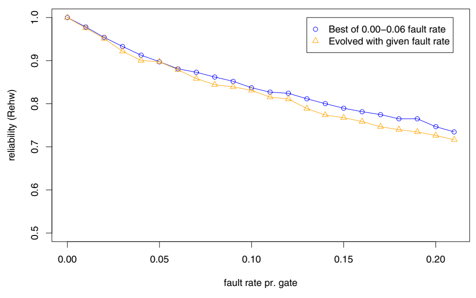

Figure 9. Rehw: Evolved Fixed vs Evolved Increasing

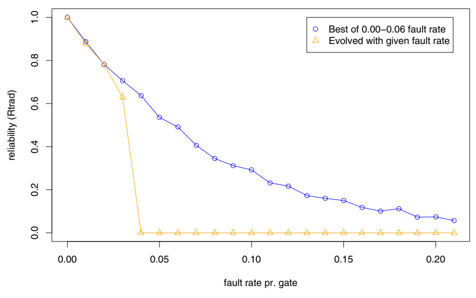

Figure 10. Rtrad: Evolved Fixed vs Evolved Increasing

6.2 Reliability of Traditional Circuits

To calculate R_{ehw} for the traditional circuit, 1000 fault scenarios were applied and their average calculated for each of the fault rates from 0 to 0.2. Figure 11 illustrates the results for both the traditional and traditional with TMR version of the circuit. Unfortunately, but not unexpectedly, the weighting of the 16 vulnerable gates in the voter compared to the small 9 gate multiplier modules is a big disadvantage to TMR as a methodology. Also the sheer size of the circuit makes the average number of occurring faults relatively large due to faults being generated based on gate reliability. The traditional circuits without redundancy are, therefore, more reliable than those with TMR.

Figure 12 illustrates R_{trad} for both the traditional and TMR version of the tradition circuit. Statistical estimates which illustrate the validity of the results received are also plotted. It should be noted that the TMR multiplier results deviate from the statistical estimates. The reason for this is that the statistical estimates do not take into account a number of features inherent in TMR. One such feature is the fact that two faults may occur on a TMR triple gate without

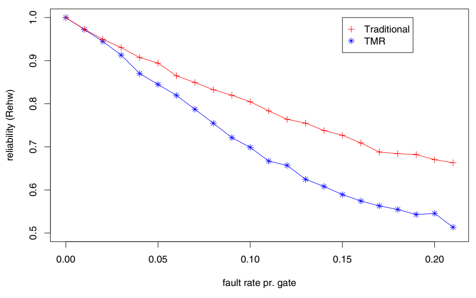

Figure 11. Rehw: Traditional Circuits

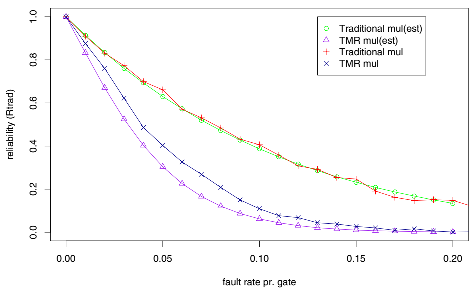

Figure 12. Rtrad: Traditional Circuits

resulting in gate failure. As such, the measured reliability will be higher than the estimated results. It may, also, be noticed that the traditional circuit with TMR performs, as in the R_{ehw} case, poorer than the non redundant version.

Similar to the evolved circuits, it may be seen that R_{ehw} results give a more positive impression of reliability than the R_{trad} results.

6.3 Traditional vs Evolved

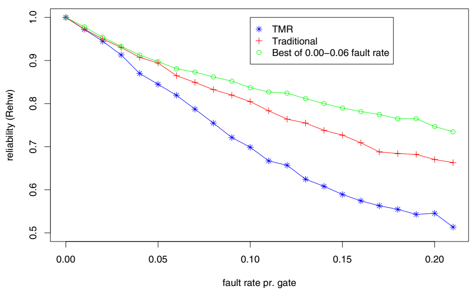

Figure 13. Rehw: Traditional vs Evolved

R_{ehw} for the traditional circuits is somewhat poorer than the evolved results — see figure 13, indicating that the

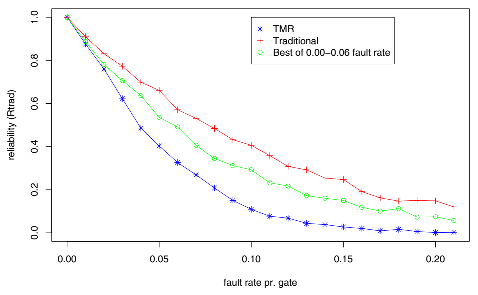

Figure 14. Rtrad: Traditional vs Evolved

evolved solutions perhaps exhibit more graceful degradation. On the other hand, one can see from figure 14 that the evolved circuits are less reliable, with respect to R_{trad}, than the traditional solutions without TMR. As stated, in the evaluation phase the evolving circuits are not being optimised against a specific fault scenario. Instead a sub-optimal solution is being sought with respect to individual fault scenarios which gives rise to a more optimal solution for the set of fault scenarios. Further, the evolved results achieve a better R_{trad} than the traditional circuits with TMR.

6.4 Gate Usage

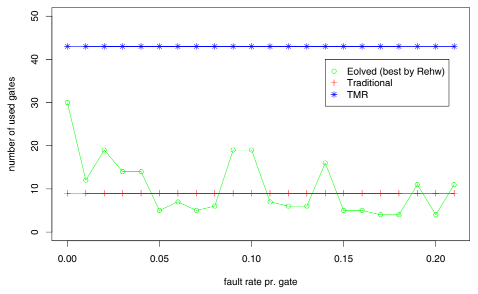

Figure 15. Rehw: Number of Gates versus Fault Rate (per gate)

Figure 15 illustrates gate usage for traditional, traditional with TMR and evolved with fixed fault rate circuits. Each point in the evolved with fixed fault rate experiments represents the best result, with respect to R_{ehw}, of the 7 result sets for each fault rate. Although the results are somewhat unstable in nature, there is a tendency for evolution to reduce the size of the circuits to achieve better R_{ehw} values. Looking at the gate count for R_{trad} —see figure 16, more stability can be seen in the gate count for the evolved circuits. This is due to the fact that, for most of the points in the graph, it is the same circuit of the 20 evolved circuits which achieves the best R_{trad}.

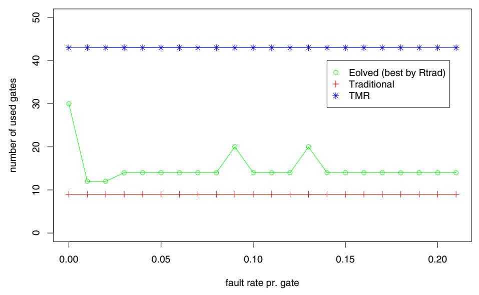

Figure 16. Rtrad: Number of Gates versus Fault Rate (per gate)

7 Conclusion

This paper has addressed the issue of applying reliability metrics to both evolved and traditional circuit solutions and compared the performance of circuits with regards to those metrics in terms of a worst case fault scenario applied on a per gate basis.

When considering the results it is worth noting that the fault scenarios used herein are hard to solve. Redundancy techniques, in general, can cope with single faults. The traditional circuit with TMR, for instance, would handle any single fault not occurring in the voter element. The faults herein are, however, applied as the probability of a gate failing on a per gate basis. All circuits are tested 1000 times in arbitrary fault scenarios. Several of these scenarios will present the circuit with a large number of failing gates, often including the output gates. Several of the generated scenarios may, thus, be fault scenarios that are impossible to solve, regardless of the circuit architecture whether evolved or designed using traditional techniques.

Evolved circuits, herein, are evolved with the goal of improving R_{ehw} in the presence of faults. On the other hand, R_{trad} is a design metric tuned to traditional design methods. As a consequence it is not so surprising that evolved circuits are more reliable in terms of R_{ehw} and traditional circuits are more reliable in terms of R_{trad}.

The circuits used in the experiments conducted may be said to be small toy problems. For larger problems observed results may differ. This is especially true in the case of TMR, as the technique is not suited for problems of the current size. Further investigations would, therefore benefit from scaling up the application. Scaling up solutions is, however, currently one of the greater challenges faced by the evolvable hardware community as a whole.

The field of evolvable hardware is still a field, far from maturity, and applying evolvable techniques is not a straight forward science. As shown, applying fixed fault rates during evolution led to greatly improved reliability, especially in terms of R_{trad}. One might expect that one could use R_{trad} to drive evolution instead of R_{ehw} and thus improve R_{trad} for evolved designs. However, R_{trad} does not provide sufficiently fine grained circuit feedback so as to separate potential solutions and, therefore, limits evolution's search for a solution to a rougher search.

Another topic that needs to be addressed in the future is the suitability of R_{ehw} and R_{trad} for different applications. Whilst R_{trad} has obvious advantages as a metric, R_{ehw} is also beneficial in certain cases such as in circuit evolution. Other cases may include those where partially correct outputs would still be of greater value than a completely incorrect output. This could, for instance, be correctness for certain input vectors in a multiplier or reduced resolution or precision in general. Also, as technology scales, production defects and lifetime faults are likely to become an increasing problem. As such, one can expect R_{trad} to fall as it is going to be harder and harder to tolerate faults 100%. It may, therefore, become important to not only investigate R_{trad} but also R_{ehw} in traditional designs to study circuits degradation in the presence of faults.

Another interesting fact observed was that, in certain cases, evolution was seen to have high tolerance to faults with up to 3 times the number of gates of the traditional non-TMR solution. In these cases a question arises. What type of redundance is evolution using so as not to experience the negative effect of a large number of gates experienced by the TMR solution? Although this is not a general result, these cases are interesting in their own right and worth further investigation to find out whether we can improve traditional redundance techniques based on architectural observations from evolved circuits.

References

[1] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.

[2] J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. *Theory and Application of Evolutionary Computation: Recent Trends*, 31, 2002.

[3] R. O. Canham and A. Tyrrell. Evolved fault tolerance in evolvable hardware. In *Proc. Congress on Evolutionary Computation (CEC'02)*, pages 1267–1272. IEEE, May 2002.

[4] M. Hartmann and P. C. Haddow. Evolution of fault-tolerant and noise-robust digital designs. *IEE Proc. -Comput. Digit. Tech.*, 151(4):287–294, 2004.

[5] International Roadmap Committee. Executive summary. In *International Technology Roadmap for Semiconductors*. 2005. http://public.itrs.net.

[6] A. H. Jackson, R. Canham, and A. M. Tyrrell. Robot fault-tolerance using an embryonic array. In J. Lohn, R. Zebulum, J. Steincamp, D. Keymeulen, A. Stoica, and M. I. Ferguson, editors, *Proc. The 2003 NASA/DoD Conference on Evolvable Hardware*, pages 91–100. IEEE Computer Society, 2003.

[7] D. Keymeulen, R. S. Zebulum, Y. Jin, and A. Stoica. Fault-tolerant evolvable hardware using field-programmable transistor arrays. *IEEE Transactions on Reliability*, 49(3):305–316, 2000.

[8] J. Koza. *Genetic Programming*. The MIT Press, 1993.

[9] J. Lohn, G. Larchev, and R. DeMara. A genetic representation for evolutionary fault recovery in virtex fpgas. In A. M. Tyrell, P. C. Haddow, and J. Torresen, editors, *Evolvable Systems: From Biology to Hardware. Fifth Int. Conf., ICES 2003*, volume 2606 of *Lecture Notes in Computer Science*, pages 47–56. Springer, 2003.

[10] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward self-repairing and self-replicating hardware: the embryonics approach. In J. Lohn, A. Stoica, D. Keymeulen, and S. Colombano, editors, *Proc. The Second NASA/DoD Workshop on Evolvable Hardware, EH 2000*, pages 205–214. IEEE Computer Society, 2000.

[11] J. F. Miller, D. Job, and V. K. Vassilev. Principles in the evolutionary design of digital circuits – part i. *Journal of Genetic Programming and Evolvable Machines*, 1(1):8–35, 2000.

[12] E. F. Stefatos and T. Arslan. An efficient fault-tolerant, vlsi architecture using parallel evolvable hardware technology. In *Proc. 2004 NASA/DoD Conference on Evolvable Hardware*, pages 1–7, 2004.

[13] A. Thompson. Evolving electronic robot controllers that exploit hardware resources. In *The 3rd European Conference on Artificial life (ECAL95)*, 1995.

[14] A. Thompson. Evolving fault tolerant systems. In *Proc. 1st IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, pages 524–529. IEE Conf. Publication No. 414, 1995.

[15] 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.

[16] T. Vladmirova, X. Wu, K. Sidibeh, D. Bernhart, and A.-H. Jallad. Enabling technologies for distributed picosatellite missions in leo. In *Proc. First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, pages 330–337. IEEE, 2006.

[17] R. S. Zebulum, D. Keymeulen, V. Duong, X. Guo, M. I. Ferguson, and A. Stoica. Experimental results in evolutionary fault-recovery for field programmable analog devices. In J. Lohn, R. Zebulum, J. Steincamp, D. Keymeulen, A. Stoica, and M. I. Ferguson, editors, *Proc. The 2003 NASA/DoD Conference on Evolvable Hardware*, pages 182–188. IEEE Computer Society, 2003.

[18] K. Zhang, R. F. DeMara, and C. A. Sharma. Consensus-based evaluation for fault isolation and on-line evolutionary regeneration. In *Proc. International Conference in Evolvable Systems (ICES'05)*, volume 3637, pages 12–24. Springer, 2005.