Fault-Tolerant Evolvable Hardware Using Field-Programmable Transistor Arrays

Didier Keymeulen, *Member, IEEE*, Ricardo Salem Zebulum, *Member, IEEE*, Yili Jin, *Member, IEEE*, and Adrian Stoica, *Member, IEEE*

***Abstract*—The paper presents an evolutionary approach to the design of fault-tolerant VLSI (very large scale integrated) circuits using EHW (evolvable hardware). The EHW research area comprises a set of applications where GA (genetic algorithm) are used for the automatic synthesis and adaptation of electronic circuits. EHW is particularly suitable for applications requiring changes in task requirements and in the environment or faults, through its ability to reconfigure the hardware structure dynamically and autonomously. This capacity for adaptation is achieved via the use of GA search techniques. In our experiments, a fine-grained CMOS (complementary metal-oxide silicon) FPTA (field-programmable transistor array) architecture is used to synthesize electronic circuits. The FPTA is a reconfigurable architecture, programmable at the transistor level and specifically designed for EHW applications.**

The paper demonstrates the power of EA to design analog and digital fault-tolerant circuit. It compares two methods to achieve fault-tolerant design, one based on fitness definition and the other based on population.

The fitness approach defines, explicitly, the faults that the component can encounter during its life, and evaluates the average behavior of the individuals. The population approach, on the other hand, uses the implicit information of the population statistics accumulated by the GA over many generations.

The paper presents experiment results obtained using both approaches for the synthesis of a fault-tolerant digital circuit (XNOR) and a fault-tolerant analog circuit (multiplier). The experiments show that the EA (evolutionary algorithm) can synthesize fault-tolerant designs for both the analog and digital functions circuits that can recover for functionality when lost due to a-priori unknown faults by finding new circuit configurations that circumvent the faults.

The paper shows that although the classic fault-tolerant design approach is able to create a reliable circuit design by evaluating the behavior of the circuit when well known faults are injected during the evolutionary process, better circuit performance, in less computation time, for a same fault-tolerant degree is achieved by allowing the evolutionary design process to be free of all faults constraints.

***Index Terms*—Evolvable hardware, fault tolerance, genetic algorithm, hardware–software co-design.**

Manuscript received December 27, 1999. The research in this paper was performed at the Center for Integrated Space Microsystems, Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the "U.S. Defense Advanced Research Projects Agency (DARPA) under the Adaptive Computing Systems Program" and by the U.S. National Aeronautics and Space Administration (NASA).

D. Keymeulen, R. Zebulum, and A. Stoica are with the Jet Propulsion Lab, M/S 303-300, Pasadena, CA 91109 USA (e-mail: {Didier.Keymeulen; Ricardo.Zebulum; Adrian.Stoica}@jpl.nasa.gov).

Y. Jin is with the MS 2-83/TP63; California Institute of Technology, Pasadena, CA 91125 USA (e-mail: Yili.Tin@caltech.edu).

Publisher Item Identifier S 0018-9529(00)11755-5.

ACRONYMS[1]

ASIC	application specific IC
CMOS	complementary metal-oxide silicon
EA	evolutionary algorithm
EHW	evolvable hardware
FT	fault tolerance
FPAA	field-programmable analog array
FPGA	field-programmable gate array
FPTA	field-programmable transistor array
GA	genetic algorithm
GAL	gate array logic
IC	integrated circuit
SPICE	simulation program with IC emphasis
VLSI	very large-scale integration

I. INTRODUCTION

LONG-TERM survivability of space systems, as required for example by outer solar system exploration and by missions to comets and planets with severe environmental conditions, has recently been approached with new ideas, such as the use of biology-inspired mechanisms for hardware adaptation. The application of evolution-inspired formalisms to hardware design and self-configuration lead to the concept of EHW. In the narrow sense EHW refers to self-reconfiguration of electronic circuits, such as those based on FPGA, by evolutionary/genetic reconfiguration mechanisms. In a broader sense EHW refers to the self-reconfiguration of a broader class of hardware, ranging from sensors and antennas to complete space systems that could adapt to changing environments and, moreover, increase their performance during the mission.

Evolutionary computation encompasses a class of search algorithms that use some aspects of natural evolution as metaphors, and, for EHW, are used for the autonomous synthesis and adaptation of electronic circuits. In particular, most of these algorithms borrow ideas from natural selection, and mimic biological mechanisms such as: genetic material recombination and mutation. GA [32] is the most widely used EA. Instead of focusing on just one potential solution to a problem, it samples a population of potential solutions. A population of individuals is initially randomly generated. Each individual is a string that encodes, by means of a particular mapping, a potential solution to the problem. Individuals are also known as chromosomes. The GA then performs operations of selection, crossover, and mutation over these individuals, corresponding, respectively, to the principles of survival of

[1]The singular and plural of an acronym are always spelled the same.

the fittest, recombination of genetic material, and mutation observed in nature. The selection step is probabilistic, but it favors individuals that have been assigned higher fitness indexes in an evaluation step performed beforehand. The fitness is a scalar measure of the performance of an individual according to the problem specification. The crossover operator splices the contents of two randomly chosen strings, producing two new individuals or offspring. The mutation operator changes a particular string position at random and is applied with a low probability of occurrence. The search process is done through the generation of successive populations until a stop criterion is met. It is anticipated that the average population fitness gradually increases along the generations.

A variety of circuits have been synthesized through evolutionary means. For example, [1] used genetic programming to grow "embryonic" circuits that satisfy desired requirements. This approach was used for evolving a broad range of circuits, including filters and computational circuits. An alternative encoding technique for analog circuit synthesis, which has the advantage of reduced computation load, was used in [2] for automated filter design. For digital circuit synthesis, [31] describes the AdAM system which applies GA at the hardware description language level of the circuit. Analog circuits [1], [2] were evolved in simulation, without concern for a physical implementation, but rather as a proof-of-concept. These experiments demonstrated that evolution can lead to designs that meet, or in certain cases exceed, performance of those designed by humans. On-chip evolution was demonstrated for the first time in [30]. Using GA, a Gate Array Logic (GAL) circuit was used to design a multiplexer and an adder. On-chip evolution was also demonstrated later in [3] using an FPGA as the programmable device, and a GA as the evolutionary mechanism, and in [24] using dedicated hardware for the circuit and the GA computation. References to current work in evolvable hardware are in [4]–[7], [22], [34]. More recently, evolutionary experiments were performed on FPAA [16] and custom-designed ASIC [9], [23]. However current programmable analog devices are very limited in capabilities. EA has been used with success for designing fault-tolerant systems, especially in the domain where the fitness function was noisy such as robotics (e.g., if it involves taking error-prone measurements from a real-world process such as the vision system of a robot) [13], [19] and recently also in electronics [14], [20], [21].

EHW can bring two main benefits to spacecraft survivability.

1) It can generate new functions (more precisely new hardware configurations can be synthesized to provide required functionality) when needed.
2) It can help preserving existing functions, in conditions where the hardware is subject to faults, aging, temperature drifts, radiation, etc.

The fault-tolerant property is extremely important for electronic components used in the space and nuclear industries where they are continuously subjected to radiation. As the limits of VLSI technology are pushed toward sub-micron levels to achieve higher levels of integration, devices become more vulnerable to radiation-induced errors. These radiation-induced errors can lead to system failure. One of the goals of future electronics is to design radiation-immune electronic components [18]. More generally, the development of novel fault-tolerant methods for circuit design will benefit not only aerospace applications, but fields with extreme temperature and radiation environments.

Our mission, therefore, is to design and develop electronic components and systems that are inherently insensitive to faults, such as silicon defects, by using on-board evolution in hardware to achieve fault-tolerant or highly reliable systems. The evolution can self-repair on-line by changing the circuit configuration in a short time or off-line self-repair by pushing further the evolution and exploiting defective components as if they were working parts [13], [14]. Another main advantage of GA, when applied to the domain of real-world electronics, compared to a one-candidate-solution-at-a-time search method (such as simple hill climbing or simulated annealing) is the robustness of the GA [28]. This is because GA work by accumulating fitness statistics over many generations; conversely, hill climbing or simulated annealing can be irrecoverably led astray. This paper reports experiments that illustrate how EA, using two different approaches, can design fault-tolerant analog and digital circuits, and recover functionality when lost due to faults, by finding new circuit configurations that circumvent the faults. The search for an electronic circuit realization of a desired transfer characteristic is made in software as in extrinsic evolution and in hardware as in intrinsic evolution. In intrinsic evolution the hardware actively participates in the circuit evolutionary process and is the support on which candidate solutions are evaluated. The FPTA reconfigurable chip, specifically designed for EHW experiments, has been used in these applications.

Section II presents the fault-tolerance principles and the evolutionary method to obtain fault-tolerant systems. Section III presents the FPTA concept. Section IV presents the experimental setup, including details of the evolutionary design tool, the FPTA chip, and the hardware test bed. Section V describes the fault-tolerant experiments to design by extrinsic evolution (using a SPICE simulator) an analog multiplier and by intrinsic evolution (on FPTA chips) a XNOR logical function. Section VI presents some lessons learned from the experiments.

II. Fault-Tolerance Principles for Evolvable Hardware

The way this paper investigates FT differs from the conventional way FT is studied. Instead of focusing on manufacturing problems, such as defective level or realistic faults [25], [26], this paper concentrates on component failures that can occur after a long period of circuit operation, possibly due to component aging.

The definition of FT is: A fault in a component does not cause the overall system to malfunction [12]. In general FT is considered that "no single failure" causes the system to malfunction, but in equipment such as manned and unmanned space vehicles, the system can be required to tolerate multiple failures before the system malfunctions. The malfunction is usually a "loss of service" that can be total or partial, e.g., on a computer network. The characteristic of FT is not absolute. It is important to note

that no system can be truly made tolerant to every possible combination of faults, and there are always some combinations of events and failures that lead to the disruption of the system. The question is one of degree: the amount of "required tolerance to faults" varies from application to application. In our electronic experiment, the malfunction is calculated by the mean square error between a desired output transient characteristic and the actual output.

Fault tolerant systems are evaluated by two criteria: their reliability and availability. The reliability measures how long the system can operate before malfunctioning even in the presence of faulty components. The availability measures the mean proportion of time that the system is available for use. In our experiments, circuit

- reliability is measured by evaluating the malfunction of the electronic circuit when injecting faults;
- availability is measured by calculating the time needed by the evolution process to retrieve a satisfactory circuit design. There are four principles for designing fault-tolerant systems [27]:
  1) redundancy,
  2) fault isolation,
  3) fault detection and annunciation,
  4) on-line repair.
- Redundancy is well understood: if part of a system fails, there is an "extra or spare" that can operate in place of the failed component such that the system operation is uninterrupted.
- Fault-isolation consists of containing or isolating the fault to prevent a single failure's causing multiple failures that can easily cause a system malfunction.
- Fault detection and annunciation require that the system must first detect the failure (avoiding latent failure) and, once detected, it must announce that malfunction in some manner so that a repair can be made.
- On-line repair requires that the system with a failed component be made unavailable as less as possible while the system is in service. Three of these four principles can be applied to fault-tolerant evolutionary design. Redundancy is obtained by using a circuit with many connections and elements (transistors). Fault detection and annunciation is obtained by evaluating continuously the circuit performance. On-line repair is obtained by swapping circuit configuration; it can be performed by searching, in the population, for a correct circuit, or by running the GA during a limited number of generations.

Two approaches were proposed to build fault-tolerant systems using EA:

1) Population-Based Fault-Tolerant Design: It consists of extracting, from a population of evolved circuits, the individual that adequately performs a task in the presence of a fault. The evolution process is continued to attain a performance equal to that before the fault occurred [14].
2) Fitness-Based Fault-Tolerant Design: It consists of introducing the faults during the evolutionary process. Particularly, apply faults known *a priori* that can occur in the circuit during its life-time [17].

While the population fault-tolerant approach uses the population statistics accumulated by the GA without constraining it to particular faults, the fitness fault-tolerant approach needs a previous knowledge of the faults that can occur in the circuit during its life-time. This paper show that the former approach is superior to the latter. Section III presents the FPTA and the evolutionary platform on which the experiments are conducted; then describes the experiments and their results.

III. FPTA for Evolvable Hardware

The FPTA idea was introduced first in [9]. FPTA is a concept design for hardware reconfigurable at transistor level. As both analog and digital CMOS circuits ultimately rely on functions implemented with transistors, the FPTA appears as a versatile platform for the synthesis of both analog and digital (and mixed-signal) circuits. Currently available reconfigurable devices can be classified as FPGA or FPAA [8]. Most FPGA models consist of an arrangement of cells that perform digital logic, such as basic gates, multiplexers, and flip-flops [8]. The user can configure the cells' connections and, in some models, their functionality. The FPGA analog counterparts, the FPAA, are usually regular architectures consisting of a matrix of CAB (configurable analog block). These CAB are often nominally identical, being made of 1 OpAmp (operational amplifier) with programmable interconnections. While FPGA were developed for applications in the domain of digital-signal processing and re-configurable computing, most FPAA models are being developed for applications in programmable mixed-signal circuits and filtering. In addition to the intrinsic flexibility of CAB, which confers advantageous features to standard electronic design, the FPGA and FPAA are also the focus of research in the Evolvable Hardware. Contrasted with currently available FPGA and FPAA, the FPTA presents a fine granularity, being programmed at the transistor level. This feature is advantageous from the EHW perspective, since it allows the sampling of novel architectures together with the possibility of implementing standard architectures.

The FPTA cell is an array of transistors interconnected by programmable switches implemented with transistors, acting as simple T-gate switches. The status of the switches (`On` or `Off`) determines a circuit topology and consequently a specific response. Thus, the topology can be considered as a function of the switch states, and can be represented by a binary sequence, e.g., "1011 . . .," where by convention 1 implies a switch turned `On` and 0 implies a switch turned `Off`. The FPTA architecture allows the implementation of bigger circuits by cascading FPTA cells. To offer sufficient flexibility the module has all transistor terminals connected via switches to expansion terminals (except those connected to power or ground). Further issues related to chip expandability are treated in [9]. Fig. 1 is an example FPTA cell consisting of 8 transistors and 24 programmable switches; the transistors P1–P4 are PMOS and N5–N8 are NMOS, and the switch-based connections are in sufficient number to allow a majority of meaningful topologies for the given transistor arrangement, and yet less than the total number of possible connections. Programming the switches `On` and `Off` defines a circuit for which the effects of nonzero, finite impedance of the

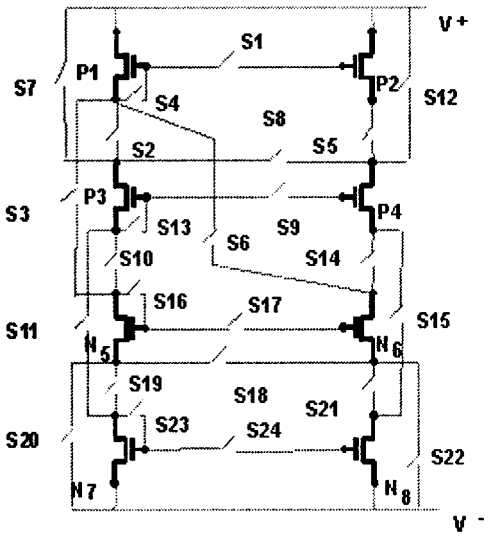

Fig. 1. Module of the FPTA cell.

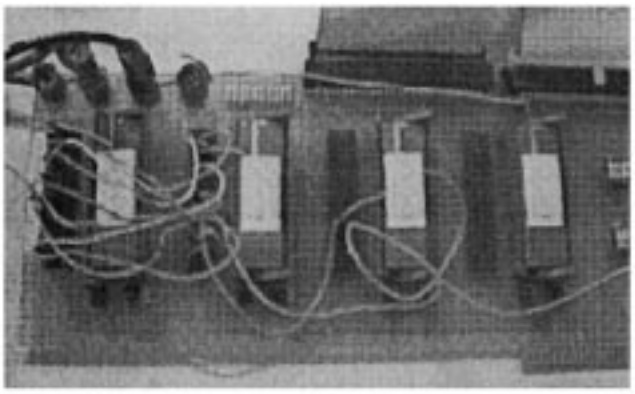

Fig. 2. A test board with 2 cascaded FPTA.

switches can be neglected in the first approximation. One FPTA module was fabricated as a Tiny Chip through MOSIS,[2] using 0.5-micron CMOS technology. Fig. 2 illustrates the test board with 4 chips mounted on it. The board uses 6 external connections between the 2 FPTA and 8 external connections to 2 voltage inputs, 1 probe output, and 1 current bias. (The 2 FPTA used in the experiment are on the left side of the figure.)

In the context of electronic synthesis on reconfigurable devices such as the FPTA, the architectural configurations are encoded in "chromosomes" that define the state of the switches connecting elements in the reconfigurable hardware. Fig. 3 illustrates the main steps in evolutionary synthesis of electronic circuits. First, a population of chromosomes is randomly generated to represent a pool of circuit architectures. The chromosomes are converted into control bit strings, which are downloaded onto the programmable hardware. In the particular case of the FPTA cell, the chromosome has 24 bits that determine the state of the 24 switches (Fig. 1). Second, circuit responses are compared against specifications of a target response using the rms error as the fitness criterion. The individuals are ranked based on their fitness. Preparation for a new iteration loop involves generating a new population of individuals from the pool of the best individuals in the previous generation. Third, individuals are selected probabilistically based on their fitness. Some are taken as they were and some others are modified by genetic operators, such as chromosome crossover and mutation. The process is repeated for several generations, resulting in individuals with increasingly better fitness. The GA is usually ended after a given number of generations, or when the closeness to the target response has been reached. In practice, one or several solutions can be found among the individuals of the last generation. In addition to this procedure (called intrinsic EHW or hardware evolution), Fig. 3 also shows an alternate way to carry on evolutionary circuit synthesis, by using simulators instead of reconfigurable chips. In this particular case, the chromosome is mapped into a SPICE circuit model, which will be simulated and evaluated. The procedure using the simulator is called extrinsic EHW or software evolution. The chromosome is mapped into the circuit netlist by examining the chromosome values bit by bit. According to each bit value (0 or 1), the state of its corresponding switch is set in the circuit netlist. After the states of all the switches are determined, the circuit is simulated.

[2]A low-cost prototyping and small-volume production service for VLSI circuit development [http://www.mosis.com].

IV. Hardware Test Bed for Evolutionary Experiments

An evolutionary design tool EHWPack (Fig. 4) was developed to facilitate experiments in hardware and software evolution [15], as defined in Section III. This tool incorporates the public domain Parallel GA package PGAPack [29] as a genetic engine running on a UNIX workstation. For hardware evolution, the tool was very useful in testing architectures of reconfigurable hardware and in demonstrating evolution on FPTA reconfigurable chips. The chromosomes are downloaded from a UNIX workstation to a PC board with 4 FPTA chips, and the circuit response is read back to the UNIX workstation through a TCP/IP connection. The PC test bed is built around National Instruments data acquisition hardware and software (LabView). An interface-code links the GA with the hardware where potential designs are evaluated, while a Graphical User Interface (GUI) allows easy problem formulation and visualization of results. At each generation the GA produces a new population of binary chromosomes, which get converted into configuration bits for the 4 PTA's reconfigurable chips. Configuration bits are further downloaded by the PC into the FPTA reconfigurable chips by LabView. Referring to software evolution, we incorporated in the EHWPack, the SPICE 3F5 [33] as circuit simulator. An interface code links the GA with the simulator where potential designs are evaluated, while a Graphical User Interface (GUI) allows easy "problem formulation" and "visualization of results." At each generation, the GA produces a new population of binary chromosomes, which get converted into Netlists that describe candidate circuit designs, and are further simulated by SPICE.

V. Fault-Tolerant Experiments

The aim of the experiments in Section V is to test and compare the reliability and availability of a circuit design obtained by, respectively, a population and a fitness-based evolution. Two case studies were investigated: the evolution of 1) an analog

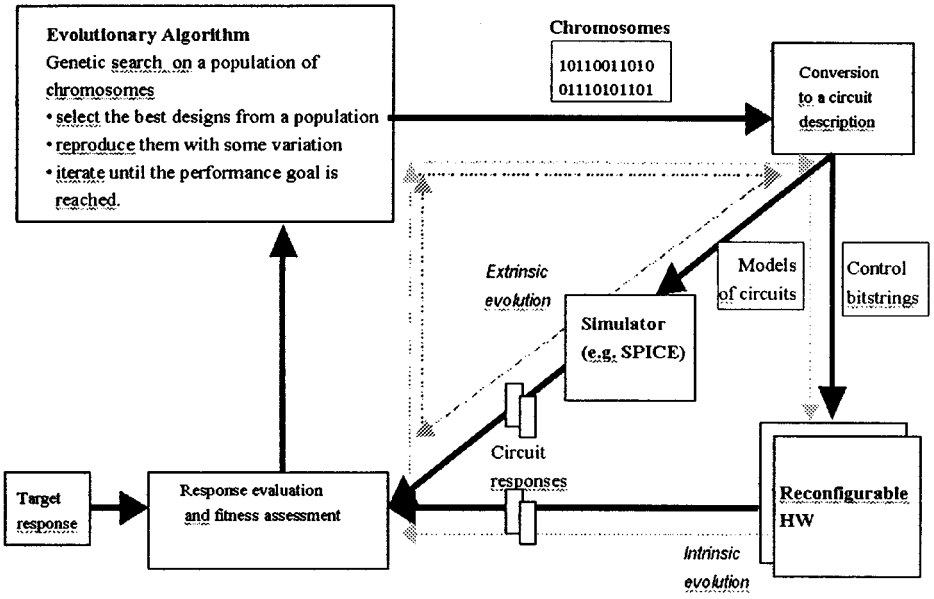

Fig. 3. Main steps for the evolutionary synthesis of electronic circuits.

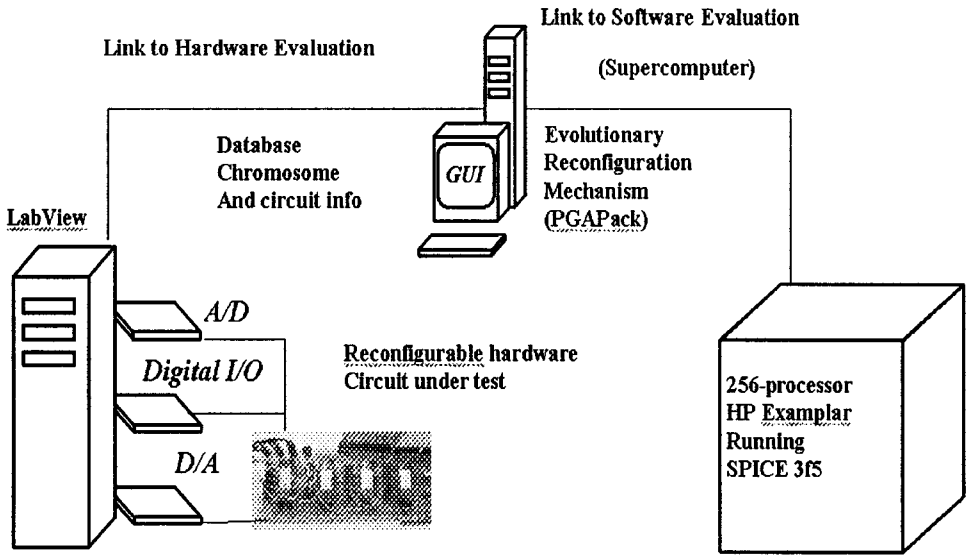

Fig. 4. Environment for evolutionary hardware design.

2-inputs multiplier, and 2) a digital XNOR gate. In 1, software evolution was used, and, in 2, the hardware evolutionary procedure was used. Sections V-A and V-B discuss 2 cases.

A. Software Evolution of an Analog Multiplier

The circuit in Fig. 5 shows that 2 cascaded FPTA cells are used, and are connected by 6 switches. The interconnection switches are associated with the faults to be applied in the circuit, faults 0–5. The chromosome consists of 54 bits: 2 sets of 24 bits controlling the internal switches of the 2 cells, and 6 bits controlling the interconnecting switches. The circuit receives 2 inputs, `In1` and `In2`, and has 1 probed output, Out. Since this is an extrinsic EHW experiment, the circuit is represented by a netlist that is simulated by the SPICE circuit simulator.

Before launching the GA, a fitness function must be defined to evaluate how close each circuit comes to the specification. In this case, a nested DC sweep analysis has been performed, where `In1` and `In2` are swept from 1 to 4 volts in increments of 0.3 volts. The fitness of each circuit is:

$$1 - \sqrt{\frac{\sum_i \sum_j [\texttt{In1}(i) \cdot \texttt{In2}(j) - \texttt{Out}(i,j)]^2}{n}}. \qquad (1)$$

The i and j point to the current values of `In1` and `In2`, respectively. $\texttt{Out}(i, j)$ is the circuit output for a particular input configuration $\texttt{In1}(i)$ and $\texttt{In2}(j)$. The target value is:

$$\texttt{In1}(i) \cdot \texttt{In2}(j).$$

n is the total number of output samples; $n = 121$ in this case. The fitness increases as the rms error to the target response

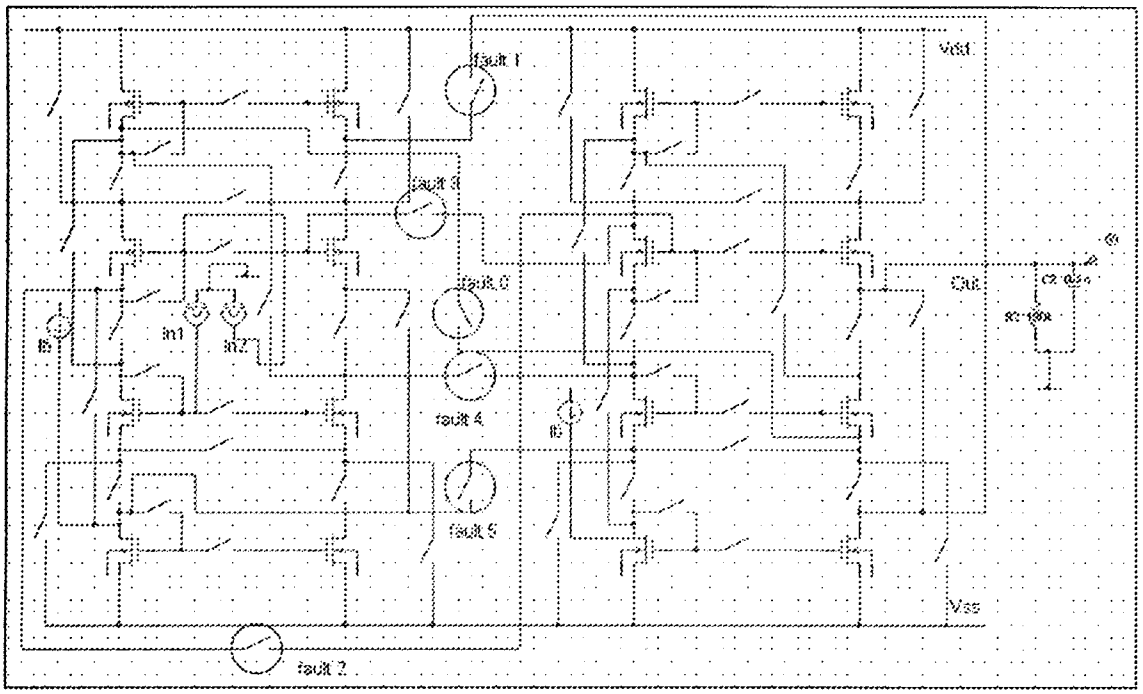

Fig. 5. Cascaded FPTA used to design a fault-tolerant multiplier circuit with 54 switches.

decreases. Since the input and output values are normalized, the fitness $\in [0, 1]$.

Sections V-A-1 and V-A-2 describe two sets of experiments, population based and fitness based. The experiments used a GA with parameters [28]:

population size: 128,
chromosome size: 54,
uniform cross-over probability: 0.7,
uniform mutation probability: 0.04,
tournament selection of size 2,
elite strategy: 50%.

1) Population-Based Experiment: After 60 generations, the GA converged to a particular solution. We then tested this individual against 6 different faults (see Fig. 5):

faults 0, 1, 4 are opened switches;
faults 2, 3, 5 are closed switches.

We also tested the individual against inverse faults, defined as:

inverse faults 3, 5 are opened switches,
inverse fault 4 is closed switch.

These faults were chosen in such a way that the interconnecting switch configuration is opposite to the one attained by the best individual. The performance of the best individual was investigated for these faults. An individual scores the highest fitness among the population when no fault is applied. Also investigated were which individuals in the same population (mutants) could display the best response to each fault; see Table I.

Table I compares the individuals according to their normalized fitness, by applying a transformation where the worst and best fitness are 0 and 1, respectively. The performance of the best individual attained by the GA is compared with the other mutants in the population. When presented with the 6 faults, the best individual performance deteriorated for faults 0, 1, 3, 4, 5. Mutants with better responses for these faults, are shown in Table I. However, we could not find mutants with an acceptable performance for faults 0, 1, 4. This procedure was then carried on for each of these 3 faults: re-starting the GA, and evaluating the individuals under each particular faulty condition. Thus we could "self-repair" the circuit, and increase the fitness for these faults. For fault 0, the GA converged after 100 generations; for faults 1, 4, the GA converged after 60 generations.

TABLE I
ACHIEVED FITNESS—POPULATION—BASED (INV-F≡ INVERSE FAULT)

	Best Individual	Best Mutants	Self Repaired	Number of Evaluations
No fault	**0.9229**	0.9229		7680
Fault 0	0.3797	0.4151	**0.8833**	12800
Fault 1	0.4169	0.6074	**0.8664**	7680
Fault 2	**0.9229**	0.9229		0
Fault 3	0.4386	**0.9168**		128
Fault 4	0.6674	0.8032	**0.9168**	7680
Fault 5	0.6002	**0.9147**		128
Inv-F 3	**0.9229**	0.9229		0
Inv-F 4	**0.9229**	0.9229		0
Inv-F 5	**0.9229**	0.9229		0
Total				36096

Table I also illustrates the performance of the individuals when the inverse of faults 3, 4, 5 are applied. This is equivalent to applying no faults, and thus providing the same results. However, this is an interesting test for the fitness based method.

Referring to the number of evaluations in Table I, the first GA execution took 7680 evaluations. Whenever the best individual provides the best response for a particular fault (fault 2, in Table I), no evaluation is necessary. If the mutant provides the best response to a particular fault, it means that the whole population (128 individuals) had to be re-evaluated for the particular

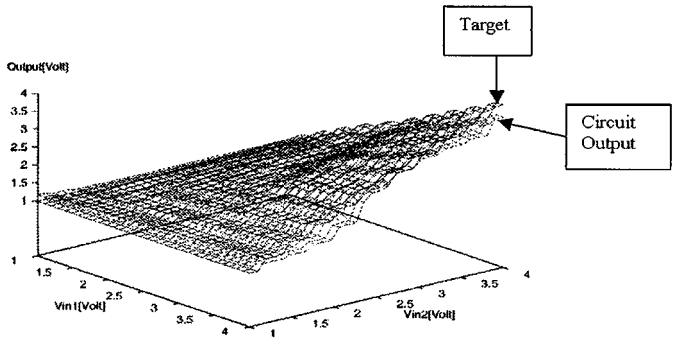

Fig. 6. Response of the best analog multiplier with no faults applied. Population-based experiment.

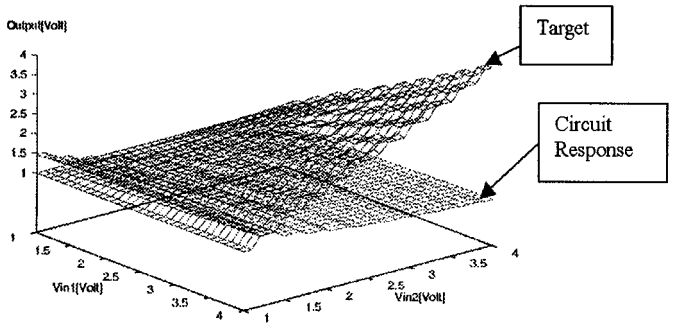

Fig. 7. Response of the best analog multiplier when fault 5 is applied. Population-based experiment.

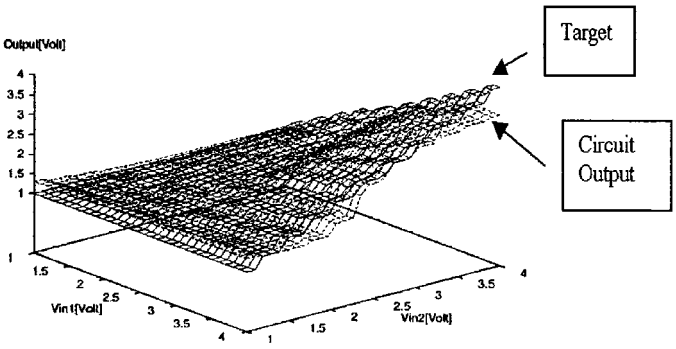

Fig. 8. Response of the best mutant with respect to fault 5 for the multiplier. Population-based experiment.

fault. For the self-repaired individuals, the number of evaluations is the number of individuals multiplied by the number of generations necessary for convergence of the second GA.

Fig. 6 illustrates the best individual response when no fault is applied. Fig. 7 displays the response of the same individual when fault 5 is applied; there is a deterioration in its response when compared to the target. Fig. 8 depicts the response of the best mutant with respect to fault 5. The response again approaches the target.

2) Fitness-Based Experiment: The fitness-based experiment encompasses the evaluation of 7 versions of each individual: 1 with no faults, and 6 with the faults defined in Section V-A-1. It is anticipated to have more robust individuals at the expense of more evaluations. After 50 generations the GA converged to the best individual. Table II shows the results for the fitness-based experiments; it shows the performance

TABLE II
ACHIEVED FITNESS—FITNESS-BASED (INJ ≡ INJECTED; INV-F ≡ INVERSE FAULT)

	Best Individuals	Best Mutants	Number of Evaluations
No fault	**0.9082**	0.9082	44800
Fault 0 (inj)	0.8592	**0.8758**	128
Fault 1 (inj)	**0.9082**	0.9082	0
Fault 2 (inj)	0.8459	**0.8877**	128
Fault 3 (inj)	**0.9052**	0.9052	0
Fault 4 (inj)	0.8697	**0.8807**	128
Fault 5 (inj)	**0.9082**	0.9082	0
Inv-F 3	**0.9082**	0.9031	0
Inv-F 4	**0.9082**	0.9031	0
Inv-F 5	0.8903	**0.8949**	128
Total			45312

TABLE III
COMPARISON OF POPULATION AND FITNESS BASED METHODS FOR THE ANALOG MULTIPLIER (POP-B⇒POPULATION-BASED; FIT-B⇒FITNESS-BASED)

	Best Pop-B	Best Fit-B	Performance
No fault	**0.9229**	0.9082	1.62%
Fault 0 (inj)	**0.8833**	0.8758	0.86%
Fault 1 (inj)	0.8664	**0.9082**	-4.82%
Fault 2 (inj)	**0.9229**	0.8877	3.96%
Fault 3 (inj)	**0.9168**	0.9052	1.28%
Fault 4 (inj)	**0.9168**	0.8807	4.09%
Fault 5 (inj)	**0.9147**	0.9082	0.72%
Inv-F 3	**0.9229**	0.9082	1.59%
Inv-F 4	**0.9229**	0.9082	1.59%
Inv-F 5	**0.9229**	0.8949	3.12%
Average	**0.9112**	0.8985	1.41%

of the best individual and mutants when exposed to faults and inverse faults.

Table II shows that, in contrast to the population-based experiments, the application of inverse fault 5 reduces the performance of the best individual.

3) Comparison of Population and Fitness Based Methods: Table III compares the results for population and fitness based methods; the population-based method achieved slightly superior results for 9 out of 10 faults. Taking an average, the results for the population-based method are 1.4% superior to the fitness-based method. However, taking the number of individuals evaluated in each method (Tables I and II), the population-based method evaluated 20% fewer individuals than the fitness-based method.

A second GA experiment was conducted for both population and fitness based methods, exhibiting similar results to that in this section.

B. Hardware Evolution of a Digital XNOR Gate

The experiment consists of 2 cascaded FPTA but differs from the experiments in Section V-A. Each FPTA is programmed by 24 internal switches. The 2 FPTA are connected by 6 external wires controlled by 6 programmable switches (Fig. 9). The interconnection switches are associated to the six faults to be applied in the circuit, fault 0 to fault 5, by imposing the switches to be always closed or open. Each FPTA is connected through 4 programmable switches to `In1` and `In2`, 1 current bias, and 1 probe output, `Out`. These 4 programmable switches allow the

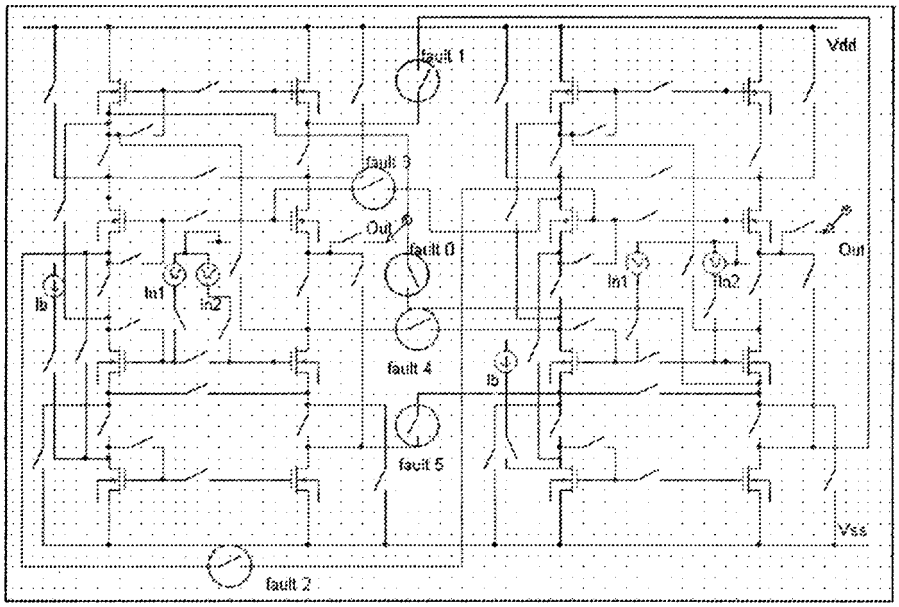

Fig. 9. Cascaded FPTA used to design a fault-tolerant XNOR circuit with 62 switches. The 6 connections indicated by a circle are subjected to — faults 0, 3, 5, are open switches; faults 1, 2, 4 are closed switches.

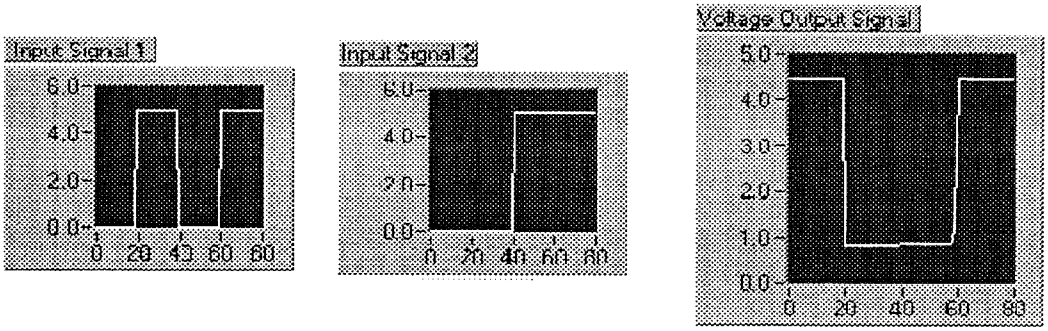

Fig. 10. Input signal 1 (100 Hz), input signal 2 (50 Hz), output signal of XNOR configuration. (x axis: 0.25 msec/unit; switches: 1 volt/unit.

GA to select the terminals in the circuit to apply the input voltages and to collect the output voltage. The 2 cascaded FPTA are programmed by a total of 62 switches which represent the chromosome of the GA (Fig. 9).

The experiment consisted of the evolutionary design of a XNOR logic function using 2 square-wave voltage inputs, at 50 Hz and 100 Hz (Fig. 10). The fitness function is:

$$1 - \sqrt{\frac{\sum_i \sum_j [\mathrm{XNOR}(\mathtt{In1}(i), \mathtt{In2}(j)) - \mathtt{Out}(i, j)]^2}{4}}. \quad (2)$$

The sum reflects the "rms error between the output response of the circuit obtained by evolution and the ideal output signal of a XNOR logic function calculated for 4 logic states of the input: (0, 0) (1, 0), (0, 1), (1, 1)" as illustrated in Fig. 10. i, j point to the current values of In1, In2 respectively. $\mathtt{Out}(i, j)$ is the circuit output for a particular input configuration $\mathtt{In1}(i)$, $\mathtt{In2}(j)$; the target value is the XNOR function applied to $\mathtt{In1}(i)$, $\mathtt{In2}(j)$.

Two sets of experiments, population based and fitness based, are described. The experiments used a GA with parameters:

chromosome size 62,
population of 200,
tournament selection of size 10;
uniform mutation probability: 0.04,
uniform cross-over probability: 0.7,
elite strategy: 10%.

1) Population Based Experiment: After 60 generations, the GA obtained the circuit with the correct XNOR response. Fig. 11 shows the circuit configuration of the best individual, and indicates that the best configuration uses only the output probe which is connected only to the left FPTA.

To evaluate the FT of the best circuit configuration found after 60 generations, this individual was tested against 6 different faults chosen:

fault 0, fault 3, fault 5 are open switches,
fault 1, fault 2, fault 4 are closed switches.

We also tested the individual against inverse faults defined as:

inverse faults 3, 5 are closed switches;
inverse fault 2 is open switch.

The performance of the best individual was analyzed for these faults as shown in Fig. 12 and in Table IV (column 1). Also investigated was which individuals in the same population (mutants) could display the best response to each fault as shown in Fig. 13 and in Table IV (column 2).

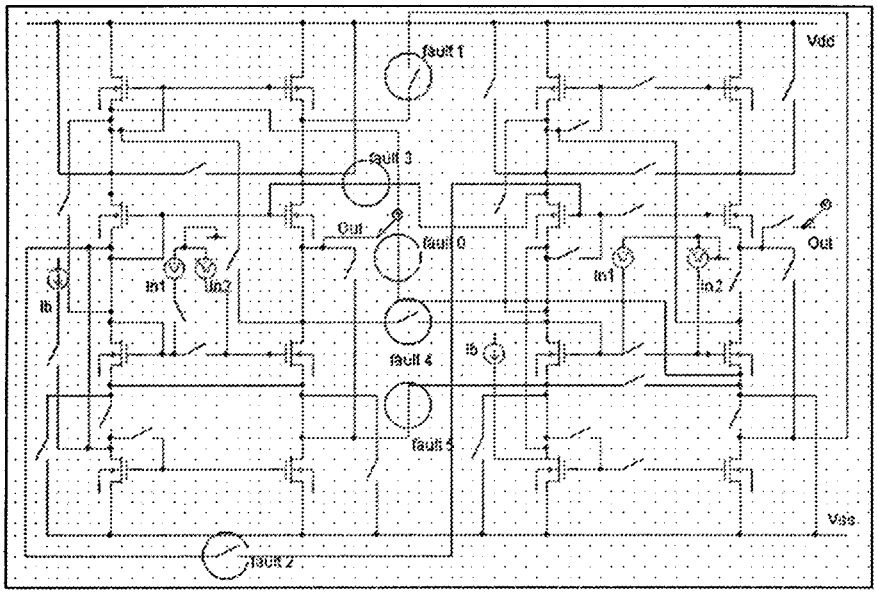

Fig. 11. Circuit design of best digital XNOR obtained at generation 60. (Population-based experiment).

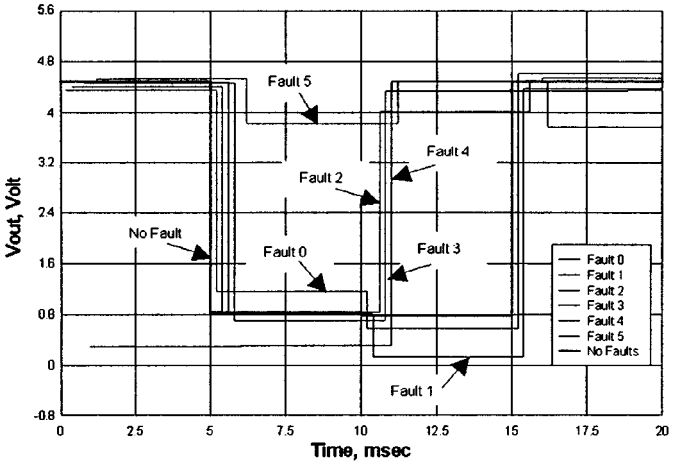

Fig. 12. Response of best digital XNOR circuit with no-fault and 6-faults applied. (Population-based). The best circuit design could not realize the XNOR functionality for faults 2, and faults 3, 4, 5. (The responses are shifted in time in all the figures to enhance the illustration.)

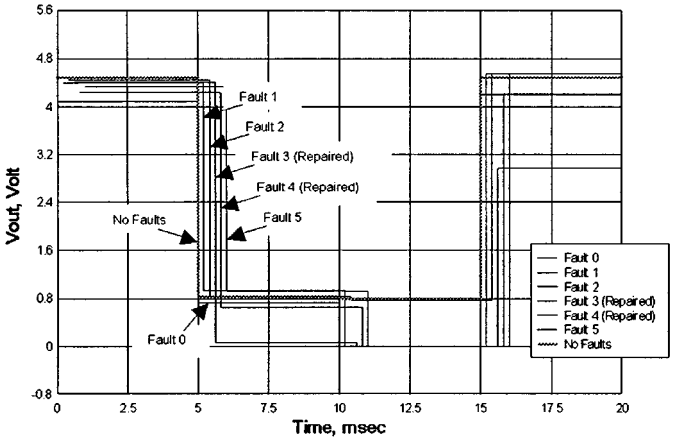

Fig. 13. Response of best mutants with respect to each fault for XNOR. (Population-based). Further evolution was needed to find an XNOR circuit for faults 3, 4.

TABLE IV
FITNESS ACHIEVED BY THE BEST INDIVIDUALS, MUTANTS, SELF-REPAIRED INDIVIDUALS (FOR THE POPULATION-BASED FT EXPERIMENT FOR THE DIGITAL XNOR) COLUMN 5 SHOWS THE NUMBER OF CIRCUIT EVALUATIONS FOR EACH TEST. THE BEST FITNESS IS IN BOLD. THE NUMBER OF EVALUATIONS IS GIVEN BY THE POPULATION SIZE (200) TIMES THE NUMBER OF GENERATIONS

	Best Individuals	Best Mutants	Self Repaired	Number of Evaluations
No fault	**0.8641**	0.8640		12000
Fault 0	0.8492	**0.8581**		200
Fault 1	**0.8814**	0.8805		0
Fault 2	0.5824	**0.8661**		200
Fault 3	0.5529	0.7687	**0.7910**	6000
Fault 4	0.3473	0.4905	**0.8731**	6000
Fault 5	0.3948	**0.8779**		200
Inv-F 2	**0.8641**	0.8641		0
Inv-F 3	**0.8641**	0.8641		0
Inv-F 5	**0.8641**	0.8641		0
Total				24800

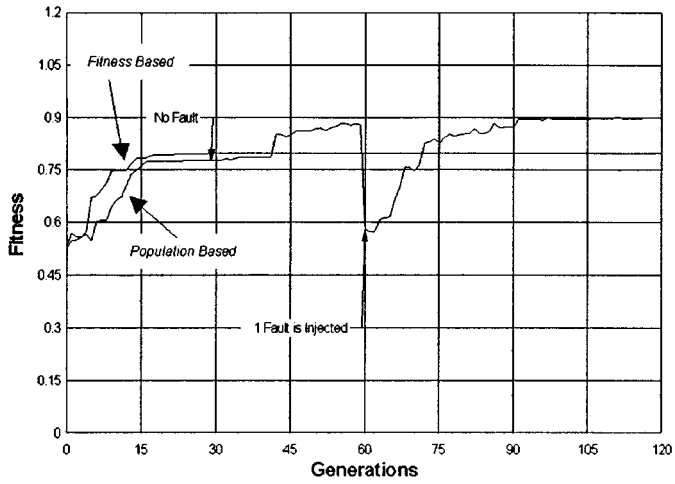

Fig. 14. Fitness of the best digital XNOR circuit through generation for population-based and fitness-based.

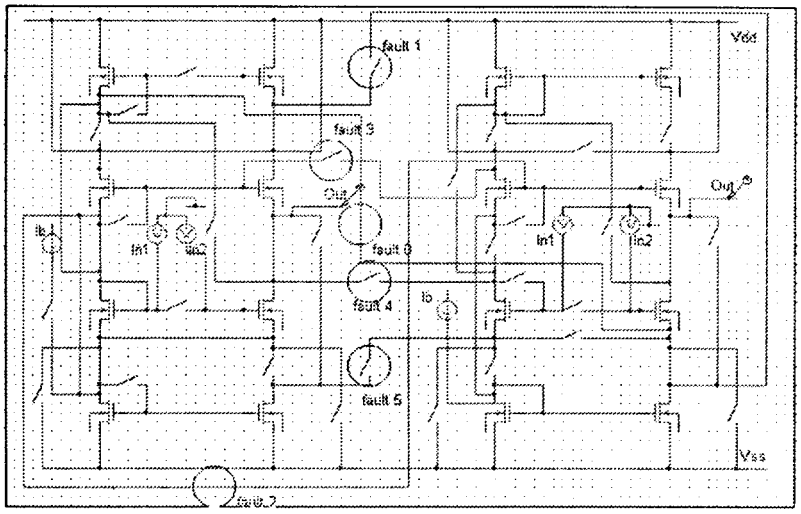

Fig. 15. Circuit design of the best digital XNOR circuit obtained at generation 60. (Fitness-based experiment).

Fig. 12 shows that the best circuit configuration does not achieve the XNOR functionality for faults 2, 3, 4, 5. Looking at the population in generation 60, some mutants had better responses for faults 2, 5 as shown in Fig. 13 and Table IV. However no mutants were found with acceptable performance for faults 3, 4 (Fig. 13 and Table IV). As in Section V-A-1 with the analog multiplier, the GA with the population of its last run was restarted, evaluating the individuals under fault 3 and fault 4 conditions. The GA was able to increase the performance with these faults and to off-line self-repair the circuit. The behavior of the faulty part became just another component to be used: the EA did not "know" that the part was supposed to do something else [13]. For fault 3, and starting with the last available population, it took half the number of generations (30 generations) to recover than when starting with a random population, as shown on Fig. 14. These experiments illustrate that the population effect can find instantaneously and on-line a circuit configuration to resolve the faults.

2) Fitness-Based Experiment: The chromosomes are evaluated in 4 circuit states: 1 without fault, and 3 with fault 2, 3, 5, respectively. The fitness of the chromosome is the average of the four evaluations. After 60 generations the GA obtained the circuit that best satisfied the requirement (Fig. 14). Fig. 15 shows the circuit configuration, and indicates that the input signals and the output probe are connected to both FPTA.

To evaluate the fault-tolerance of the best circuit-configuration found after 60 generations, the individual was tested against the 6 different faults defined in Section V-B-1 As anticipated, Fig. 16 and Table V show that the best circuit-configuration achieves the XNOR functionality for each of the 3 faults. The circuit-schematic shows that the switch-states are identical to the faults which make the circuit insensitive to the faults: fault 2 closed and faults 3, 5 opened (Fig. 11). Table V and Fig. 16 show that the circuit can also achieve the XNOR functionality with faults not included into the fitness function such as faults 0, 1, 4, but with lower performance: the circuit is robust to faults not encountered during evolution. However the circuit performance degrades when inverse faults were applied

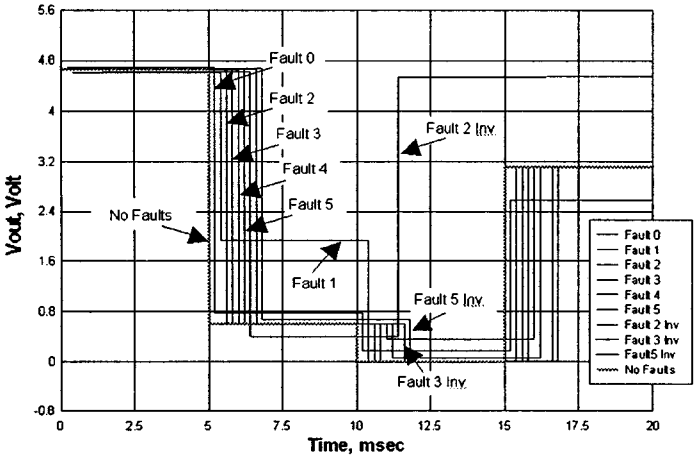

Fig. 16. Response of the best digital XNOR circuit with no fault and 5 faults applied. (Fitness-based).

TABLE V
FITNESS ACHIEVED BY THE BEST INDIVIDUALS AND MUTANTD FOR THE FITNESS-BASED FT EXPERIMENT FOR THE DIGITAL XNOR

	Best Individuals	Best Mutants	Number of Evaluations
No fault	**0.7993**	0.7993	42000
Fault 0	0.7439	**0.7790**	200
Fault 1	0.72725	**0.7996**	200
Fault 2 (inj)	0.7993	**0.7964**	200
Fault 3 (inj)	**0.7927**	0.79927	0
Fault 4	0.7953	**0.7973**	200
Fault 5 (inj)	**0.8002**	0.7987	0
Inv-F 2	0.5408	**0.6951**	200
Inv-F 3	0.7999	**0.7997**	200
Inv-F 5	0.7974	**0.7995**	200
Total			43400

because its design is specialized to the faults injected during the evolution. Looking at the population at generation 60, mutants with better responses were found for the inverse faults (Fig. 17 and Table V).

Table VI compares the results for population and fitness-based methods; it shows that the population-based

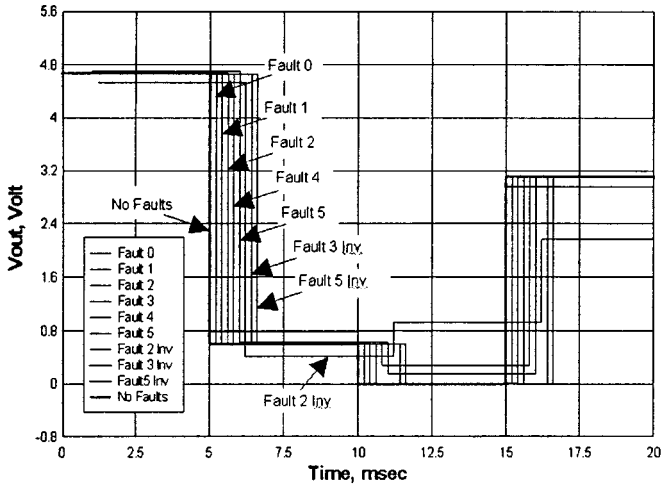

Fig. 17. Response of the best mutants with respect to each fault for the XNOR. (Fitness-based).

TABLE VI
COMPARISON OF POPULATION AND FITNESS-BASED METHODS FOR DIGITAL XNOR (POP-B⇒POPULATION-BASED; FIT-B⇒FITNESS-BASED)

	Best Pop-B	Best Fit-B	Performance
No fault	0.8641	0.7993	8%
Fault 0	0.8581	0.7790	10%
Fault 1	0.8814	0.7996	10%
Fault 2 (inj)	0.8661	0.7964	8%
Fault 3 (inj)	0.7910	0.7927	-0.2%
Fault 4	0.8731	0.7973	9%
Fault 5 (inj)	0.8779	0.8002	9%
Inv-F 2	0.8641	0.6951	24%
Inv-F 3	0.8641	0.7997	8%
Inv-F 5	0.8641	0.7995	8%
Avg Perf	0.8601	0.78588	9.5%

Pop-B ⇒ Population-Based

Fit-B ⇒ Fitness-Based

method achieved much better results for 9 out of the 10 faults. On average, the results for the population-based method are 9.5% superior to the fitness-based method. Using the number of individuals evaluated, the population-based method evaluated (Tables IV and V) 57.1% less individuals than the fitness-based method.

A second GA experiment was conducted for both the population-based and fitness-based methods; its results were similar to the ones in this Section V-B-2.

VI. LESSONS LEARNED

Two approaches for designing fault-tolerant FPTA were experimentally compared. The population-evolution approach has appreciably more advantages than the fitness-evolution approach:

1) The population approach builds circuits with better performance in a no-fault situation than does the fitness fault-tolerant approach. In the latter case, the evolution is constrained by the faults imposed on the circuit. But the fitness-based fault-tolerant approach has the advantage of obtaining a single circuit robust to multiple faults.
2) The population approach offers an on-line self-repair mechanism able to find circuits in the population with better performance than the circuits obtained by the fitness approach. Although the best circuit configuration for a faulty situation is not robust, the population contains mutant configurations able to achieve the desired functionality with the faulty circuit. They even display a better performance than the best configuration and mutants obtained by the fitness approach.
3) Although the population approach offers a self-repair mechanism, it must be done off-line in 20% to 50% less time needed to obtain a solution from scratch.
4) The population approach requires 20% to 50% less computation than the fitness approach. The speed-up increases considerably if multiple faults are injected simultaneously.

These experiments open the way for further investigation of the property of fault-tolerant evolutionary techniques applied to electronics, such as the behavior of the fault-tolerant system when arbitrary and many faults are injected, and the unavailability time is limited, or when the faults do not have the same phenotypic effect as genetic operators. New methodologies can be conceived by combining the population and the fitness approaches, or by including, more explicitly, redundancy in the system such as explored in the 'embryological' development [11].

These initial experiments, while illustrating the power of GA to design digital circuits and to maintain functionality by recovering from faults, only prepare the ground for further questions related to its application. Examples of further questions include:

- addressing how the evolutionary mechanism can be protected such that its implementation is not itself subject to faults,
- how should the fitness function be computed/stored.

ACKNOWLEDGMENT

The authors would like to thank A. Thakoor, T. Daud, and R. Tawel of Jet Propulsion Laboratory, for their visionary thinking on evolvable systems, their valuable discussions, and their encouragement and support.

REFERENCES

[1] J. Koza, F. H. Bennett, D. Andre, and M. A. Keane, "Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming," in *Proc. Genetic Programming Conf.*, 1996, pp. 28–31.
[2] J. Lohn and S. P. Colombano, "Automated analog circuit synthesis using a linear representation," in *Evolvable Systems: From Biology to Hardware*. ser. Lecture Notes in Computer Science, M. Sipper, D. Mange, and A. Perez-Uribe, Eds: Springer-Verlag, 1998, pp. 125–133.
[3] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined in physics," *Evolable Systems: From Biology to Hardware*, ser. Lecture Notes in Computer Science, pp. 390–405, 1996.
[4] E. Sanchez and M. Tomassini, Eds., *Towards Evolvable Hardware*: Springer-Verlag, 1996.
[5] T. Higuchi, M. Iwata, and W. Liu, Eds., "Evolvable systems: From biology to hardware," in *ICES 96*. ser. Lecture Notes in Computer Science: Springer-Verlag, 1996.
[6] M. Sipper, D. Mange, and A. Perez-Uribe, Eds., "Evolvable systems: From biology to hardware," in *ICES 98*. ser. Lecture Notes in Computer Science: Springer-Verlag, 1998.
[7] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, *Genetic Programming III—Darwinian Invention and Problem Solving*: Morgan Kaufman, 1999.

[8] E. Vitoz, "Analog VLSI processing: Why, where and how," in *J. VLSI Processing*: Kluwer, 1993.
[9] A. Stoica, "Toward evolvable hardware chips: Experiments with a programmable transistor array," in *7th Int. Conf. Microelectronics for Neural, Fuzzy, and Bio-Inspired Systems*: IEEE Computer Society Press, 1999.
[10] P. Layzell, "A new research tool for intrinsic hardware evolution," in *Proc. ICES'98*, ser. Springer-Verlag Lecture Note in Computer Science, 1998, pp. 47–56.
[11] P. Marchal *et al.*, "Embryological development on silicon," in *Artificial Life IV*, R. Brooks and P. Maes, Eds: MIT Press, 1994, pp. 365–366.
[12] R. White and F. Miles, "Principles of fault tolerance," in *Proc. Eleventh Annual Applied Power Electronic Conf. & Expo.*: IEEE Press, 1996, vol. 1, pp. 18–25.
[13] A. Thompson, "Evolving fault tolerant systems," in *Proc. First Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications*: IEEE Press, 1995, pp. 524–529.
[14] P. Layzell, "Inherent qualities of circuits designed by artificial evolution: A preliminary study of populational fault tolerance," in *Proc. First NASA/DoD Workshop on Evolvable Hardware*: IEEE Computer Society Press, 1999, pp. 85–86.
[15] A. Stoica, D. Keymeulen, and R. Tawel *et al.*, "Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital CMOS circuits," in *Proc. First NASA/DoD Workshop on Evolvable Hardware*: IEEE Computer Society Press, 1999, pp. 76–84.
[16] R. Zebulum *et al.*, "Analog circuits evolution in extrinsic and intrinsic modes," in *Proc. ICES 98*, ser. Springer-Verlag Lecture Notes in Computer Science, 1998, pp. 154–165.
[17] G. Devarayanadurg *et al.*, "Test set selection from structural faults in analog IC's," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 1026–1039, Jul. 1999.
[18] S. Niranjan and J. F. Frenzel, "A comparison of fault-tolerant state machine architecture for space-borne electronics," *IEEE Trans. Reliability*, vol. 45, no. 1, pp. 109–113, Mar. 1996.
[19] D. Keymeulen, M. Iwata, Y. Kuniyoshi, and T. Higuchi, "On-line evolution for a self-adapting robotic navigation system using evolvable hardware," in *Artificial Life*, ser. Special Issue on Evolutionary Robotics: MIT Press, 1999, vol. 4, pp. 359–393.
[20] C. Ortega and A. Tyrrell, "Reliability analysis of self-repairing bio-inspired cellular hardware," in *Proc. IEEE Half-day Colloquium on Evolutionary Hardware Systems*: IEEE Press, 1999, pp. 2/1–2/5.
[21] R. Zebulum *et al.*, "Evolvable hardware: Automatic synthesis of analog control systems," in *Proc. IEEE Aerospace Conf.*: IEEE Press, 2000.
[22] T. Higuchi *et al.*, "Real-world applications of analog and digital evolvable hardware," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 3, pp. 220–235, Sep. 1999.
[23] M. Murakawa, S. Yoshizawa, and I. Kajitani *et al.*, "The GRD chip: Genetic reconfiguration of DSP's for neural network processing," *IEEE Trans. Computers*, vol. 48, no. 6, pp. 628–639, 1999.
[24] I. Kajitani, T. Hoshino, and N. Kajihara *et al.*, "An evolvable hardware chip and its application as a multi-function prosthetic hand controller," in *Proc. 16th Nat. Conf. Artificial Intelligence (AAAI-99)*: AAAI Press, 1999, pp. 182–187.
[25] F. M. Goncalves, I. C. Teixeira, and J. P. Teixeira, "Integrated approach for circuit and fault extraction of VLSI circuits," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*: IEEE Press, 1999.
[26] L. Kuen-Jong, T. Jing-Jou, and H. Tsung-Chu, "BIFEST: A built-in intermediate fault effect sensing and test generation system for CMOS bridging faults," in *ACM Trans. Design Automation of Electronic Systems*: ACM Press, Apr. 1999, vol. 4, pp. 194–218.
[27] A. Avizienis, "Toward systematic design of fault-tolerant systems," *IEEE Computer*, vol. 30, no. 4, pp. 51–58, Apr. 1997.
[28] M. Melanie, *An Introduction to Genetic Algorithms*: MIT Press, 1996.
[29] D. Levine. PGAPack parallel genetic algorithm. [Online]. Available: http://www-unix.mcs.anl.gov/levine/PGAPACK/
[30] T. Higuchi, T. Niwa, and T. Tanaka *et al.*, "Evolving hardware with genetic learning: A first step toward building a Darwin machine," in *Animals to Animats 2: Proc. Second Int. Conf. Simulation of Adaptive Behavior*, J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, Eds: MIT Press, 1993, pp. 417–424.
[31] H. Hemmi, T. Hikage, and K. Shimohara, "AdAM: A hardware evolutionary system," in *Proc. First IEEE Conf. Evolutionary Computation*: IEEE Press, 1994, vol. I, pp. 193–196.
[32] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Second, 1992; MIT Press ed: University of Michigan Press, 1975.
[33] T. Quarles, A. R. Newton, D. O. Pederson, and V. A. Sangiovanni, *SPICE 3, Ver. 3F5, User's Manual*. Berkeley: Department of Electrical Engineering and Computer Science, Univ. of California, Mar. 1994.
[34] A. Stoica, D. Keymeulen, and J. Lohn, Eds., *Proc. First NASA/DoD Workshop on Evolvable Hardware*: IEEE Computer Society Press, 1999.

Didier Keymeulen received his Ph.D. in electrical engineering and computer science from the Free University of Brussels. Before joining JPL in 1998, he worked at the National Electrotechnical Laboratory in Japan on the applications of evolvable hardware for robotics. At JPL, he is responsible for the applications of evolvable hardware to fault-tolerant electronics and adaptive sensor technology. His interests include adaptive hardware for embedded systems.

Ricardo S. Zebulum is a Post-Doc Scholar at JPL. He received his B.S. (1992) in electronic engineering, M.Sc. (1995) in electrical engineering, and Ph.D. (1999) in electrical engineering, at the Catholic University of Rio, Brazil. He was at Sussex University, 1997–1999, as a Visiting Ph.D. Student. He has been involved with research in Evolvable Hardware since 1996. His research interests also include fault-tolerant systems, low power electronics, and analog VLSI design.

Yili Jin received her M.S. in engineering (Research) from Victoria University of Technology in Melbourne, Australia. In her Master's thesis she introduced new 3D computational models for heat and pressure distribution inside enclosures functioning in environments with large temperature variations. She works for Caltech in the Engineering and Estimation Department. She does joint research with JPL, performing modeling and simulation experiments for extending the life of electronics operating at extreme temperatures and under intensive radiation. She has published several papers in this area. She is a member of ASME and IEEE.

Adrian Stoica received his Ph.D. in electrical engineering and computer science from Victoria University of Technology, Melbourne, Australia. He is the Principal Investigator for a DARPA project on evolvable hardware for adaptive computing. His interests include adaptive and learning hardware for autonomous systems. He received the Lew Allen Award for his research in evolvable hardware.