Design and Analysis of a New Self-Testing Adder Which Utilizes Polymorphic Gates

Lukas Sekanina
Faculty of Information Technology, Brno University of Technology
Bozetechova 2, 612 66 Brno, Czech Republic
Email: sekanina@fit.vutbr.cz

***Abstract*— This paper describes a new self-testing 1-bit full adder. This circuit consists of three polymorphic NAND/NOR gates, two XOR gates and two inverters. The adder is able to detect a reasonable number of stuck-at-faults without the need of any additional logic and diagnostic signals. A fault is indicated by oscillations at the carry-out output. Properties of n-bit carry-propagate adder which is composed of the proposed 1-bit self-testing adders are investigated.**

I. Introduction

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 [9]. Almost all CED techniques decompose circuits into two modules: the functional logic and the checker [1], [8]. 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.

Automatic synthesis methods for totally self-checking (TSC) with less than duplication overhead have been proposed (see, e.g., [14], [5], [3], [4]). Garvie has presented a method 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 [2].

In order to reduce the overall cost and wiring, a new adder is proposed in which the user function (addition) is completely merged with the test procedure. Furthermore, as the circuit indicates a faulty behavior by oscillations at the carry-out output, no additional input or output signals are utilized. The goal of this paper is to describe this unconventional design and analyze self-testing properties of the circuit. The adder utilizes conventional as well as polymorphic gates.

Polymorphic circuits exhibit one or more additional functions in addition to the "main" function of the circuit [12]. The additional functions can be activated under certain conditions by changing control parameters (such as temperature, Vdd, light, an external control voltage etc.) of the circuit. In the recent years, several polymorphic gates have been described [12], [13]. These gates were utilized to construct more complicated multifunctional circuits which perform two different functions (e.g. addition vs. multiplication) for two different levels of the control variable (e.g. for Vdd = 1.8V and Vdd = 3.3V) of the polymorphic gates [10]. Similarly to [17], instead of two functions, the proposed adder performs only one function – addition; however, by two different implementations which are switched using a control variable during circuit normal operation.

II. Polymorphic Circuits

The concept of polymorphic electronics was proposed by Stoica et al [12]. In fact, polymorphic circuits are multi-functional 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. [13], [12].

Table I gives examples of the polymorphic gates reported in literature. For instance, the NAND/NOR gate is the most famous example [11]. 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 $\pm 10\%$ variations of Vdd and for temperatures in the range of –20°C to 200°C.

Figure 1 demonstrates the behavior of a polymorphic gate-level circuit. 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

TABLE I

EXAMPLES OF EXISTING POLYMORPHIC GATES

Gate	control values	control	trans.	ref.
AND/OR	27/125°C	temperature	6	[12]
AND/OR/XOR	3.3/0.0/1.5V	ext. voltage	10	[12]
AND/OR	3.3/0.0V	ext. voltage	6	[12]
AND/OR	1.2/3.3V	Vdd	8	[13]
NAND/NOR	3.3/1.8V	Vdd	6	[11]
NAND/NOR/NXOR/AND	0/0.9/1.1/1.8V	ext. voltage	11	[16]

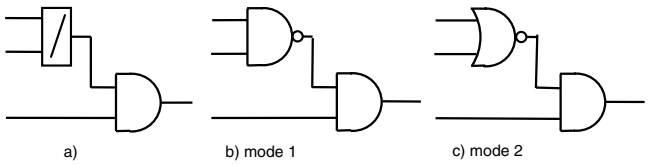

Fig. 1. (a) A polymorphic gate-level circuit which contains an AND gate and a polymorphic NAND/NOR gate, (b) its behavior in the first mode, (c) its behavior in the second mode

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 [10].

III. A NEW SELF-TESTING ADDER

This section describes a standard 1-bit full adder and introduces a new self-testing adder and its variants.

A. Standard Adders

A standard 1-bit full adder has got two operands, A and B, and input carry, C_{in}. It generates the sum

$$S = A \oplus B \oplus C_{in} \tag{1}$$

and the output carry

$$C_{out} = AB + BC + AC \tag{2}$$

A standard static CMOS VLSI implementation of the 1-bit full adder costs 24 transistors [15].

In order to construct adders with specific properties, standard 1-bit full adders are usually equipped with a few input/output signals. For example, Carry-Look-Ahead adders use 1-bit full adders that generate two signals – propagate, $P = A \oplus B$, and generate, $G = AB$ utilized to look ahead to predict the carry-out [15]. Self-checking adders contain the parity prediction logic [14], [4].

B. Description of the Proposed Adder

Figure 2 shows a new 1-bit self-testing adder which consists of two XOR gates, two inverters and three polymorphic NAND/NOR gates. Except the two inverters, the sum calculation is quite standard,

$$S = \overline{\overline{A} \oplus B \oplus C_{in}}. \tag{3}$$

On the other hand, the carry output is calculated unconventionally using the three NAND/NOR gates that are connected to

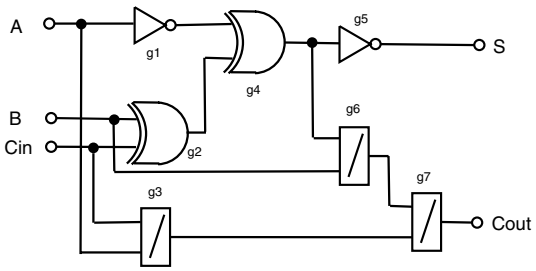

Fig. 2. Proposed 1-bit self-testing adder

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. Let C^{nand} denote the carry-out for the NAND mode and C^{nor} denote the carry-out for the NOR mode. We can observe that

$$C^{nand} = \overline{\overline{\overline{S}B}.\overline{CA}} \tag{4}$$

$$C^{nor} = \overline{\overline{\overline{S} + B} + \overline{C + A}} \tag{5}$$

which can be transformed using DeMorgan's theorem as

$$C^{nand} = \overline{S}B + CA \tag{6}$$

$$C^{nor} = (\overline{S} + B)(C + A) = \overline{S}C + \overline{S}A + BC + AB \tag{7}$$

We can express $\overline{S}$ as

$$\overline{S} = \overline{A}.\overline{B}.\overline{C} + AB\overline{C} + A\overline{B}C + \overline{A}BC \tag{8}$$

and use it in C^{nand} and C^{nor}. Then we obtain

$$C^{nand} = AB\overline{C} + \overline{A}BC + CA = AB + BC + AC \tag{9}$$

and

$$C^{nor} = A\overline{B}C + \overline{A}BC + AB\overline{C} + BC + AB = AB + BC + AC, \tag{10}$$

i.e.

$$C^{nand} = C^{nor} = C_{out}. \tag{11}$$

As the inverter costs 2 transistors, the XOR gate costs 8 transistors and the NAND/NOR gate costs 6 transistors (see Table I), proposed adder would cost 38 transistors, i.e. the overhead is 42% in comparison with a conventional transistor-level implementation. The overhead is similar to other self-checking circuits [7], [2]; however, the proposed adder does not require any additional wiring 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 proposed implementation does not bring any benefits from the functionality point of view. However, it exhibits interesting properties for testing purposes.

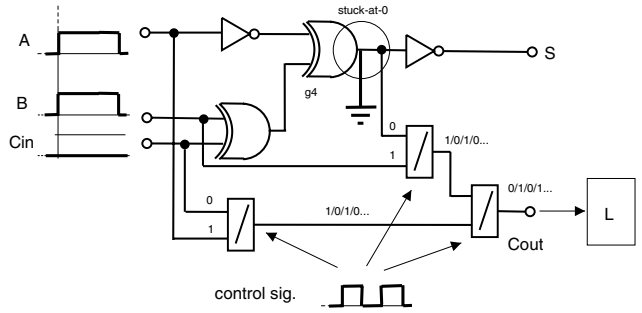

Fig. 3. For the input $A = 1, B = 1, C_{in} = 0$ and stuck-at-0 at the output of gate 4, the carry-out oscillates as Vc oscillates

C. Self-testing Properties

Consider that all the NAND/NOR gates are controlled by Vdd or by some external voltage, Vs (see Table I). Let Vc denote this control voltage, independently of the fact that it can be Vdd or Vs, for the rest of the paper. 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 sum and carry-out outputs of the adder would remain steady (neglecting here some switching spikes), which is a normal operation of the adder.

However, this adder is designed in such a way that if a stuck-at fault is present within the adder then the carry-out output should 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, the carry-out output works as the indicator of a stuck-at fault in the adder.

Figure 3 explains this concept. The control signal is activated for polymorphic gates whenever the adder 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 adder are then considered as test vectors. In this mode, the control signal would oscillate permanently.

D. Analysis of Self-testing Behavior

In order to investigate basic self-testing properties of the proposed adder, we will analyze the circuit 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 gates.

1) For $i = 1$ to $i \leq 7$ 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 – vector of carry-out values for all possible combinations of input values
   d) Set polymorphic gates to mode 2
   e) Calculate R_2 – vector of carry-out values for all possible combinations of input values
   f) Calculate M_{g_i} – the set of input vectors that have induced **different** carry-out values for gate i.

end

Table II 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 when injected to gate 5 or 7. The reason is that these gates are connected directly to the primary outputs of the adder. Hence we will not deal with these gates and faults at these gates in the following text. It is easy to derive from Table II that by applying test vectors $\{1, 2, 3, 5\}$ 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. When a fault is present, the output sum, S, does not oscillate at all. Its value remains correct for stuck-at-faults at gates 3, 6 and 7, and inverted for the remaining faults.

TABLE II
FAULT COVERAGE FOR PROPOSED 1-BIT FULL ADDER

M_g /vector	0	1	2	3	4	5	6	7	stuck-at
M_{g_1}			x		x		x		0
M_{g_2}			x	x	x	x			0
M_{g_3}		x			x				0
M_{g_4}				x		x	x		0
M_{g_5}									0
M_{g_6}	x		x						0
M_{g_7}									0
M_{g_1}		x		x		x			1
M_{g_2}		x					x		1
M_{g_3}				x			x		1
M_{g_4}		x	x		x				1
M_{g_5}									1
M_{g_6}						x		x	1
M_{g_7}									1

E. The n-bit Self-testing Adder

By cascading 1-bit self-testing adders (SDA) we can construct carry-propagate adders. Consider a 2-bit adder (see Fig 4). When a fault is present in SDA1 then the fault is indicated by oscillations at Cout1 as explained for the 1-bit full adder in the previous section. In order to detect a fault at Cout1 which is present in SDA0 then Cout0 has to propagate through SDA1. This can 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 gate 5, 7, 12 or 14.

Only four test vectors, for example (10, 14, 17, 21), are needed to detect all single faults at remaining gates and thus to initiate the oscillations at Cout1. Similarly only four test vectors are sufficient to perform this task for 4-bit adder.

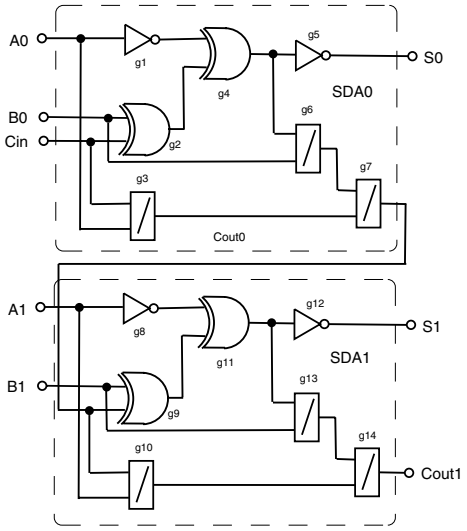

Fig. 4. Proposed 2-bit self-testing adder

The 2-bit adder can naturally be extended to n-bit carry propagate adder. For a 2-bit self-testing adder, the probability of fault detection in SDA1 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 SDA0. 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.

IV. Conclusions

We presented a low-cost implementation of a self-testing adder which indicates 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 test itself. In fact, the adder can permanently be under test because its common input values can serve as test vectors. If these input values are diverse and updated very often then the probability of fault detection increases.

In future work, we will apply the proposed concept of self-testing 1-bit adder to construct more complicated circuits such as fast self-testing adders and multipliers. We will also deal with the problems we have not discussed in this paper. Detailed comparison of the area overhead, the maximum operating frequency, power consumption and fault coverage achieved in the proposed approach and traditional designs of BIST and self checking circuits have to be performed.

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*.

References

[1] M. Diaz, P. Azéma, and J.-M. Ayache. Unified design of self-checking and fail-safe combinational circuits and sequential machines. *IEEE Trans. Computers*, 28(3):276–281, 1979.

[2] M. Garvie. *Reliable Electronics through Artificial Evolution*. PhD thesis, University of Sussex, 2005.

[3] M. Gössel, A. Dmitriev, V. V. Saposhnikov, and V. V. Saposhnikov. A new method for concurrent checking by use of a 1-out-of-4 code. In *6th IEEE International On-Line Testing Workshop*, pages 147–152. IEEE Computer Society, 2000.

[4] A. P. Kakaroudas, K. Papadomanolakis, V. Kokkinos, and C. E. Goutis. 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 International Workshop*, volume 1918 of *Lecture Notes in Computer Science*, pages 187–194. Springer, 2000.

[5] A. Morozov, V. V. Saposhnikov, V. V. Saposhnikov, and M. Gössel. New self-checking circuits by use of berger-codes. In *6th IEEE International On-Line Testing Workshop*, pages 141–146. IEEE Computer Society, 2000.

[6] O. Novak, E. Gramatova, and R. Ubar. *Handbook of Testing Electronic Systems*. Czech Technical University Publishing House, 2005.

[7] V. Ocheretnij, D. Marienfeld, E. S. Sogomonyan, and M. Gössel. Self-checking code-disjoint carry-select adder with low area overhead by use of add1-circuits. In *10th IEEE International On-Line Testing Symposium*, pages 31–36. IEEE Computer Society, 2004.

[8] S. J. Piestrak. Feasibility study of designing tsc sequential circuits with 100% fault coverage. In *17th IEEE Int. Symposium on Defect and Fault-Tolerance in VLSI Systems*, pages 354–364. IEEE Computer Society, 2002.

[9] D. K. Pradhan. *Fault-Tolerant Computer System Design*. Prentice Hall, 1996.

[10] L. Sekanina, L. Starecek, Z. Gajda, and Z. Kotasek. Evolution of multifunctional combinational modules controlled by the power supply voltage. In *Proc. of the 1st NASA/ESA Conference on Adaptive Hardware and Systems*, pages 186–193. IEEE Computer Society, 2006.

[11] A. Stoica, R. Zebulum, X. Guo, D. Keymeulen, I. Ferguson, and V. Duong. Taking Evolutionary Circuit Design From Experimentation to Implementation: Some Useful Techniques and a Silicon Demonstration. *IEE Proc.-Comp. Digit. Tech.*, 151(4):295–300, 2004.

[12] A. Stoica, R. S. Zebulum, and D. Keymeulen. Polymorphic electronics. In *Proc. of Evolvable Systems: From Biology to Hardware Conference*, volume 2210 of *LNCS*, pages 291–302. Springer, 2001.

[13] A. Stoica, R. S. Zebulum, D. Keymeulen, and J. Lohn. On polymorphic circuits and their design using evolutionary algorithms. In *Proc. of IASTED International Conference on Applied Informatics AI2002*, Insbruck, Austria, 2002.

[14] N. A. Touba and E. J. McCluskey. Logic synthesis of multilevel circuits with concurrent error detection. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 16(7):783–789, 1997.

[15] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective (3rd edition)*. Addison Wesley, 2004.

[16] R. S. Zebulum and A. Stoica. Four-Function Logic Gate Controlled by Analog Voltage. *NASA Tech Briefs*, 30(3):8, 2006.

[17] R. S. Zebulum and A. Stoica. Multifunctional Logic Gates for Built-In Self-Testing. *NASA Tech Briefs*, 30(3):10, 2006.