Evolution of Polymorphic Self-Checking Circuits

Lukas Sekanina

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
sekanina@fit.vutbr.cz

Abstract. This paper presents elementary circuit components which exhibit self-checking properties; however, which do not utilize any additional signals to indicate the fault. The fault is indicated by generating specific values at some of standard outputs of a given circuit. In particular, various evolved adders containing conventional as well as polymorphic gates are proposed with less than duplication overhead which are able to detect a reasonable number of stuck-at-faults by oscillations at the carry-out output when the control signal of polymorphic gates oscillates.

1 Introduction

The use of self-checking circuits is important to various applications nowadays. As it is assumed that fault-tolerance issues will be more important in the era of nanoelectronics, the use of this type of circuits will certainly grow. Self-checking circuits are conventionally constructed by adding checking logic around an unmodified original output of the circuit. Garvie utilized the evolutionary algorithm to design small self-checking circuits which merge the original function with the checker logic [1]. He reported better results than conventional approaches if the total area of the circuit is measured. However, conventional as well as evolutionary approaches require special signals which indicate that a fault is present in the circuit. When more complex circuits are composed of these elementary self-checking circuits, these special signals have to be interconnected and aggregated to obtain the global information about the fault.

The goal of this work is to propose elementary circuit components which exhibit self-checking properties; however, which do not utilize any additional signals to indicate the fault. The fault is indicated by generating specific values at some of standard outputs of a given circuit. This research is motivated by the fact that some of future computing architectures will probably contain a large massively parallel array of locally interacting computing elements in which the issue of efficient wiring will be very important. The existence of some extra signals or even global signals will be problematic [2]. In order to reduce the overall cost and wiring, new circuits are proposed in which the user function (e.g. addition) is completely merged with the test procedure. Furthermore, as the circuits indicate the faulty behavior by oscillations at one of their output signals, no additional testing input/output signal wires are needed. This paper

describes and analyzes some evolved adders which exhibit this property. These circuits utilize conventional as well as polymorphic gates.

Polymorphic gates perform two or more logic functions which are activated under certain conditions by changing control parameters (such as temperature, Vdd, light, external control voltage etc.) of the circuit [3, 4]. Similarly to [5], instead of two functions, the proposed circuits perform only one function in the both modes of polymorphic gates. These circuits are designed in such a way that if a fault is present in the circuit, one of its outputs oscillates when the control signal of polymorphic gates oscillates. Thus, the fault can be detected.

The plan for this paper is as follows: Section 2 summarizes basic principles of digital circuit testing. Section 3 introduces the concept of polymorphic electronics and examples of existing polymorphic gates and circuits. In Section 4 and 5 the proposed method is described which allows obtaining self-checking circuits with a reasonable overhead and without any supporting diagnostic wires. While evolved circuits are presented in Section 6, Section 7 discusses them. Conclusions are given in Section 8.

2 Self-Checking Circuits

A built-in self-test (BIST) mechanism within an integrated circuit is a function which verifies all or a portion of the internal functionality of the integrated circuit [6]. Typically, a pseudo-random sequence generator produces the input signals for a section of combination circuitry and a signature analyzer observes the output signals and produces a syndrome. The syndrome is compared with the correct syndrome to determine whether the circuit is performing according to a specification. BIST techniques can be classified as concurrent and non-concurrent. Concurrent BIST uses embedded checkers for on-line testing during normal operation. Non-concurrent BIST autonomously performs off-line testing of the device in which is built in, before normal operation.

Circuits with Concurrent Error Detection (CED) are capable of detecting transient and permanent faults and are widely used in systems where dependability and data integrity are important [7]. CED techniques can also enhance off-line testability and reduce BIST overhead. Their importance grows as shrinking process technology makes logic sensitive to radiation hazards such as Single Event Upsets. Almost all CED techniques decompose circuits into two modules: the functional logic and the checker [8, 9]. The functional logic provides output encoded with an error detecting code and the checker determines if the output is a codeword. The checker itself traditionally provides a two-rail signal in which an error is indicated by both rails being at the same value. A classic CED technique is duplication in which the output function generator and check symbol generator are functionally identical and the checker simply compares their output.

One well-defined class of self-checking circuits, called totally self-checking (TSC), offers two desirable properties in the presence of faults from the set of likely faults F [9]: (1) it is fault-secure (FS) with respect to F, i.e., no fault from

F can cause undetected error for any input codeword; and (2) it is self-checking with respect to F, i.e., if for every fault f from F, a circuit produces a noncode space output for at least one code space input. In a TSC circuit, the occurrence of the first error could be immediately signaled by activating the error signal.

Automatic synthesis methods for TSC circuits with less than duplication overhead have been proposed (see, e.g., [10, 11, 12]). Garvie has presented a method based on evolutionary design capable of adding logic around an unmodified original output generating function circuit to make it TSC with less than duplication overhead. Moreover, he proposed a method capable of generating TSC circuits with unconstrained structure, i.e. circuits not strictly adhering to the function-checker modular decomposition [1].

3 Polymorphic Circuits

The concept of polymorphic electronics was proposed by Stoica et al [3]. In fact, polymorphic circuits are multifunctional circuits. The change of their behavior comes from modifications in the characteristics of components (e.g. in the transistor's operation point) involved in the circuit in response to controls such as temperature, power supply voltage, light, etc. [4, 3]. Table 1 gives examples of the polymorphic gates reported in literature. For instance, the NAND/NOR gate is the most famous example [13]. This circuit consists of six transistors and operates as NAND when Vdd is 3.3V and as NOR when Vdd is 1.8V. The circuit was fabricated in a 0.5-micron CMOS technology. The circuit is stable for ±10% variations of Vdd and for temperatures in the range of –20°C to 200°C.

Table 1. Examples of existing polymorphic gates and their implementation cost

Gate	control values	control	transistors	ref.
AND/OR	27/125°C	temperature	6	[3]
AND/OR/XOR	3.3/0.0/1.5V	external voltage	10	[3]
AND/OR	3.3/0.0V	external voltage	6	[3]
AND/OR	1.2/3.3V	Vdd	8	[4]
NAND/NOR	3.3/1.8V	Vdd	6	[13]
NAND/NOR/NXOR/AND	0/0.9/1.1/1.8V	etxternal voltage	11	[14]

The use of polymorphic gates as building blocks offers an opportunity to design multifunctional digital modules at the gate level. Once the circuit is designed at the gate level (abstracting thus from the electric level), it does not matter whether this circuit is "reconfigured" by a level of Vdd, temperature or light. For example, using the polymorphic NAND/NOR gate and some standard gates is possible to create a circuit which operates as a three-bit multiplier in the first environment and as a six-input sorter in the second environment [15].

4 Proposed Method

Similarly to [5], the goal is to propose such a polymorphic circuit whose output Y_{osc} will oscillate when a fault is present in the circuit and simultaneously the polymorphic mode is changed. In other words, it is required that the polymorphic circuit performs the same function in each mode of its polymorphic gates. Note that Y_{osc} is not a special testing output signal; Y_{osc} is one of functional circuit outputs.

4.1 Polymorphic Self-Checking

In contrast to [5], presented circuits utilize conventional as well as polymorphic gates. Consider that only one type of polymorphic gates – the NAND/NOR gate (controlled by Vdd or by some external voltage Vs) – is utilized. Let Vc denote this control voltage, independently of the fact that it might be Vdd or Vs. The NAND/NOR gates can be switched simultaneously between NAND and NOR function by changing the control value Vc. If the inputs were not changed and the control signal were switched at the frequency kf (k is a positive integer and f is an operational frequency of the circuit) then the circuit outputs would remain steady (neglecting here some switching spikes), which is a normal operation of the circuit.

However, it is required that the circuit operates in such a way that if a stuck-at-fault is present within the circuit then Y_{osc} output will oscillate between 0 and 1 at the same frequency as the control signal oscillates between its NAND and NOR levels. In addition to its primary function, this output works as the indicator of a stuck-at-fault in the circuit. The control signal is activated for polymorphic gates whenever the circuit should be tested. The test can be performed either before the system is put into operation or in some time slots devoted to testing of the system. If it is unproblematic with respect to the system function the proposed scheme allows testing the system permanently, during circuit operation. Common input values of the circuit are then considered as test vectors. In this mode, the control signal has to oscillate permanently.

4.2 Analysis of Self-Checking Capabilities

In order to investigate basic self-checking properties of proposed circuits, we will analyze these circuits at the gate level. The following procedure is applied for two types of faults (stuck-at-0 and stuck-at-1) injected to the outputs of all K gates within the circuit.

1. For $i = 1$ to K do
2. begin
   (a) Inject a stuck-at-fault at the output of gate i.
   (b) Set polymorphic gates to mode 1.
   (c) Calculate R_1 – the vector of Y_{osc} values for all possible combinations of input values.

(d) Set polymorphic gates to mode 2.
(e) Calculate R_2 – the vector of Y_{osc} values for all possible combinations of input values.
(f) Calculate M_{g_i} – the set of input vectors that have induced **different** Y_{osc} values for gate i.

end

This algorithm allows us to calculate the fault coverage for all possible test vectors (i.e. for the trivial test). In addition, we are able to find those input vectors which induce oscillations at the Y_{osc} output for the particular stuck-at-fault. It is unrealistic to expect that the oscillations will always appear when only a single test vector is applied. In most cases we have to apply several test vectors in order to indicate a stuck-at-fault by oscillations at Y_{osc}. The goal is to find such a minimal subset, M_{min}, of all possible input vectors which, when applied simultaneously with changing Vc, will cause oscillations at Y_{osc} in the case that a stuck-at-fault is present in the circuit. However, the approach has one disadvantage: when a stuck-at-fault is present at Y_{osc} then no oscillations are observable and the fault has to be detected in another way. It can also happen for some gates that when a stuck-at-fault is present at these gates then no oscillations are detectable at Y_{osc} for any input vector. The goal is to avoid the use of such gates and subcircuits in the circuit because they decrease the efficiency of the approach.

4.3 Test Problem: Full Adder

The approach will be tested on a standard 1-bit full adder which has two operands, A and B, and input carry, C_{in}. It generates the sum $S = A \oplus B \oplus C_{in}$ and the output carry $C_{out} = AB + BC_{in} + AC_{in}$. A standard optimized static CMOS VLSI implementation of the 1-bit full adder costs 24 transistors [16].

In order to construct adders with specific properties, standard 1-bit full adders are usually equipped with several additional input/output signals. For example, Carry-Look-Ahead adders use 1-bit full adders that generate two additional signals – propagate, $P = A \oplus B$, and generate, $G = AB$ utilized to look ahead to predict the carry-out [17]. Self-checking adders contain the parity prediction logic [10, 12].

In our case, no additional signals have to be utilized to indicate a stuck-at-fault. Either S or C_{out} will perform this task. As Section 6 will show, it is useful to utilize C_{out} (i.e., $Y_{osc} = C_{out}$) because the oscillations can propagate through next adders connected in the carry propagate chain. Thus, it could be possible to create n-bit self-checking adder which indicates a stuck-at-fault by oscillations at its most significant C_{out}.

A self-checking adder consisting of fourteen NAND/NOR gates was proposed in [5] (denoted as FA-14 in Table 2). The authors conjectured that polymorphic circuits performing the identical function in the both modes would be more fault-tolerant to radiation induced faults as their function could be restored by changing the mode of operation. It seems that the adder was not designed with

the aim of detecting stuck-at-faults by oscillations at C_{out}. We analyzed this circuit using the proposed method and recognized that the stuck-at-fault is not recognized for nine gates (when oscillations are measured at C_{out}, i.e. at gate 13) and for six gates (when oscillations are measured at S, i.e. at gate 14). Therefore, this adder is not suitable for our purposes. Moreover, its implementation cost is relatively high (14 x 6 = 84 transistors).

5 Evolutionary Design of Polymorphic Circuits

In order to evolve gate-level polymorphic circuits which perform the identical function in the both modes, we will use an evolutionary algorithm (EA) inspired by Cartesian Genetic Programming [18, 15].

A candidate digital circuit is represented using a two-dimensional array of $(n_r \times n_c)$ programmable nodes. Each programmable node has two inputs, a single output and can be programmed to implement one of functions given in function set Γ. The role of EA is to find the interconnection of nodes and functions performed by the nodes for a given specification expressed by means of a truth table. A candidate circuit is encoded as an arrays of integers of the size $3.n_r.n_c + n_o$, where n_o is the number of circuit outputs. As only combinational circuits are evolved, no feedback links are allowed in candidate circuits. Hence the node input can be connected either to an output of a node placed in some of preceding columns or to a primary circuit input.

The EA uses one genetic operator – mutation – which modifies m integers of the chromosome. Either a node or an output connection is modified. The EA operates with the population of λ individuals. The initial population is randomly generated. Every new population consists of a parent (the fittest individual from the previous population) and its mutants. In case that two or more individuals have received the same fitness score in the previous generation, the individual which did not serve as the parent in the previous population will be selected as a new parent. The fitness value is defined as follows:

$$fitness = B_1 + B_2 + (W - z) \tag{1}$$

where B_1 (resp. B_2) is the number of correct output bits for the first (resp. second) polymorphic mode obtained as the response for all possible input combinations, z denotes the number of transistors utilized in a particular candidate circuit and W is a suitable constant ($W = 10.n_c.n_r$). The last term is considered only if the circuit behavior is perfect in the both modes; otherwise $W - z = 0$. The number of transistors is calculated for the nodes used in the phenotype as follows: the NAND/NOR costs 6 transistors [13], the XOR costs 8 transistors and inverter costs 2 transistors. We intentionally restricted Γ to contain only the gates NAND/NOR, XOR and NOT because these gates were recognized during experiments as most useful for our task.

6 Results

This section presents properties of selected self-checking circuits that we evolved using the following basic setup: population size is 15, one randomly selected gene is mutated and 100k generations are produced in each run. Other parameters are described in particular subsections. Table 2 summarizes some features of evolved circuits. The FA-14 is shown for comparison.

Table 2. Parameters of self-checking circuits. Last column lists the gates with unrecognizable faults when measured at C_{out} or S (see also Fig. 4). The overhead is calculated in relation to the number of transistors in optimized conventional solutions

Circuit	Gates	Trans.	Delay	Overhead	Unrecognizable faults at gates:
FA-14 [5]	14	84	5	250%	(3, 4, 5, 8, 9, 11, 12, 13, 14)-C_{out} (1, 2, 3, 10, 13, 14)-S
FA3n	6	36	5	50%	(6)-C_{out}
FA3 [19]	7	38	4	58%	(5, 7)-C_{out}
FA2	8	52	3	116%	(3, 6, 8)-C_{out}
FA1	9	54	6	125%	(6,7,8,9)-C_{out}, (7,9)-S
HA	4	28	3	100%	(4)-C_{out}
ANDFA1	10	64	7	113%	(5, 10)-C_{out}

6.1 Full Adders

Best-Evolved Adder (FA3n) In this experiment, we utilized 20 programmable nodes organized in array of 20 x 1 elements and Γ = {NAND/NOR, XOR, NOT}. Figure 1 shows the FA3n adder which exhibits the best self-checking properties out of all experiments. It consists of two XOR gates, an inverter and three polymorphic NAND/NOR gates. The sum calculation is quite standard. On the other hand, the carry output is calculated unconventionally using the three NAND/NOR gates that are connected to the circuit inputs as well as to inverted sum, $\overline{S}$. Independently of the level of the control signal of the polymorphic gates, this logic network always generates a correct carry-out signal when there is no fault present in the circuit.

The proposed adder would cost 36 transistors, i.e. the overhead is 50% in comparison with a conventional transistor-level implementation. The overhead is similar to conventionally designed self-checking circuits [20, 1] (for example, the self-checking adder reported in [11] also consists of 36 transistors); however, the proposed adder does not require any additional wires when Vdd controls the polymorphic gates. It is assumed that standard gates (the XORs and inverters) operate correctly for the both modes of polymorphic gates (for example, for Vdd = 1.8V as well as Vdd = 3.3V). Note that other adders reported in this paper exhibit the overhead more than 100%.

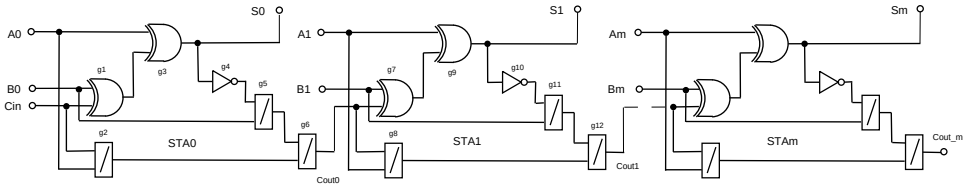

Fig. 1. FA3n – the best evolved 1-bit self-checking adder and its utilization in the carry-propagate adder

Figure 4 shows the fault coverage for all possible test vectors (i.e. for the trivial test). Test vectors are indexed 0–7 which corresponds with the circuit inputs ordered as (Cin, B, A). Symbol "x" means that a corresponding test vector is able to induce oscillations at the carry-out output for the particular stuck-at-fault. We can observe that the stuck-at-fault can not be detected only when injected to gate 6. The reason is that this gate is connected directly to the primary output of the adder. It is easy to derive from Fig. 4 that by applying test vectors $M_{min} = \{1, 2, 3, 5\}$ or $\{1, 2, 5, 6\}$ or $\{2, 4, 5, 6\}$, all single faults can be detected. In other words, at least four test vectors have to be applied in order to initiate oscillations at the carry-out output when a single fault is present in the adder. The probability of fault detection is 0.325 when only a single randomly generated test vector is applied.

The n-bit Self-Checking Adder As Fig 1 shows, by cascading 1-bit self-checking adders FA3n we can construct carry-propagate adders. Consider a 2-bit adder. When a fault is present in STA1 then the fault is indicated by oscillations at C_{out1} as explained for the 1-bit full adder in the previous paragraph. In order to detect at C_{out1} a fault which is present in STA0 then C_{out0} has to propagate through STA1. This can only be achieved by setting $A_1 \neq B_1$. Similarly to previous section, we can observe that the stuck-at-fault can not be detected when injected to gates 6 or 12.

Only four test vectors, for example $M_{min} = \{10, 14, 20, 21\}$, are needed to detect all single faults at remaining gates and thus to initiate the oscillations at C_{out1}. In general, four test vectors have to be used to perform this task for n-bit self-checking adder. For the 2-bit self-checking adder, the probability of fault detection in STA1 is 0.325 when only a single randomly generated test vector is applied. However, the probability of fault detection is only 0.1625 when a fault is present in STA0. Recall that $A_1 \neq B_1$ has to be ensured to propagate the oscillations. In general, this probability decreases twice with every next 1-bit adder which is closer to the least significant bit. Therefore, it is much easier to detect faults in more significant bits of the adder than those in less significant bits when only a single randomly chosen test vector is applied. However, this problem could be overcome by observing also internal carry-out signals.

Shortest-Delay Adder (FA2) In this experiment, we utilized $n_r = 8$, $n_c = 3$ and $\Gamma = \{\text{NAND/NOR, XOR, NOT}\}$. Figure 2 shows an evolved self-checking adder which exhibits the shortest delay. As Fig. 4 shows, some stuck-at-faults cannot be recognized by oscillations at C_{out}.

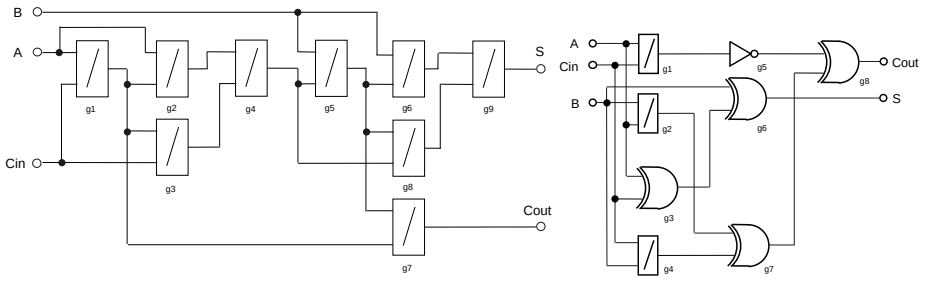

Fig. 2. FA1 – self-checking adder solely composed of NAND/NOR gates (left), FA2 – the shortest delay self-checking adder (right)

An Adder Composed of NAND/NOR Gates Only (FA1) In this experiment, we utilized $n_r = 2$, $n_c = 6$ and $\Gamma = \{\text{NAND/NOR}\}$. Figure 2 shows an evolved self-checking adder which consists of nine NAND/NOR gates. Its main feature is that it contains fewer gates than existing FA-14 adder. As many stuck-at-faults cannot be recognized by oscillations at C_{out} (see Fig. 4), this adder is not practically useful for creating carry-propagate adders. However, almost all stuck-at-faults are recognizable at the S output.

6.2 Half Adder (HA)

Figure 3 shows evolved self-checking half adder. It is implemented using 24 transistors which means the 100% overhead. As Fig. 4 shows, at least two test vectors have to be applied in order to detect all recognizable stuck-at-faults. In this experiment, we utilized 24 programmable nodes organized in array of 3 x 8 elements and $\Gamma = \{\text{NAND/NOR, XOR, NOT}\}$.

6.3 Extended adder (ANDFA1)

By the *extended adder* we mean a circuit of four inputs A, B, C and D which firstly calculates H = A AND B. Then, the sum S and the carry-out are calculated using H, C and D as inputs. This circuit serves as a building block of combinational multipliers [17]. Figure 3 shows evolved ANDFA1 adder which contains 10 gates (64 transistors). This represents the overhead of 113%. In this experiment, we utilized 28 programmable nodes organized in array of 7x4 elements and $\Gamma = \{\text{NAND/NOR, XOR, NOT}\}$.

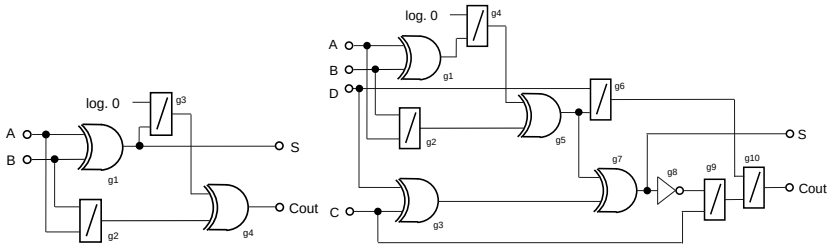

Fig. 3. HA – self-checking half adder (left), ANDFA1 – self-checking extended adder (right)

7 Discussion

All evolved circuits exhibit the required logic behavior in the both modes of polymorphic gates. Regarding testability properties, only FA3n and HA circuits can indicate all stuck-at-faults at the C_{out} output. However, the problem is that stuck-at-faults located directly at the C_{out} output cannot be principally eliminated within this scheme. This represents the main weakness of the proposed approach. As the area overhead of FA3 and HA circuits is 50% (100%, respectively), they are also good alternatives to the dual-module redundancy.

These circuits can be utilized as building blocks of larger adders and multipliers in which a stuck-at-fault can be indicated by changes at one of their outputs when the mode of polymorphic gates is changed. In the case of multipliers, more effort will be needed to find cheaper implementations of extended adders. Breaking the carry chain in any of these circuits represents a serious problem for the method. Another problem is related to the decreasing probability of detecting stuck-at-faults for less significant bits in carry-propagate adders.

Proposed adders show very promising behaviors especially when we consider that the requirement on testability was not included into the fitness function in this initial study. We just evolved polymorphic adders and then verified their testability. In our future research, we will include the requirement on testability directly to the fitness function.

The computational time for this problem is not high. For the 1-bit full adder we performed 370 experiments. On average, 11 thousands generations are produced in each experiment which takes approx. 0.25 sec on a standard PC equipped with Athlon64 X2 4800+ processor.

8 Conclusions

We presented low-cost implementations of self-checking adders which indicate a stuck-at-fault by oscillations at the carry-out output. When it is possible for a target application to switch the control signal of polymorphic gates (e.g. Vdd) either in some time slots or during the entire operation, the adder would check

itself. In fact, the adder can permanently be under test because its common input values can serve as test vectors. If these inputs values are diverse and updated very often then the probability of fault detection increases. Proposed circuits can be utilized as components for creating larger self-checking adders and multipliers which do not require any additional test signals.

Acknowledgements

This work was partially supported by the Grant Agency of the Czech Republic under contract No. 102/06/0599 *Methods of polymorphic digital circuit design* and the Research Plan No. MSM 0021630528 - *Security-Oriented Research in Information Technology.*

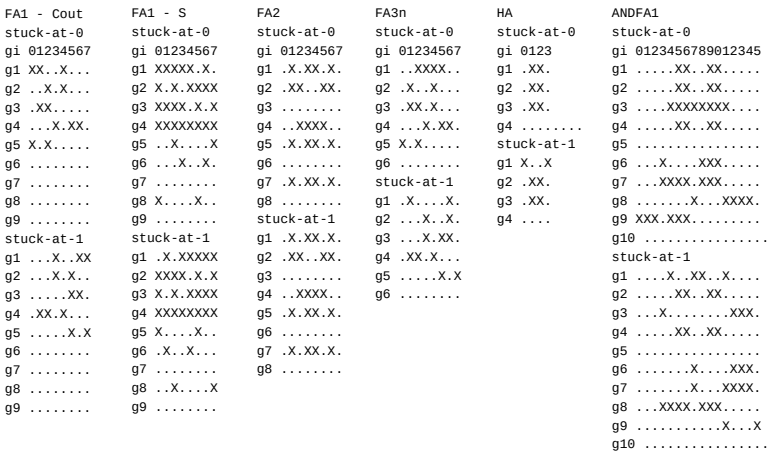

Fig. 4. Fault coverage for evolved circuits. Symbol "x" means that a corresponding test vector is able to induce oscillations at the output for the particular stuck-at-fault.

References

[1] Garvie, M.: Reliable Electronics through Artificial Evolution. PhD thesis, University of Sussex (2005)
[2] Frank, M.: Reversibility for Efficient Computing. PhD thesis, Massachusetts Institute of Technology (1999)

[3] Stoica, A., Zebulum, R.S., Keymeulen, D.: Polymorphic electronics. In: Proc. of Evolvable Systems: From Biology to Hardware Conference. Volume 2210 of LNCS., Springer (2001) 291–302
[4] Stoica, A., Zebulum, R.S., Keymeulen, D., Lohn, J.: On polymorphic circuits and their design using evolutionary algorithms. In: Proc. of IASTED International Conference on Applied Informatics AI2002, Insbruck, Austria (2002)
[5] Zebulum, R.S., Stoica, A.: Multifunctional Logic Gates for Built-In Self-Testing. NASA Tech Briefs **30**(3) (2006) 10
[6] Novak, O., Gramatova, E., Ubar, R.: Handbook of Testing Electronic Systems. Czech Technical University Publishing House (2005)
[7] Pradhan, D.K.: Fault-Tolerant Computer System Design. Prentice Hall (1996)
[8] Diaz, M., Azéma, P., Ayache, J.M.: Unified design of self-checking and fail-safe combinational circuits and sequential machines. IEEE Trans. Computers **28**(3) (1979) 276–281
[9] Piestrak, S.J.: Feasibility study of designing tsc sequential circuits with 100% fault coverage. In: 17th IEEE Int. Symposium on Defect and Fault-Tolerance in VLSI Systems, IEEE Computer Society (2002) 354–364
[10] Touba, N.A., McCluskey, E.J.: Logic synthesis of multilevel circuits with concurrent error detection. IEEE Trans. on CAD of Integrated Circuits and Systems **16**(7) (1997) 783–789
[11] Marienfeld, D., Ocheretnij, V., Gössel, M., Sogomonyan, E.S.: Partially duplicated code-disjoint carry-skip adder. In: Proc. of the 17th IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, IEEE (2002) 78–86
[12] Kakaroudas, A.P., Papadomanolakis, K., Kokkinos, V., Goutis, C.E.: Comparative study on self-checking carry-propagate adders in terms of area, power and performance. In: Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation, 10th Int. Workshop. Volume 1918 of LNCS., Springer (2000) 187–194
[13] Stoica, A., Zebulum, R., Guo, X., Keymeulen, D., Ferguson, I., Duong, V.: Taking Evolutionary Circuit Design From Experimentation to Implementation: Some Useful Techniques and a Silicon Demonstration. IEE Proc.-Comp. Digit. Tech. **151**(4) (2004) 295–300
[14] Zebulum, R.S., Stoica, A.: Four-Function Logic Gate Controlled by Analog Voltage. NASA Tech Briefs **30**(3) (2006) 8
[15] Sekanina, L., Starecek, L., Gajda, Z., Kotasek, Z.: Evolution of multifunctional combinational modules controlled by the power supply voltage. In: Proc. of the 1st NASA/ESA Conference on Adaptive Hardware and Systems, IEEE Computer Society (2006) 186–193
[16] Weste, N., Harris, D.: CMOS VLSI Design: A Circuits and Systems Perspective (3rd edition). Addison Wesley (2004)
[17] Wakerly, J.: Digital Design: Principles and Practices. Prentice-Hall (2000)
[18] Miller, J., Job, D., Vassilev, V.: Principles in the Evolutionary Design of Digital Circuits – Part I. Genetic Programming and Evolvable Machines **1**(1) (2000) 8–35
[19] Sekanina, L.: Design and Analysis of a New Self-Testing Adder Which Utilizes Polymorphic Gates. In: Proc. of the 10th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop, IEEE Computer Press (2007) 1–4
[20] Ocheretnij, V., Marienfeld, D., Sogomonyan, E.S., Gössel, M.: Self-checking code-disjoint carry-select adder with low area overhead by use of add1-circuits. In: 10th IEEE Int. On-Line Testing Symposium, IEEE (2004) 31–36