Evolution of Combinatonial and Sequential On-Line Self-Diagnosing Hardware

Miguel Garvie and Adrian Thompson
Centre for Computational Neuroscience and Robotics,
School of Cognitive and Computing Sciences,
University of Sussex,Brighton BN1 9QH, UK.
{mmg20,adrianth} @cogs.susx.ac.uk

Abstract

The evolution of circuits with on-line built-in self-test is attempted in simulation for a full adder, a two bit multiplier and an edge triggered D-Latch. Results show that evolved designs perform full diagnosis using less or equal number of components than hand-designed equivalents.

1 Introduction

Traditional approaches to BIST use the Test Pattern Generation (TPG) - Design Under Test (DUT) - Test Response Evaluation (TRE) model, with the first and last being implemented using Linear Feedback Shift Registers (LFSR) [4]. Variations exist such as hierarchic, test-per-scan, BILBO (built-in logic block observer), PRPG-MISR (pseudo random pattern generator - multi input signature register), circular BIST, and Reconfigurable Matrix Based Built-In Test Processor (RMBITP) [20, 29, 6]. Even though techniques such as RMBITP are successful at providing BIST for designs as large as 5 million gates with only around 11% overhead, they all suffer from two main disadvantages. The first is that they require the circuit's operation to go off-line periodically to feed in the test patterns. The second is a bootstrapping problem: if the testing logic fails we will never know if the rest of the circuit is functioning properly.

Voting systems solve both these problems because on one hand faults are detected immediately during on-line operation and on the other hand the testing logic is usually small – all at the expense of complete redundant copies of the main circuit. Redundancy with on-line checking can be at the level of cells in a VLSI array [14, 11] or at the level of circuit modules, which could even be diverse designs of the same module, failing in different ways [2]. These, and other techniques for on-line diagnosis [18, 3], share the problem that the benefits of self-checking must outweigh the higher fault rate due to increased silicon area.

There are examples in the evolutionary electronics literature of designs that operate in surprising or intricate ways [7, 17, 19, 10, 16, 21, 13]. A first attempt at harnessing this creativity for BIST design evolved a hybrid on-line/off-line BIST for both a full adder and a two bit multiplier and a full on-line BIST for the adder, all evolved BIST solutions requiring roughly half the component overhead of their conventional equivalents [5]. Some questions posed in that paper are still relevant to new evolved BIST solutions:

1. Do they reuse logic for the main task and BIST?
2. Are these solutions competitive with conventionally designed ones?
3. Are there any principles of operation we could extract from them, perhaps to add to our own conventional design toolset?
4. Does this method scale up for larger problems?

Section 2 will describe the Genetic Algorithm (GA), the simulator, the fault model, the tasks to be evolved and the fitness evaluation mechanism for BIST. Section 3 presents the results achieved while section 4 discusses what was learned and future avenues.

2 Method

2.1 The Genetic Algorithm

A generational GA is used with a population size of 32 with 2 elites where 60% of the next generation is created through mutation and the rest by single-point crossover. Following from earlier work [23] we adopted the model of a small genetically semi-converged population evolving for many generations. Fitness ranges from 0 (worst) to 1 (best). An adaptive mutation rate is used, analogous to a *Simulated Annealing* strategy moving from "exploring" to "exploiting". For each individual exactly m mutations are made,

where $m = \lfloor k_r \times \ln\left(1/\overline{f}\right) \rfloor + m_{\text{floor}}$, $k_r = \frac{m_{\text{roof}}}{\ln(1/f_{\min})}$, $\overline{f}$ is the current average fitness, $f_{\min}$ is minimum possible fitness above 0, $m_{\text{floor}} = 1$ is the number of mutations applied as $\overline{f}$ reaches 1, and $m_{\text{roof}} = 10$ is the number of mutations that would be applied if $\overline{f} = f_{\min}$. This assumes that $\overline{f} > 0$ which is safe under the settings used in this paper. Informal preliminary experiments indicated that solutions were found in less generations than with a constant mutation rate. Linear rank selection is used, such that the elite has twice the selective advantage of the median of the population.

The genotype-phenotype mapping used is similar to [16] excepting that: the locations of circuit outputs are fixed, there are no limitations on connectivity allowing sequential circuits, and the genotype is encoded in binary. The genotype length depends on the task being evolved. To allow implementation across a network of processors, an island based model was used [22] with a low migration rate. This population structure may have aided the search, but the details are not thought to be crucial to the results.

2.2 The Simulator

The simulator used is a simple version of an event driven digital logic simulator in which each logic unit is in charge of its own behaviour when given discrete time-slices and the state of its inputs. Logic units are Look-Up Tables (LUT) of two inputs capable of representing any two input logic gate. Any unit can be connected to any other allowing sequential circuits, so care must be taken to update all units "simultaneously". This is achieved by sending the time-slices to the logic units in two waves: the first to read their inputs and the second to update their outputs. During each evaluation, circuit inputs were kept stable for 25 time-slices and the outputs were read in the second half of this cycle allowing them time to settle.

Gate delays are simulated in time-slice units and are randomized with a Gaussian distribution ($\lfloor \mu = 1.5, \sigma^2 = 0.5 \rfloor$). This amounts to a noisy fitness evaluation with the intention to facilitate transfer of sequential circuits to real hardware as in a "Minimal Simulation" [8]. At the start of each generation, e different sets of logic delays are generated, simulating e different variations to the circuit delays from manufacturing or environmental variation. Each individual's performance is then evaluated in each of the e conditions, and its fitness is the mean value. The number e is occasionally adjusted by hand during the run, such that more evaluations are used as the population leaves the "exploring" stage and enters the "exploiting" stage. It was set such that twice the standard error of the series of fitness trials is significantly smaller than the difference in fitness between adjacent individuals in the rank table with non-equal fitnesses.

The Single-Stuck-At (SSA) Fault model was chosen because it simulates the most common type of on-line failure – that produced by radiation hazards in the absence of mishandling or manufacturing defects [20, 1]. Moreover, it is an industry standard and it has been shown that tests generated for SSA faults are also good at detecting other types of faults. SSA faults can be introduced at any of the logic units of the simulator simply by setting its output always to 0 or 1.

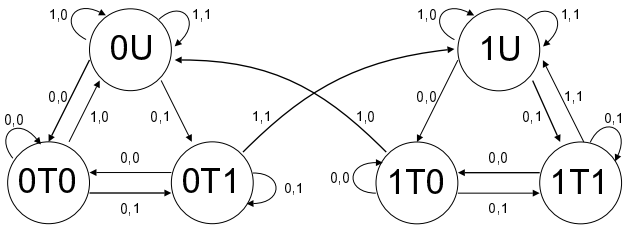

Figure 1. Directed graph of ETDL Moore FSM used to generate randomized test patterns.

2.3 The Tasks

Three simple problems were chosen: a full adder, a two bit multiplier and an edge triggered D-latch (ETDL). The adder was chosen as a small combinational circuit complex enough to be a good starting point for attempting to evolve BIST. It has three inputs A, B, C_{in} and two outputs S, C_{out} such that $S = A \oplus B \oplus C_{in}$ and $C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in}$. The multiplier, chosen as a step up from the adder, has four inputs $A_1 A_0 B_1 B_0$ and four outputs $P_3 P_2 P_1 P_0$ where $P = A \times B$. The D-latch, chosen as a starting point for sequential circuits, has two inputs C, D and one output Q such that Q holds the value D had during C's most recent rising edge.

Since the networks evolving for the combinational tasks may be recurrent, these could show an unwanted dependence on the order in which inputs are presented, and on the networks' internal state. To demand insensitivity to input ordering, the same approach was taken as for the randomization of logic delays (above): at the start of each generation, e (the same number e defining the number of evaluations with random gate delays above) different orderings of the full set of possible inputs for that task were generated, and the individuals of that generation evaluated on all of them. On each of the e evaluations the circuit state was reset, then the ordering of the full set of inputs was presented twice in sequence, to prevent dependence on initial conditions.

The same procedure was carried out for the sequential task excepting the random test pattern generation: a directed graph is built from the Moore finite state machine (FSM) of

the ETDL as in Fig. 1 with six nodes representing the untriggered (`0U,1U`), triggered-0 (`0T0,1T0`) and triggered-1 (`0T1,1T1`) states for both current outputs 0 and 1, and with edges representing state transitions such that their labels contain the input vector (C, D) driving the transition. We choose `0U` or `1U` at random as a start node and add vectors (0,0),(1,0) or (0,1),(1,1) respectively to the new test pattern. A random walk is now begun such that walking along an edge appends its label to the test pattern being generated and removes it from the graph. Thicker edges in Fig. 1 are not removed. The walk ends when all edges have been walked along ensuring the random test pattern generated will test all state transitions of the desired FSM. Test patterns generated using this method are under 4% the size of those used in previous attempts at evolving sequential circuits of similar size [15, 12] and are also exhaustive enforcing the correctness of solutions. These test patterns may not detect extra unwanted behaviour encoded by FSMs with greater number of states. However these FSMs are very unlikely to evolve under the enforced circuit size limit and parsimony pressure.

The task evaluation score was measured as follows. Let Q_r be the concatenation of the series of values at the r^{th} output bit for the final 12 time-slices of the presentation of each input vector during an evaluation, and Q'_r the desired response. We take the modulus of the correlation of Q_r and Q'_r, averaged over all N outputs:

$$f_{\text{t}} = \frac{\sum_{r=0}^{N-1} |corr(Q_r, Q'_r)|}{N} \quad (1)$$

2.4 Evolving BIST

An extra output E was recorded from circuits with the aim that it would go high whenever a fault affected any other output. The performance of a circuit at its main task f_{t} and at BIST behaviour f_{b} were evaluated separately. BIST behaviour itself was evaluated with two fitness measures:

1. BIST per fault f_{b_F}: Let u_{f} be the number of faults affecting task performance for which E does not go high during evaluation. Then f_{b_F} encourages faults to be "detectable": $f_{\text{b}_\text{F}} = 1/\left(1 + u_{\text{f}} \times k_{\text{f}}\right)$ where k_{f} was chosen to be 25, to give f_{b_F} good sensitivity when u_{f} is small.

2. BIST per instance f_{b_I}: Let u_{i} be the number of instances of all combinations of SSA faults and input vectors used during evaluation, for which the task output is incorrect but E is low. Then f_{b_I} encourages immediate detection of faults: $f_{\text{b}_\text{I}} = 1/\left(1 + u_{\text{i}} \times k_{\text{i}}\right)$ where k_{i} was chosen to be 200.

Notice that having a high f_{b_F} but low f_{b_I} is similar to offline BIST solutions while having a high f_{b_I} is like an online BIST detecting faults at the first instance they affect circuit behaviour.

u_{f} is measured by evaluating task fitness f_{t} separately under all SSA faults to every unit able to affect the task outputs. The same set of e evaluation conditions chosen for the current generation is used. If f_{t} falls by at least 0.01 due to a fault, then it is considered to affect task performance. u_{i} is measured by comparing the output of the circuit under each fault and input vector to its output for the same input vector under no faults. The output is deemed unaffected if it is the same at steps 12, 18 and 25 of the 25 time steps for which inputs are stable. Hence most faults that induce oscillations will be detected. If during the simulated time E goes high for more than $eSize$ consecutive time slices then it is considered to have gone high. $eSize$ is set at 7: greater than average race conditions yet not excluding a wide range of behaviours. A circuit exhibiting a high E when no faults were in place was deemed to have $f_{\text{b}_\text{F}} = f_{\text{b}_\text{I}} = 0$.

It seems foolish to try to detect faults at every instance when some faults are not detected at any instance. The objectives were given the priority $f_{\text{t}} > f_{\text{b}_\text{F}} > f_{\text{b}_\text{I}}$, and when sorting the individuals for rank selection the comparison operator only considered an objective if higher priority objectives were equal. An extra objective encouraging parsimony was also used, having the lowest priority of all.

3 Results

3.1 Bloated Full Adder with Small Glitch BIST

For this run the maximum number of logic units was constrained to 13 and the genotype length was 156 bits. A full adder can be implemented with a minimum of 5 units and its BIST using the TPG-DUT-TRE architecture would require 13 while a voter with two copies requires 8 extra units.

This run was started from a population of random individuals, did not include parsimony pressure and used older fitness functions $f_{b_F} = \frac{\text{faults correctly diagnosed}}{\text{faults tested}}$ and $f_{b_I} = \frac{\text{instances correctly diagnosed}}{\text{instances tested}}$ that encourage anti-parsimony since $\frac{n+1-x}{n+1} > \frac{n-x}{n}$ until $x = 0 \Rightarrow f_{b_F} = f_{b_I} = 1$. During this run $eSize$ was set to 3, so all high pulses at E of 3 or more time-slices were interpreted as flagged faults. This meant false flags during fault-free evaluation, resulting in $f_{b_F} = f_{b_I} = 0$, were hard to avoid due to race conditions.

The final elite of this run shown in Fig. 2 displays a modular decomposition which is striking given it evolved under no external aid or incentive towards modularity (such as [10]). As expected all 13 available units are used, yet the task module α uses the minimum of 5 and the BIST module β requires only 4 (unit 3 can safely be skipped feeding C_{in} to unit 5) to achieve $f_{b_F} = 1$ and cover 96% of all fault-input vector instances. The 3 units forming module γ

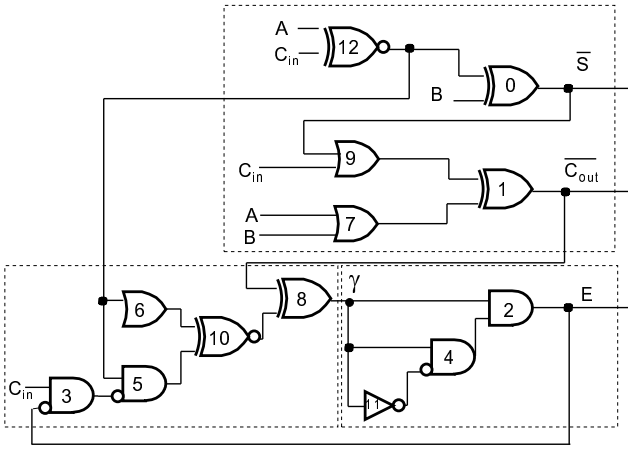

Figure 2. Evolved full adder with on-line BIST robust to race conditions is modular.

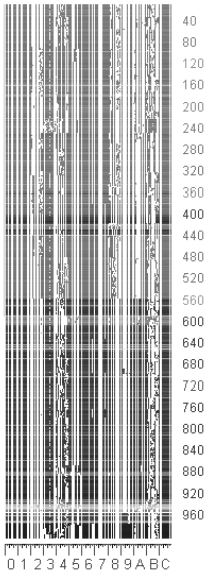

Figure 3. Best elite genotypes as bitmaps through the second half of the adder run with darkness proportional to fitness. Units 0,1,7,9,12(C) (module α) settled in the first half of the run. Units 3,5,6,8,10,2 (used in mod.β) settle at generation labelled 560 increasing fitness. Units 11(B) and 4 (mod.γ) settle last near the end.

have remarkably evolved to become a low pass filter dealing with race conditions in E by cutting glitches by 2 gate delays. Self-diagnosis is achieved through a voting system that exploits design diversity (unit 10=$\overline{C_{out}}$) and by cascading outputs using non-canalizing Xor gates. Modules were incorporated into the design incrementally (Fig. 3), speculation attributes this to the natural dependencies of the problem coupled with the fact that completion of previous modules provides re-encodings of inputs useful for the evolution of further modules.

This circuit can be trimmed to be smaller than the hand designed voter while achieving nearly full on-line fault coverage. By never emitting large undesired E pulses it can be clocked twice as fast as the hand designed voter which emits them of up to 6 time-slices. This circuit is also more robust to large delay discrepancies.

3.2 Two Bit Multiplier with Full On-Line BIST

The maximum circuit size allowed in this run was 28 LUTs and the genotype length was 392 bits. The smallest implementations of a two bit multiplier use 7 two input logic gates [16], while adding BIST costs 22 extra gates using TPG-DUT-TRE and 14 extra gates using a voter.

The circuit of Fig. 4 is a two bit multiplier with full on-line BIST using only 9 extra gates which evolved after around 4 million generations from a population of random individuals. Its outputs form a chain that uses non-canalizing Xor gates to cascade errors down to P_0. Every output computes its value using its predecessor in the chain while E compares P_0 to what it should be (unit 23=$A_0 \cdot B_0$), rather like a checksum. Logic is seamlessly used for multiplication and BIST in a non-modular structure which evolved under parsimony pressure from a modular structure 3 million generations earlier which was the first full on-line BIST solution using 14 extra gates. Some unit pairs are duplicates (13-24,5-23,8-11) and when merged the resulting circuit has only 6 extra gates and 90% on-line fault coverage.

Using just over half the overhead of the hand designed voter it has a totally unconventional structure and a worst case performance penalty of 4 gate delays.

3.3 Full On-Line Edge Triggered D-Latch

The maximum number of logic units was constrained to 14 and the genotype was 168 bits. The number of evaluations e was set to 20 and $eSize$ was set to 15 to accommodate the increased sensitivity of sequential circuits to gate delays. Half of the individuals previously created through mutation were in this run created through a gene copy operator. Dividing the genotype into 14 genes each defined by 12 adjacent bits, the gene copy operator randomly picks

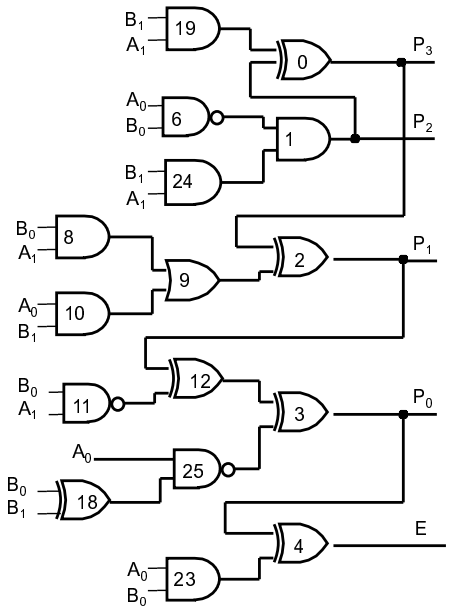

Figure 4. Evolved two bit multiplier with full on-line BIST using 9 extra gates.

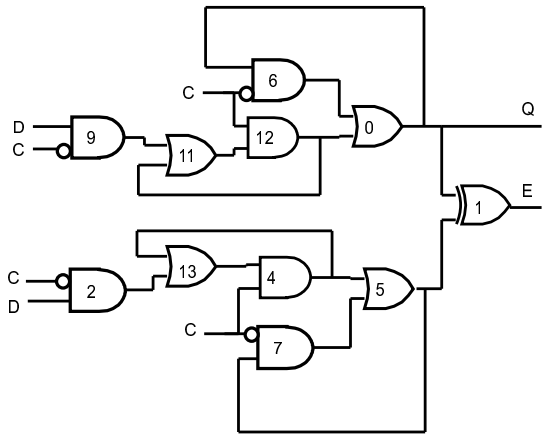

Figure 5. Evolved full on-line BIST solution for ETDL using 6 extra gates.

source and destination genes and overwrites the contents of the latter with the former's. An ETDL can be implemented with a minimum of 5 two input logic gates while adding a voter BIST requires 6 extra gates.

Around 3 million generations after the run was started from a population of random individuals, the elite was the circuit shown in Fig. 5. Its full on-line self-diagnosing capacity is provided by a voting system with two identical copies of an unconventional D-latch version. The exactness of the copies may be attributed to the gene copy operator. The correspondence in structure between this circuit and the conventional voter could be due to the fact that the voter is an optimal full on-line BIST solution for an ETDL. This may not be the case for more complex sequential circuits.

This is the first BIST solution evolved for a sequential circuit, it has full on-line fault coverage and the same amount of overhead as the conventional solution.

4 Conclusion

The method set out in [5] has been successfully applied to the evolution of more complex on-line BIST behaviours: a full adder with BIST robust to race conditions, a two bit multiplier with full BIST and a sequential edge triggered D-latch with full BIST. Answering the questions set out in §1: evolved circuits with BIST can reuse components for the main task and BIST functionality and they are competitive in overhead to conventional solutions. These solutions would function correctly in real hardware because the slightly unconventional simulator does capture the processes and variations influencing combinational circuits. Furthermore, the simulated variation in gate delays that evolved sequential circuits are robust to is at least an order of magnitude greater than the variation likely to be found in real hardware. As well as SSA faults, all circuits diagnosed *transient faults* and *delay faults* which could uncover *parametric faults* such as those occurring in [23]. Evolution has explored design space containing established methods for BIST design and beyond. Some solutions use a "checksum" formula on the current circuit state to evaluate correctness, others increase testability by cascading outputs so that errors are propagated to a single output and others exploit design diversity to minimize redundancy in a voting system. These methods could prove useful to be adopted by designers. For circuits larger than those considered here, it is still unclear whether the enhanced BIST strategies produced by evolution would be worth the computational effort needed to produce them since the evolution of large BIST circuits faces the same problems – and perhaps, solutions – as with other circuits [9, 25, 26, 28]. However, Vassilev et al. [27] have suggested that when large circuits can be evolved (perhaps from a hand-designed seed), their size leads to greater scope for evolutionary optimization.

Most evolved solutions arrived at modular designs (as in [24, 16, 5]). Further understanding of this effect may aid future experiments. The modularity of the circuits can be useful in identifying design patterns or principles. Any single circuit will have its own characteristics for which a particular BIST strategy is most suited. Evolution through blind variation and selection is capable of searching for this strategy without constraints.

The evolution of self-diagnosing analog hardware is an interesting possibility where the E line could give an estimate of *how* wrong the outputs are. Future work could also include the adoption of a more comprehensive fault model simulating multiple faults, LUT memory or routing failure – as in FPGAs exposed to radiation – or other common failure modes. Larger circuits could be tackled perhaps using techniques set out in [9, 25, 26, 28], a silicon area calculation for a given technology could replace the current component counting parsimony measure, and an extra fitness measure of the performance penalty incurred due to the BIST logic could be added.

Acknowledgments

Thanks to Rahel Kunz for invaluable proof-reading and to the COGS bursary that supports Miguel Garvie's research.

References

[1] The NASA/GSFC Radiation Effects and Analysis Home Page. `http://radhome.gsfc.nasa.gov/`.

[2] A. Avizienis and J. P. J. Kelly. Fault-tolerance by design diversity: Concepts and experiments. *Computer*, 17(8):67–80, August 1984.

[3] D. Bradley and A Tyrrell. Immunotronics: Novel finite state machine architectures with built in self test using self-nonself differentiation. *IEEE Transactions on Evolutionary Computation*, 6(3):227–238, 2001.

[4] C. Dufaza. Theoretical properties of LFSRs for built-in self test. *INTEGRATION, the VLSI journal*, 25:17–35, 1998.

[5] M. Garvie and A. Thompson. Evolution of self-diagnosing hardware. In A. Tyrrell, P. Haddow, and J. Torresen, editors, *Proc. 5th Int. Conf. on Evolvable Systems (ICES'03): From biology to hardware*, volume 2606 of *LNCS*, pages 238–248. Springer-Verlag, 2003.

[6] H. Golnabi and J. Provence. RMBITP: A reconfigurable matrix based built-in self-test processor. *Microelectronics Journal*, 28:115–127, 1997.

[7] T. Higuchi, M. Iwata, and L. Weixin, editors. *Proc. 1st Int. Conf. on Evolvable Systems: From Biology to Hardware*, volume 1259 of *LNCS*. Springer-Verlag, 1997.

[8] N. Jakobi. Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In Phil Husbands and Inman Harvey, editors, *Proc. 4th Eur. Conf. on Artificial Life (ECAL'97)*, pages 348–357. MIT Press, 1997.

[9] T. Kalganova. Bidirectional incremental evolution in extrinsic evolvable hardware. In J. Lohn, A. Stoica, and D. Keymeulen, editors, *The Second NASA/DoD workshop on Evolvable Hardware*, pages 65–74, Palo Alto, California, 13-15 2000. IEEE Computer Society.

[10] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane. Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming. In T. Higuchi, M. Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems: From biology to hardware (ICES'96)*, number 1259 in LNCS, pages 312–326. Springer-Verlag, 1996.

[11] J. Lach, W. Mangione-Smith, and M. Potkonjak. Low overhead fault-tolerant FPGA systems, 1998.

[12] J. Lohn, G. Larchev, and R. DeMara. A genetic representation for evolutionary fault recovery in virtex fpgas. In A. Tyrrell, P. Haddow, and J. Torresen, editors, *Proc. ICES'03:From biology to hardware*, volume 2606 of *LNCS*, pages 47–49. Springer-Verlag, 2003.

[13] J. Lohn, A. Stoica, D. Keymeulen, and S. Colombano, editors. *Proc. 2nd NASA/DoD workshop on Evolvable Hardware*. IEEE Computer Society, 2000.

[14] D. Mange, A. Stauffer, and G. Tempesti. Embryonics: A microscopic view of the molecular architecture. In A. Perez-Uribe M. Sipper, D. Mange, editor, *Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98): From biology to hardware*, volume 1478 of *LNCS*, pages 285–195. Springer-Verlag, 1998.

[15] C. Manovit, C. Aporntewan, and P. Chongstitvatana. Synthesis of synchronous sequential logic circuits from partial input/output sequences. In A. Perez-Uribe M. Sipper, D. Mange, editor, *Proc. ICES'98: From biology to hardware*, volume 1478 of *LNCS*, pages 98–105. Springer-Verlag, 1998.

[16] J. Miller, D. Job, and V. Vassilev. Principles in the evolutionary design of digital circuits - part I. *Genetic Programming and Evolvable Machines*, 1(3), 2000.

[17] J. Miller, A. Thompson, P. Thomson, and T. Fogarty, editors. *Proc. 3rd Int. Conf. on Evolvable Systems (ICES2000): From Biology to Hardware*, volume 1801 of *LNCS*. Springer-Verlag, 2000.

[18] N. Shnidman, W. Mangione-Smith, and M. Potkonjak. On-line fault detection for bus-based field programmable gate arrays. *IEEE Transactions on VLSI systems*, 6(4):656–666, 1998.

[19] M. Sipper, D. Mange, and A. Pérez-Uribe, editors. *Proc. ICES'98:From biology to hardware*, volume 1478 of *LNCS*. Springer-Verlag, 1998.

[20] A. Steininger. Testing and built-in self-test - a survey. *Journal of Systems Architecture*, 46:721–747, 200.

[21] A. Stoica, D. Keymeulen, and J. Lohn, editors. *Proc. 1st NASA/DoD workshop on Evolvable Hardware*. IEEE Computer Society, 1999.

[22] R. Tanese. Distributed genetic algorithms. In J.D. Schaffer, editor, *Proc. of the Third International Conference of Genetic Algorithms*, pages 434–439. Morgan Kauffmann, 1989.

[23] A. Thompson, I. Harvey, and P. Husbands. Unconstrained evolution and hard consequences. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware: The evolutionary engineering approach*, volume 1062 of *LNCS*, pages 136–165. Springer-Verlag, 1996.

[24] A. Thompson and P. Layzell. Analysis of unconventional evolved electronics. *Communications of the ACM*, 42(4):71–79, April 1999.

[25] P. Thomson. Circuit evolution and visualisation. In J. Miller, A. Thompson, P. Thomson, and T. Fogarty, editors, *ICES2000: From biology to hardware*, volume 1801 of *LNCS*, pages 229–240. Springer-Verlag, 2000.

[26] J. Torresen. Evolving multiplier circuits by training set and training vector partitioning. In A. Tyrrell, P. Haddow, and J. Torresen, editors, *Proc. ICES'03:From biology to hardware*, volume 2606 of *LNCS*, pages 228–237. Springer-Verlag, 2003.

[27] V. Vassilev, D. Job, and J. Miller. Towards the automatic design of more efficient digital circuits. In J. Lohn, A. Stoica, and D. Keymeulen, editors, *The Second NASA/DoD workshop on Evolvable Hardware*, pages 151–160, Palo Alto, California, 13-15 2000. IEEE Computer Society.

[28] V. Vassilev and J. Miller. Scalability problems of digital circuit evolution: Evolvability and efficient designs. In J. Lohn, A. Stoica, and D. Keymeulen, editors, *The Second NASA/DoD workshop on Evolvable Hardware*, pages 55–64, Palo Alto, California, 13-15 2000. IEEE Computer Society.

[29] H. Wunderlich. BIST for systems-on-a-chip. *INTEGRATION, the VLSI journal*, 26:55–78, 1998.