Polymorphic Gates in Design and Test of Digital Circuits*

Lukas Sekanina, Lukas Starecek, Zdenek Kotasek, Zbysek Gajda
Faculty of Information Technology, Brno University of Technology
Bozetechova 2, 612 66 Brno, Czech Republic
sekanina@fit.vutbr.cz, starecek@fit.vutbr.cz, kotasek@fit.vutbr.cz, gajda@fit.vutbr.cz

Abstract

Polymorphic gates are unconventional logic components which can switch their logic functions according to changing environment. The first part of this study presents an evolutionary approach to the design of polymorphic modules which exhibit different logic functions in different environments. The most complicated circuit that we evolved contains more than 100 gates. The second part of this study shows how to reduce the number of test vectors of a digital circuit by replacing some of its gates by polymorphic gates. In the first polymorphic mode, the circuit implements the original function. When switched to the second polymorphic mode, it can be tested using fewer test vectors than in the first polymorphic mode; however, the same fault coverage is obtained. The number of test vectors was reduced on 50-91% of its original volume for six benchmark circuits. The paper also discusses various obstacles which one has to deal with during a practical utilization of polymorphic gates.

1. Introduction

A number of novel principles to implement elementary logic operations have emerged in connection with nanotechnology and molecular electronics in the recent years [22, 2, 20, 6, 4]. These principles range from technological improvements of current technology to truly new molecular devices. Among others *polymorphic electronics* exhibits interesting features. While it merges the capability of performing logic operations with sensing, it can still be implemented using current CMOS technology. Moreover, its potential is enormous with a connection to molecular electronics.

John von Neumann, inspired by neural nets introduced by McCulloch and Pitts [9], proposed nets of digital gates to implement first processors. Stoica et al have introduced the concept of *polymorphic gates* to implement *polymorphic nets* [17, 19, 18]. Polymorphic gates are configurable; their functionality depends on some external factors, for example, on the level of the power supply voltage (Vdd), temperature, light etc. Polymorphic gates would be very useful in building the *embodied intelligence*—intelligent devices whose function emerges in an interaction with a physical environment [1].

For example, a polymorphic gate exists which operates as NAND when Vdd=3.3V and as NOR when Vdd=1.8V. Another gate operates as AND when temperature is 27°C and as OR when temperature is 125°C. It was demonstrated that it is possible to use polymorphic gates together with

*This is draft of paper: Sekanina, L., Starecek, L., Kotasek, Z., Gajda, Z.: Polymorphic Gates in Design and Test of Digital Circuits. Int. Journal of Unconventional Computing. Vol 4, No 2, 2008, p. 125 - 142

ordinary gates to design *multifunctional digital circuits* [14, 16]. These multifunctional circuits exhibit interesting behaviors not visible in standard digital circuits. For example, a circuit was designed that operates as a small adder in the first environment and as a multiplier in the second environment. Its topology is invariable; the only feature which is changed is the functionality of some gates.

The aim of this paper is to demonstrate new principles in design and test of digital circuits which benefit from the use of polymorphic gates. The first goal is to present the most complex multifunctional circuits which can be designed at the gate level nowadays. Cartesian genetic programming is used to perform this task. The second goal of this study is to show that by using polymorphic gates we can reduce the number of test vectors needed to test a combinational circuit. For a given logic circuit we are looking for such a replacement of some of its ordinary gates by polymorphic gates which enables to perform the original function in one of polymorphic modes and easily test the circuit in another polymorphic mode.

The rest of this paper is structured as follows. Section 2 introduces the area of polymorphic electronics and its potential applications. In Section 3, an evolutionary approach to the design of multifunctional circuits is presented. In particular, limits in the complexity of evolved polymorphic circuits are discussed. Section 4 describes a new method proposed for test vector volume reduction using polymorphic gates. Finally, conclusions and future work directions are given in Section 5.

2. Polymorphic Electronics

The concept of polymorphic electronics was proposed by Stoica et al [18]. In fact, polymorphic circuits are multifunctional circuits. The change of their behavior comes from modifications in the characteristics of components involved in the circuit in response to controls such as temperature, power supply voltage, light, etc. [19, 18]. These modifications are related to the transistor's operation point in ordinary technology, or we can speculate that electrical properties of suitable molecules will be changed by applying voltage pulses in molecular electronics (as demonstrated for nitroaniline by Tour [20]).

Research papers indicate various areas in which polymorphic gates could be utilized. The following list provides some examples (see a thorough analysis in [18, 19]): (a) The automatic control of the power consumption when a battery voltage decreases (a circuit realizes another function for lower battery voltage; however, its structure remains unchanged). (b) Implementation of a hidden function, invisible to the user, which can be activated in a specific environment (e.g. watermarking at the hardware level). (c) Intelligent sensors for biometrics, robotics and industrial measurement. (d) Reverse engineering protection. (e) Implementation of low-cost adaptive systems that are able to adjust their behavior inherently in response to certain control variables. (f) Implementation of novel concepts for testing and diagnosing of electronic circuits.

2.1. Polymorphic Gates

Table 1 gives examples of the polymorphic gates reported in literature. For instance, the NAND/NOR gate is the most famous example [17]. The circuit consists only of 6 transistors. It 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° to –200°C. However, most of these gates were only simulated.

2.2. Multifunctional Modules

The use of polymorphic gates as building blocks offers the opportunity to design multifunctional digital modules at the gate level. Once the circuit is designed at the gate level (abstracting thus

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

Gate	control values	control	transistors	ref.
AND/OR	27/125°C	temperature	6	[18]
AND/OR/XOR	3.3/0.0/1.5V	ext. voltage	10	[18]
AND/OR	3.3/0.0V	ext. voltage	6	[18]
AND/OR	1.2/3.3V	Vdd	8	[19]
NAND/NOR	3.3/1.8V	Vdd	6	[17]
NAND/NOR/NXOR/AND	0/0.9/1.1/1.8V	etx. voltage	11	[24]

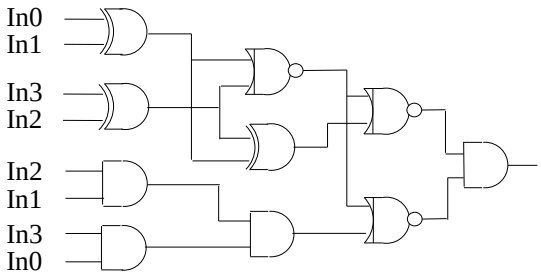

Figure 1. Even-parity circuit when Vdd = 1.2V and majority indicator when Vdd = 3.3V

from the electric level), it does not matter whether this circuit is "reconfigured" by a level of Vdd, temperature or light.

Figure 1 shows an example – a circuit which operates as an even-parity detector when Vdd = 1.2V and as a majority indicator when Vdd = 3.3V. The parity/majority circuit was designed using our methodology presented in [16]. The circuit utilizes ten gates; three of them are the polymorphic NAND/NOR gates controlled by Vdd. When Vdd = 1.2, the polymorphic gates operate as NANDs and the circuit implements the parity function. When Vdd = 3.3V, the polymorphic gates operate as NORs and the circuit implements the majority function. In the both modes, the circuit topology as well as functionality of other gates remains unchanged. Figure 2 shows the four inputs and the output of this circuit for the both levels of Vdd.

In some cases, simulated multifunctional modules contain hypothetical polymorphic gates; in other cases they are composed solely of physically existing polymorphic gates. Research results indicate that it is very difficult to discover circuit topologies for nontrivial multifunctional modules [14, 16]. Evolutionary design techniques are utilized in this process.

2.3. Single-Function Polymorphic Circuits

In another research [25, 15], the goal was to propose self-checking circuits which use polymorphic gates. These circuits perform the same function in both modes of polymorphic gates; however, some of their outputs oscillate when a fault is present in the circuit and the mode of polymorphic gates is periodically changed. These changes in the polymorphic mode represent the state when the circuit is under test. In particular, various evolved adders containing conventional as well as polymorphic gates were 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 [15].

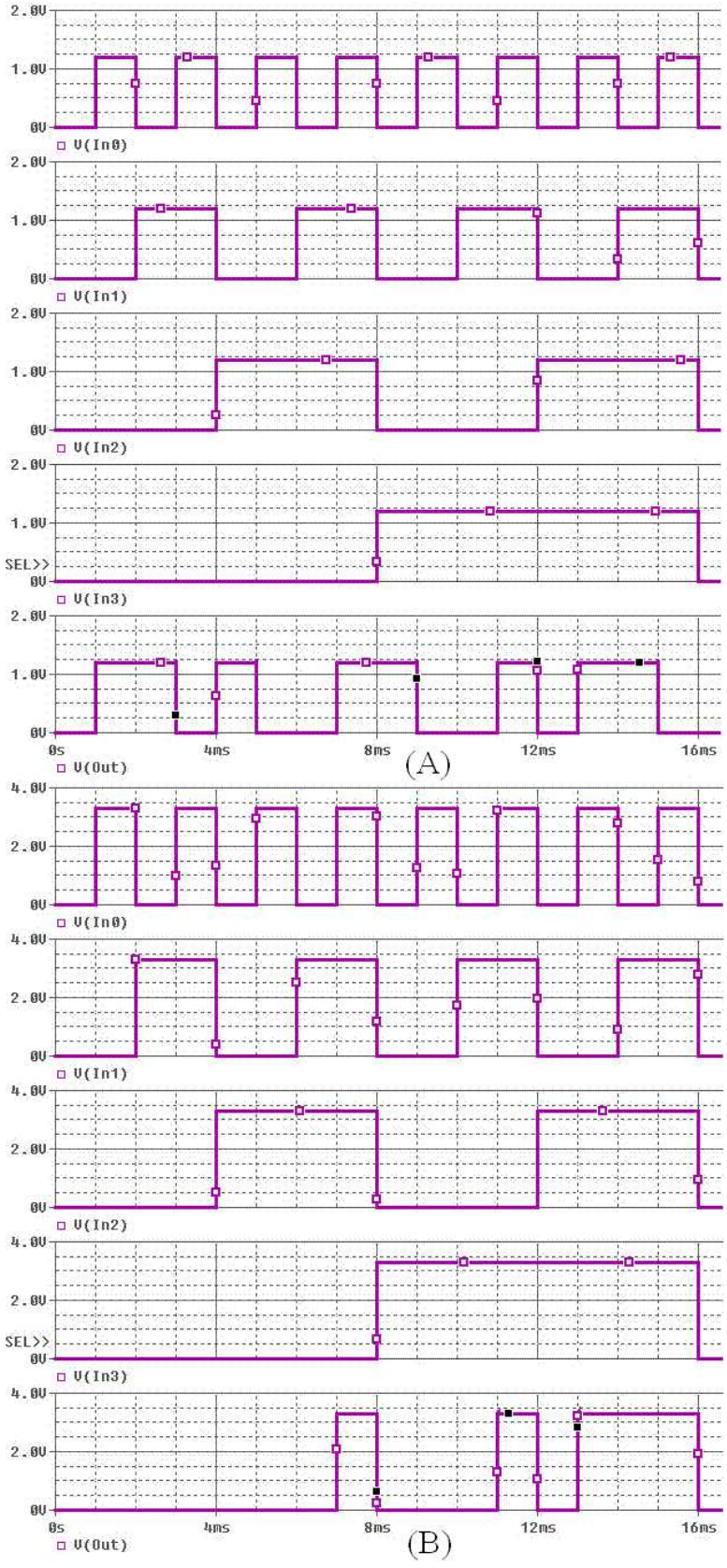

Figure 2. PSpice simulation: (A) even-parity circuit when Vdd = 1.2V, (B) majority indicator when Vdd = 3.3V

3. Evolutionary Synthesis of Polymorphic Modules

Having the polymorphic gates and ordinary gates, our task is to find a digital circuit which performs the first desired function under the first environment and the second desired function under the second environment. Unfortunately, standard methods for logic synthesis are not able to solve this problem. The reason is that these standard methods (such as Espresso, Binary Decision Diagram-based methods etc.) suppose special representations (e.g. a disjunctive normal form) and a set of transformations that manipulate these representations in order to find an optimal implementation of a circuit. Currently, it is unknown how to neither represent polymorphic circuits formally nor perform useful transformations over these representations. A formal background for polymorphic circuit synthesis is needed. In order to find multifunctional circuits at the gate level, an evolutionary design approach is presented which has allowed us to obtain the most complex multifunctional circuits to our best knowledge. This method extends the original work [14].

3.1. Problem Formulation

Let $\Gamma^{(1)}$ denote a set of ordinary gates. Let $\Gamma^{(2)}$ denote a set of polymorphic gates. A polymorphic gate implements two[1] functions according to a control signal which can hold two different values. A gate is in *mode j* (and so performing the j-th function) in the case when j-th value of the control signal is activated. For purposes of this paper, we will denote a polymorphic gate as X_1/X_2, where X_i is its i-th logic function. For example, NAND/NOR denotes the gate operating as NAND in the *mode 1* and as NOR in the *mode 2*. Note that ordinary gates can perform only one function; however, their functionality must be fully defined for each mode. For example, the conventional NAND gate considered for polymorphic circuits must perform the NAND function in the both modes (denoted as NAND/NAND). Let Γ denote a set of all gates, $\Gamma = \Gamma^{(1)} \cup \Gamma^{(2)}$.

A polymorphic circuit can formally be represented by a graph $G = (V, E, \varphi)$, where V is a set of vertices, E is a set of edges between the vertices, $E = \{(a, b)|a, b \in V\}$, and φ is a mapping assigning a function (gate) to each vertex, $\varphi : V \rightarrow \Gamma$. As usually, V models the gates and E models the connections of the gates. A circuit (and also its graph) is in the *mode j* in the case when all gates are in the *mode j*.

Given Γ and logic functions f_1 and f_2 required in different modes, the problem of the multifunctional circuit synthesis at the gate level is formulated as follows: Find a graph G representing the digital circuit which performs logic function f_1 in its first mode and logic function f_2 in its second mode. Additional requirements can be specified, e.g. to minimize the delay, area, power consumption etc.

3.2. Evolutionary Algorithm

The problem defined in the previous section is approached using Cartesian genetic programming (CGP) which has recently been applied by several researchers especially for the evolutionary design of combinational circuits [11]. In CGP, a circuit is modeled as an array of u (columns) $\times$ v (rows) of programmable elements (gates). The number of circuit inputs, n_i, and outputs, n_o, is fixed. Feedback is not allowed. Each gate input can be connected to the output of a gate placed in the previous L columns or to some of circuit inputs. The L parameter, in fact, defines the level of connectivity and thus reduces/extends the search space. For example, if L=1 only neighboring columns may be connected; if $L = u$, the full connectivity is enabled. Each gate is programmed

[1] This can naturally be extended for k different functions.

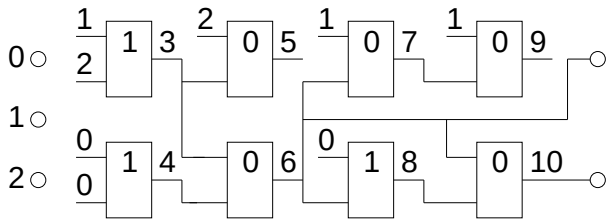

Figure 3. An example of a 3-input circuit. CGP parameters are as follows: $L = 3$, $u = 4$, $v = 2$, Γ = {AND (0), OR (1)}. Gates 5 and 9 are not utilized. Chromosome: 1,2,1, 0,0,1, 2,3,0, 3,4,0 1,6,0, 0,6,1, 1,7,0, 6,8,0, 6, 10. The last two integers indicate the outputs of the circuit.

to perform one of functions defined in the set Γ. Figure 3 shows an example and a corresponding chromosome. Every individual is encoded using $u \times v \times 3 + n_o$ integers.

CGP operates with the population of λ individuals (typically, $\lambda = 5 - 20$). The initial population is randomly generated. Every new population consists of the best individual and its mutants. In case when two or more individuals have received the same fitness score in the previous population, the individual which did not serve as a parent in the previous population will be selected as a new parent. This strategy is used to ensure the diversity of population. In case when the evolution has found a solution which produces correct outputs for all possible input combinations, the number of gates is getting to minimize. Delay is not optimized in this research. The evolution is stopped when the best fitness value stagnates or the maximum number of generations were exhausted.

The fitness value is defined as follows:

$$fitness = B_1 + B_2 + (u.v - z) \tag{1}$$

where B_1 (resp. B_2) is the number of correct output bits for f_1 (resp. f_2) obtained as response for all possible input combinations, z denotes the number of gates utilized in a particular candidate circuit and $u.v$ is the total number of programmable gates available. The last term is considered only if the circuit behavior is perfect in the both modes; otherwise $uv - z = 0$.

3.3. Results

We evaluated proposed algorithm using difficult benchmark circuits – Multiplier/Sorting Network circuits of 4 – 7 inputs. These circuits multiply two operands in the first mode and sort an input vector in the second mode. These circuits were chosen because multipliers are standard (and nontrivial) benchmark circuits for evolutionary circuit design techniques. The best-known results (implementation costs) for small multipliers (evolved solutions with up to 4-bit operands) are given in [21]. Implementation costs of sorting networks (SN) are given in [8] (one comparator is implemented using two gates – AND and OR).

Table 2 summarizes parameters of experiments and obtained results. In all experiments, the population size was 15 and up to 100 million generations were produced in each run. The numbers of generations correspond to other experiments with the evolutionary multiplier design using CGP in which, for example, 100 million generations are needed to evolve a 4×3-bit multiplier [21].

Similarly to the evolution of multipliers [21], we can observe that the average number of generations grows exponentially with the growing number of inputs. It is difficult to evolve 7-input and more input circuits. No correct solution was obtained for 8-input circuits at the gate level. Figure 4 shows the best implementation of 2×2-bit multiplier/4-bit sorter which utilizes 23 gates (18 polymorphic NAND/NOR gates and 5 conventional AND gates). Moreover, two NAND/NOR gates can

Table 2. Parameters and results of the evolutionary design using CGP and circuit simulator. Gates in "Gate set" are numbered as: (1) NAND/NOR, (2) AND, (3) OR, (4) XOR, (5) NAND, (6) NOR, (7) NOT A, (8) NOT B, (9) MOV A and (10) MOV B, where MOV denotes the identity operation.

Multiplier/Sorter	$2b \times 2b/4b$	$3b \times 2b/5b$	$3b \times 3b/6b$	$4b \times 3b/7b$
$u \times v$	10×12	100×1	120×1	16×16
L-back	1	100	120	16
Mutation (genes)	1	2	4	4
Gate set	1, 2, 9, 10	1–4, 9, 10	1–10	1, 2, 9, 10
Runs	10	10	10	10
Successful runs	10	10	9	3
Generations (average)	52,580	854,900	26,972,648	62,617,151
Min. # of gates	23	30	52	113

Table 3. Comparison of the implementation cost (# of gates) for polymorphic modules implemented by (a) multiplexing conventional solutions and (b) using CGP and polymorphic gates

Inputs (bits)	(1) Multiplier	(2) Sorter	(a) Multiplexing (1), (2)	(b) CGP
2 + 2	7	10	$17 + 4c_m$	23
3 + 2	13	18	$31 + 5c_m$	30
3 + 3	23	24	$47 + 6c_m$	52
3 + 4	38	32	$70 + 7c_m$	113

be replaced by a single inverter. This implementation has 2 gates less than the best known solution reported in [14].

The computation time can be expressed as follows. In order to generate 1 million generations for a 4-input multifunctional circuit, 80 seconds are spent at Athlon64 3200+ processor. For 7-input multifunctional circuit, 207 seconds have to be spent at the same processor.

3.4. Discussion

The 7-input multiplier/sorter is the most complex multifunctional module evolved so far. We probably reached the limit of this method in terms of generated circuit complexity. A similar limit was reported for the evolution of ordinary combinational circuits using CGP where the 4×4-bit multiplier is the most complicated circuit evolved directly at the gate level [21]. Note that this is valid for a direct evolution at the gate level; for example, incremental evolution is not taken into account.

Another approach to the implementation of multifunctional circuits is to use conventional implementations for both required modes and perform multiplexing their outputs by polymorphic multiplexers (or by a standard multiplexer controlled by a Vdd sensor). Table 3 compares the number of gates required for the both implementations (no sharing of gates is assumed herein). It is evident that in some cases the evolved circuits are more gate-efficient than the circuits multiplexing conventional implementations. c_m denotes the cost of a polymorphic two-input multiplexer which must be taken into account (it is assumed that $c_m = 8 - 14$ transistors, i.e. approx. 2–3 gates).

Only the NAND/NOR polymorphic gate has been considered in our designs because only for this gate a transistor-level implementation is available. It is assumed that more efficient implementations

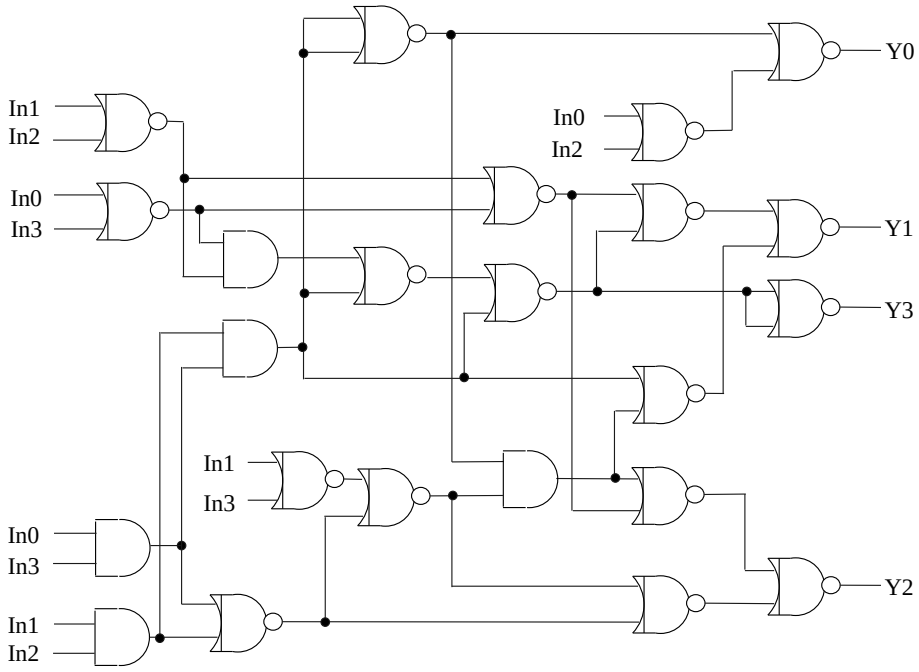

Figure 4. The best evolved 2b-Multiplier/4b-Sorter

would be evolved if other polymorphic gates were available for CGP. Definitely, the most-area efficient implementations of multifunctional modules would be obtained if the evolutionary design operates directly at the transistor level. However, as the evolutionary design of simple polymorphic gates clearly demonstrates [17], it is very difficult to find a reliable implementation even of a two-input polymorphic gate.

The solution to the multifunctional circuit synthesis problem is still an open question. As we do not know how to solve this problem by means of conventional methods, evolutionary design approaches have been utilized. However, the gate level evolution is not scalable.

4. Reduction of Test Vectors Volume

The goal of the second study is to demonstrate that by replacing some standard gates by suitable polymorphic gates we can improve testability of a given digital circuit. We will again work at the gate level, assuming that suitable polymorphic gates exist.

4.1. Test Generation

In diagnostics and testing of conventional digital electronic systems two areas of problems exist: test generation and test application. One possibility how to generate test for a sequential circuit is by means of automatic test pattern generator for sequential circuits (SATPG). Automatic test pattern generation for sequential circuits is generally considered to be a hard problem. Full scan design techniques attempt to alleviate this problem by connecting all flip-flops (FFs) or latches into a scan path during test mode so that all these elements become easily controllable and observable. Then, in a full scan based design, the portion of the circuit excluding the scan path is fully combinational which allows to use automatic test pattern generator (ATPG) to generate a test for unit under test (UUT) restructured in this way. Two reasons against the full-scan techniques exist:

1. The test application time associated with full-scan may be extremely high.

2. The full scan may be prohibitively expensive due to high area overhead.

The solution to the second objection can be seen in partial scan techniques which require analysis to choose which registers are best for scan [12, 7, 23, 5] while the reduction of test application time has been addressed in arranging scan flip-flops in parallel scan chains. Implementing full scan into UUT brings an important consequence, namely the *reduction of test vectors volume* because UUT is subdivided into structures which are seen as purely combinational components and the test can be generated by ATPG. Numerous activities exist to reduce the number of test vectors needed and thus reduce the test application time. These considerations become important in such implementations which contain thousands of flip-flops structured into full or partial scan chains. They are useful in designs which consist of complex combinational components (either being part of sequential designs or purely combinational circuits). Thus, it is reasonable to develop methodologies which reduce the number of test vectors needed to test combinational nets. Other activities the objective of which is the reduction of test data volume, can be also recognized in the field of test data compaction [3, 13].

The methodology presented in this study can be combined with any above mentioned methodology which result in test application time reduction. The methodology can be classified as "test generation strategies", which when combined with "test application strategies" can reduce both test application time and power consumption during its application.

The basic assumption of the proposed method is that ATPG tools do not inspect internal failures of gates because only the circuit structure is tested in which gates are considered as black boxes. In general, test generation methods and tools are based on the assumption that failures which arise in the internal structure of a component are propagated to component output as either stuck-at-zero or stuck-at-one values. This is the principle which is widely accepted and expresses the relation between internal and external failures of an electronic component (e. g. gate) and thus allows to generate test vectors for its external nodes only. Therefore, in our research we do not study a failure model of polymorphic gates.

4.2. Problem Formulation

Given a conventional circuit which performs function f_1 and whose topology is represented by $G^{(1)} = (V, E, \varepsilon)$, $\varepsilon : V \rightarrow \Gamma^{(1)}$, the problem targeted in this paper is to find φ which will constitute $G = (V, E, \varphi)$, $\varphi : V \rightarrow \Gamma$, with the following properties: (1) G performs logic function f_1 in its first polymorphic mode. This ensures that the original circuit function will be kept. (2) In polymorphic mode 2, G represents such a circuit which exhibits *better* diagnostic properties than the circuit switched to the first polymorphic mode.

Therefore, the goal is to find a circuit which can be tested using fewer test vectors in the second polymorphic mode compared with the first polymorphic mode; however, still providing the same fault coverage. Replacing standard gates by polymorphic gates (i.e. replacing ε by φ) increases the cost of the circuit. Hence there could be a trade off between the level of testability and the implementation cost.

4.3. Proposed Method

In order to confirm that the use of polymorphic gates can reduce the number of test vectors when a circuit is switched to the second polymorphic mode, we developed an algorithm which identifies and modifies some gates of the circuit and aims to achieve the same fault coverage with fewer test vectors. The following conditions have to be ensured during the replacements of gates:

1. Only those ordinary gates can be replaced which have the same interface as corresponding polymorphic gates (e.g. a two-input ordinary gate can be replaced only by a two-input polymorphic gate).
2. In order to keep the original circuit function in the first polymorphic mode, it is important to ensure that an ordinary gate with logic function g can be replaced only by a polymorphic gate g/h which performs logic function g in its first polymorphic mode.

For benchmark circuit C which contains gates only from $\Gamma^{(1)}$, the procedure is as follows:

1. For C, find the minimum number of test vectors, $N^{min}(C)$, which ensure p% fault coverage.
2. Generate all possible replacements (respecting conditions (1) and (2)) in C to obtain polymorphic circuits $C/R_1 \ldots C/R_{max}$ which have the same topology as C but contain gates from Γ.
3. For each circuit C/R_i which is switched to the second polymorphic mode, find the minimum number of test vectors, $N^{min}(C/R_i)$, which guarantees the p% fault coverage.
4. Select those C/R_i which exhibit $N^{min}(C/R_i) < N^{min}(C)$.

As the proposed algorithm is deterministic, it always finds the optimum solution. Unfortunately, its computational requirements are high. Consider that C contains k gates and each of them can be replaced by one of m polymorphic gates. Then the number of possible replacements is k^m. Hence the approach is applicable only for relatively small circuits. The algorithm was evaluated on a set of benchmark circuits. All test vectors are calculated using ATPG FlexTest.

4.4. Results

Table 4 shows ordinary and polymorphic gates used in resulting circuits. The cost of ordinary gates in terms of the number of transistors is given by AMI 1.2um library [10]. Note that Table 4 shows only those polymorphic gates which were utilized in discovered solutions; in fact, we tested 20 different polymorphic gates in our experiments. As utilized polymorphic gates have not been described in literature so far, their cost is estimated. In most cases we have simply added the transistor costs of both functions. With respect to Table 1, it is reasonable to believe that this cost might be significantly reduced in a real implementation. For example, the NAND/NOR gate controlled by Vdd costs 6 transistors; however, the NAND usually costs 4 transistors, the NOR usually costs 4 transistors and some transistors have to be used to implement multiplexing according to Vdd. A conventional transistor-level implementation of NAND/NOR controlled by another logic signal would cost 10 transistors. Note that AO denotes the AND-OR logic structure and AOI denotes AND-OR-Invert logic structure.

Table 5 shows results obtained for six benchmark circuits of size 4–13 gates which include three implementations of a full 1-bit adder, 3-bit comparator, 8-to-3-bit encoder and 3-to-8-bit decoder (see examples in Figs 5 and 6 in which polymorphic gates are shown as boxes labelled using the X/Y notation). Because the proposed algorithm is deterministic, only these relatively small circuits could be utilized in this initial study. In Table 5, 'pg' denotes the number of polymorphic gates used in modified circuits, 'tr' denotes the number of transistors, 'trp' is the number of transistors in modified circuits and 'cmb' is the number of combinations evaluated by the FlexTest (for $p = 100\%$). Column 'rcmb' gives the fraction of combinations which lead to some reductions in the test vectors volume. We can observe that the number of test vectors was reduced to 50-91 % and the estimated cost increased by 11-60 %. Only a few replacements out of all possible replacements have led to some reduction of test vectors volume.

Table 4. The cost of conventional and polymorphic gates

gate	transistors	gate	transistors	polymorphic gate	transistors
INV01	2	NOR03	6	INV01/BUF02	6
INV02	2	NOR04	8	INV02/BUF02	6
BUF02	4	XOR2	12	AND02/XNOR2	16
AND02	6	XNOR2	10	AND03/NOR03	14
AND03	8	AO21	12	OR02/XNOR2	16
OR02	6	AOI21	10	AO21/AOI21	22
OR03	8	AO22	10	XNOR2/OR02	16
NAND02	4	OAI21	6	XOR2/NAND02	16
NOR02	4	OAI32	10		

Table 5. Properties of benchmark circuits and their modifications

circuit	gates	pg	$N^{min}(C)$	$N^{min}(C/R)$	N^{min}(%)	tr.	trp.	tr(%)	cmb	rcmb(%)
fulladd1	4	1	6	5	83%	32	38	119%	216	8.33
fulladd2	6	3	6	4	67%	40	64	160%	15 552	16.94
fulladd3	5	2	6	4	67%	42	54	129%	7 776	17.00
comp3bit	11	2	11	9	82%	62	70	113%	124 416	1.40
enc8to3	13	1	11	10	91%	74	82	111%	165 888	0.02
dec3to8	11	4	8	4	50%	56	74	132%	524 288	2.05

Table 6. Original test vectors and modified test vectors for benchmark circuits

circuit	original test vectors → new test vectors
fulladd1	100, 000, 111, 110, 001, 010 → 100, 000, 111, 110, 101
fulladd2	100, 000, 111, 110, 001, 010 → 100, 000, 111, 110
fulladd3	100, 000, 111, 110, 001, 010 → 100, 111, 110, 001
comp3	100011, 000110, 001100, 011000, 000000, 011011, 110110, 000010, 100101, 010001, 010011 → 100011, 000110, 011000, 110110, 101100, 100111, 100100, 110111, 110100
enc8to3	10001100 00011000 00110000 01100000 11000000 00000000 00100000 01000000 00001001 00000111 00000010 → 10001100 00011000 001100000 01100000 00000000 00100000 01000000 00001001 00000111 00000010
dec3to8	100, 000, 111, 110, 011, 101, 010, 001 → 100, 000, 010, 001

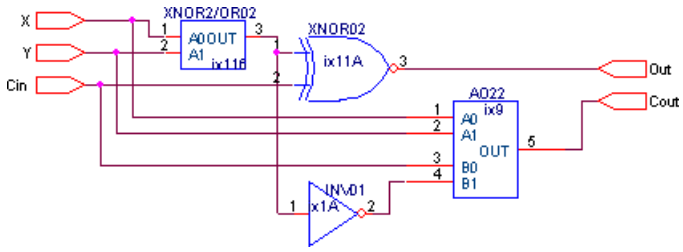

Figure 5. Circuit fulladd1 after modification

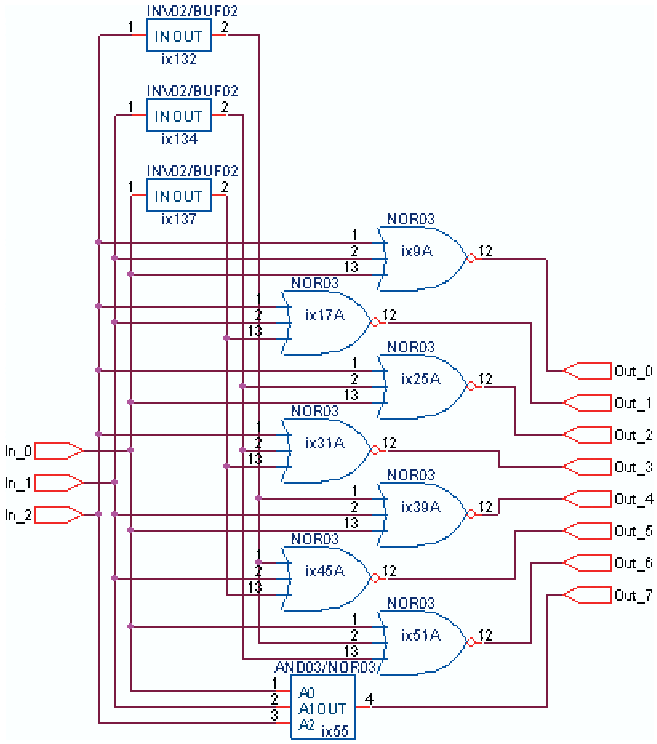

Figure 6. Circuit dec3to8 after modification

4.5. Discussion

The basic hypothesis behind this experiment has been confirmed: Yes, the number of test vectors needed to test a combinational component can be reduced if a possibility of converting the function of selected gates is available during test generation. We have shown how polymorphic gates can be used for this purpose. Now we have to deal with possible objections.

As test is not generated for the original circuit it says nothing about the original circuit. It is a fact, that one possibility how to generate a test is based on developing a test sequence for stuck-at-0 and stuck-at-1 faults on the internal connections of the component. It is evident that these faults are then the same for both versions of the circuit (i. e. the original and the modified one) as far as internal connections and I/O ports of both of them are concerned. Thus, it can be stated that the test of one of them covers the faults of the other one.

The control signal of polymorphic gates can influence the testability. It is reasonable to assume that the control signal of polymorphic gates can be either an external logic signal or Vdd. In the first case, this signal is considered as another input of the circuit which will probably influence the test generated by ATPG. Future research will be devoted to analyze this phenomenon. In case of controlling via Vdd, all circuit elements have to work correctly for two different levels of Vdd. If it is possible then the concept is valid as proposed.

Experiments were performed on small circuits. As the proposed algorithm is deterministic, its time requirements are too high. The computation time ranges from 2 minutes for the smallest circuit to 2.5 days for the largest circuit when measured on a server equipped with Opteron2216 and 4GB RAM. On the other hand the method provides an optimal solution for a given specification. To be able to evaluate the methodology on more complex circuits, we do plan to utilize a stochastic algorithm based on a genetic algorithm. A candidate solution could be encoded as a string of $k.log_2 m$ bits (where k is the number of gates and each of them can be replaced by one of m

polymorphic gates). It is reasonable to assume that this approach could provide a good suboptimal solution for circuits consisting of thousands of gates and $m \leq 32$ in a reasonable time.

The method works only for some circuits. Experiments performed so far indicate that a small reduction can be obtained in all types of circuits that we tested. Of course, the method can fail for some other circuits.

Suitable polymorphic gates could be expensive. Depending on the type of circuit, the obtained reduction could be insignificant in relation to the cost of inserted polymorphic gates. The utilization of this method depends on a particular application.

5. Conclusions

In this article we have described some of possible applications of polymorphic electronics. Firstly, it was shown that evolutionary design can generate gate-level polymorphic modules with the complexity of around 100 gates. Thus, the most complex gate-level multifunctional circuits, to our best knowledge, were obtained. Secondly, a new application of polymorphic electronics was demonstrated in the area of circuit testing. We were able to reduce the number of test vectors for six circuits when some of gates were replaced by polymorphic gates. We discussed various obstacles which one has to deal with during a practical utilization of this method. Another important consequence of the proposed methods is that by using unconventional circuit components we can solve traditional problems in a new way. Without the use of polymorphic gates, the proposed methods make no sense.

Our direction for future research is closely connected to potential applications of this technology. The class of applications that we can see now is related to a new generation of field programmable gate arrays in which some parts (such as interconnection switches, global reset functions or test circuits) could be implemented using area-efficient polymorphic gates reducing thus the area on the chip while maintaining multiple functionality and good testability. It would require reliable implementations of elementary polymorphic gates and, of course, much more work in this area.

In future, with the development of molecular electronics and synthetic biology, we could observe a class of polymorphic gates implemented in a unconventional way (e.g. as molecules changing their logic function according to the level of illumination). Then, circuits composed of those polymorphic gates could fully exploit advantages of simultaneous logic operation and the capability of sensing. We do believe that it is the main reason why the development of the methods for the multifunctional circuit synthesis and testing is important right now.

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] R. Brooks. The relationship between matter and life. *Nature*, 409(6816):409–411, 2001.

[2] J. Chen, N. Jonoska, and G. Rozenberg. *Nanotechnology: Science and Computation*. Springer, 2006.

[3] S. R. Das, C. V. Ramamoorthy, M. H. Assaf, E. M. Petriu, J. Wen-Ben, and M. Sahinoglu. Fault simulation and response compaction in full scan circuits using HOPE. *IEEE Transactions on Instrumentation and Measurement*, 54(6):2310–2328, 2005.

[4] A. DeHon and H. Naeimi. Seven strategies for tolerating highly defective fabrication. *IEEE Design and Test of Computers*, 22(4):306–315, 2005.

[5] A. Efthymiou, J. Bainbridge, and D. A. Edwards. Test pattern generation and partial-scan methodology for an asynchronous soc interconnect. *IEEE Trans. VLSI Syst.*, 13(12):1384–1393, 2005.

[6] International technology roadmap for semiconductors, 2005. URL: http://public.itrs.net.

[7] P. Kalla and M. J. Ciesielski. A comprehensive approach to the partial scan problem using implicitstate enumeration. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 21(7):810–826, 2002.

[8] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching (2nd ed.)*. Addison Wesley, 1998.

[9] W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(1):115–133, 1943.

[10] Mentor graphics, adk html data book, 2007. URL: http://germanium.cs.wustl.edu/HEP/ADK/HTMLdatabook/AMI12databook.htm.

[11] J. Miller, D. Job, and V. Vassilev. Principles in the Evolutionary Design of Digital Circuits – Part I. *Genetic Programming and Evolvable Machines*, 1(1):8–35, 2000.

[12] S. Park. A partial scan design unifying structural analysis and testabilities. *Int. J. Electronics*, 88(12):1237–1245, 2001.

[13] I. Pomeranz and S. M. Reddy. Static test compaction for multiple full-scan circuits. In *21st International Conference on Computer Design (ICCD 2003), VLSI in Computers and Processors*, pages 393–396. IEEE Computer Society, 2003.

[14] L. Sekanina. Evolutionary design of gate-level polymorphic digital circuits. In *Applications of Evolutionary Computing*, volume 3449 of *LNCS*, pages 185–194, Lausanne, Switzerland, 2005. Springer Verlag.

[15] L. Sekanina. 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 DDECS 2007*, pages 246–246, Krakow, Poland, 2007. IEEE Computer Society.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] J. M. Tour. *Molecular Electronics*. World Scientific, 2003.

[21] V. Vassilev, D. Job, and J. F. Miller. Towards the automatic design of more efficient digital circuits. In *Proc. of the 2nd NASA/DoD Workshop of Evolvable Hardware*, pages 151–160, Los Alamitos, CA, US, 2000. IEEE Computer Society.

[22] R. Waser. *Nanotechnology: Science and Computation*. Wiley-VCH, 2005.

[23] D. Xiang and J. H. Patel. Partial scan design based on circuit state information and functional analysis. *IEEE Trans. Computers*, 53(3):276–287, 2004.

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

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