Design Methods for Polymorphic Digital Circuits

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

Abstract. *Polymorphic electronic gates have been introduced in the recent years. The gates are able to change their functionality as a consequence of the change of a sensitive variable, which can be a power supply voltage, temperature, etc. In this paper various methods are proposed, analyzed and discussed for designing polymorphic gate-level electronic circuits.*

1 Introduction

Papers [2, 3, 4] have introduced the area of *polymorphic electronics*. They show that it is possible to create digital gates whose functionally can be controlled in a non-traditional way: by temperature, power supply voltage (Vdd), some external signals etc.

Polymorphic electronics represents a promising area of circuit design since it allows engineers to build inherently adaptable digital circuits. By changing the temperature, Vdd or some other conditions a circuit can change its functionality immediately, with no reconfiguration overhead. The potential applications include: special circuits that are able to decrease resolution of digital/analog converters or speed/resolution of a data transmission when a battery voltage decreases, circuits with a hidden/secret function that can be used to ensure security, intelligent sensors and novel solutions for reconfigurable cells of reconfigurable devices (such as FPGA and CPLD) [3].

The objective of this paper is to explore the ways in which polymorphic digital circuits can be designed by using polymorphic gates as elementary building blocks. Although only a few types of polymorphic gates have been proposed so far we suppose in this paper that arbitrary polymorphic gates exist and can be used as building blocks. In this paper conventional and evolutionary design methods are proposed to create polymorphic circuits.

2 Polymorphic Electronics

Polymorphic circuits are in fact 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. [3]. Table 1 gives examples of the polymorphic gates reported in literature. For instance, the NAND/NOR gate is the most famous example

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

Gate	control values	control method	transistors	ref.
AND/OR	27/125C	temperature	6	[4]
AND/OR/XOR	3.3/0.0/1.5V	external voltage	10	[4]
AND/OR	3.3/0.0V	external voltage	6	[4]
AND/OR	1.2/3.3V	Vdd	8	[3]
NAND/NOR	3.3/1.8V	Vdd	6	[2]

[2]. The circuit consists of 6 transistors and was fabricated in a 0.5-micron CMOS technology. The circuit is stable for ±10% variations of Vdd and for temperatures in the range 20C – 200C. Note that the common NAND and NOR gates cost 4 transistors, the XOR gate can be implemented using 10 transistors and 2 transistors are needed to create the inverter in the standard CMOS technology.

Potential applications are discussed in [3]. The design of polymorphic circuits is considered as the main problem because these circuits typically utilize normally unused characteristics of electronic devices and working environment. A. Thompson has shown that unconstrained evolutionary design is able to produce innovative designs that effectively utilize these characteristics [5].

3 The Proposed Design Approaches

3.1 Problem Formulation

Let P be a set of polymorphic gates. Each of them is able to implement up to K functions (K is specified beforehand) according to a control signal which holds up to K different values. A gate is in *mode* j (and so performing the j-th function) in the case that j-th value of the control signal is activated. Given P and logic functions $f_1 \dots f_K$ required in different modes, the problem of the polymorphic circuit design at the gate level is formulated as follows: Find a graph G representing the digital circuit which performs functions $f_1 \dots f_K$ in its modes $1 \dots K$. Additional requirements can be specified, e.g. to minimize delay, area, power consumption etc. In the next sections we will consider only $K = 2$.

3.2 Conventional Approach

By *Conventional approach* we mean a standard mathematically based method (see [6]) which is similar to the conventional methods used to create digital circuits from common specifications. While we are able to specify the behavior of a polymorphic circuit by means of truth table(s), we do not how to modify the conventional methods in order to design compact implementations of the polymorphic circuits (when polymorphic gates are considered as building blocks). More precisely, no efficient conventional algorithm is known to construct compact G if P and $f_1 \dots f_K$ are given. By *compact* we mean low-area demanding circuits. Therefore, the only option seems to invent a new method or to use ad hoc approaches.

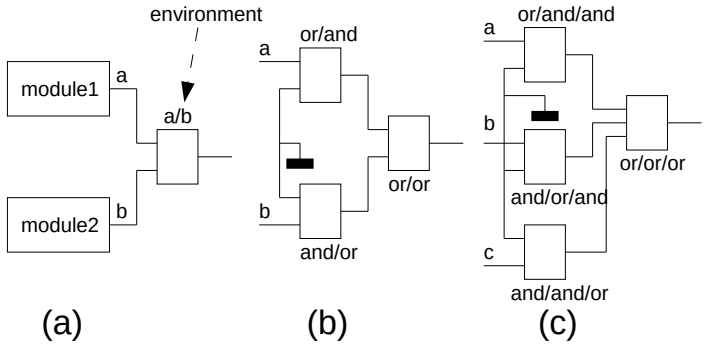

Figure 1: Polymorphic multiplexers: (a) the idea, (b) 2-input circuit, (c) 3-input circuit

3.3 Polymorphic Multiplexers

An alternative approach is to employ polymorphic multiplexers to switch between digital modules designed conventionally (see Fig. 1a). If the modules operate correctly in all required environments, only the multiplexers have to be composed of polymorphic gates. However, no polymorphic multiplexer has been presented in the literature so far. Fig. 1b and c show the solutions we are proposing at the gate level. Perhaps a more efficient transistor-level solution will be available in future (Note that a conventional CMOS multiplexer costs 6 transistors). Let C_{pmpx} denote the cost of a polymorphic multiplexer. For instance, $C_{pmpx} = 18$ for the multiplxer shown in Fig. 1b.

3.4 Evolutionary Approach

The evolutionary approach (and Cartesian genetic programming (CGP) in particular) has shown a certain success for automatic designing of combinational circuits [1]. CGP models a reconfigurable circuit, in which digital circuits are evolved, as an array of u (columns) $\times$ v (rows) of programmable elements (gates) [1]. Feedback is not allowed. Each gate input can be connected to the output of some gate placed in the previous columns or to some of the circuit inputs. L-back parameter defines the level of connectivity and thus reduces/extends the search space. We will use $v = 1$ and $u = L$ in the proposed experiments.

The proposed algorithm operates with the population of 128 individuals; every new population consists of mutants of the best four individuals. Only the mutation operator has been utilized that modifies one randomly selected gene of an individual. In case that evolution has found a solution which produces the correct outputs for all possible input combinations, the number of gates is getting to minimize. Delay is not optimized. The computation is terminated in case that no improvement of the best fitness value has been reported in a given number of last generations (typically in 50000 generations). The fitness value is defined as follows:

$$fitness = B_1 + B_2 + (u - g) \tag{1}$$

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, g denotes the number of gates utilized in a particular candidate circuit and u is the total number of gates available. The last term is considered only if the circuit behavior is perfect in the both modes; otherwise $u - g = 0$.

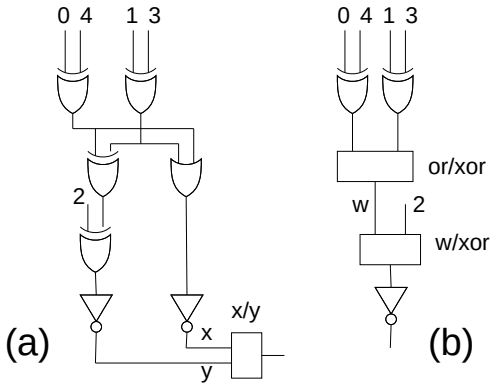

Figure 2: Polymorphic boolean symmetry – parity circuit (the inputs: 0–4): (a) with the polymorphic multiplexer, (b) the ad hoc design.

4 The Obtained Results

4.1 Polymorphic 5bit Boolean Symmetry vs. Parity Circuit

In order to illustrate the ad hoc approach we decided to design a polymorphic 5bit parity vs. Boolean symmetry circuit. The Boolean symmetry circuit returns log. 1 if the input vector is symmetric. Figure 2a shows the conventional implementations of these circuits as well as their composition in a single polymorphic circuit. As these circuits are structurally similar, it is not difficult to combine them together manually (Figure 2b). The ad hoc approach produced a compact circuit which costs 42 transistors (assuming 10 transistors per a polymorphic gate). The solution with the polymorphic multiplexer costs $46 + C_{pmpx}$ transistors. The problem is that the ad hoc approach is not applicable for more complicated circuits.

4.2 Polymorphic 5bit Boolean Symmetry vs. Median Circuit

The 5bit median circuit returns the middle bit of a sorted 5-bit input vector. Its implementation requires 5 OR and 5 AND gates, i.e. 60 transistors. Because the Boolean symmetry and median circuits are structurally totally different it is difficult to create the polymorphic circuit manually. Hence only the multiplexer-based solution and evolutionary design will be compared. Figure 3 shows the evolved polymorphic Boolean symmetry – median circuit, which costs 102 transistors. The circuit was evolved using CGP ($u = 24, v = 1$). We obtained 7 perfect circuits out of 200 runs. On the other hand the polymorphic multiplexing of standard solutions costs $28 + 60 + C_{pmpx}$ transistors for the same polymorphic circuit.

4.3 Polymorphic 5bit Median vs. Parity Circuit

A 13-component polymorphic 5bit median – parity circuit (composed of polymorphic gates NAND/XOR and XOR/NOR) was evolved. As we do not know the implementation cost of these gates we cannot compare the circuit against the solution with a polymorphic multiplexer (which costs $102 + C_{pmpx}$ tr.). Another 5bit median – parity circuit was evolved using NAND/NOR and XOR gates. That circuit is composed of 106 transistors.

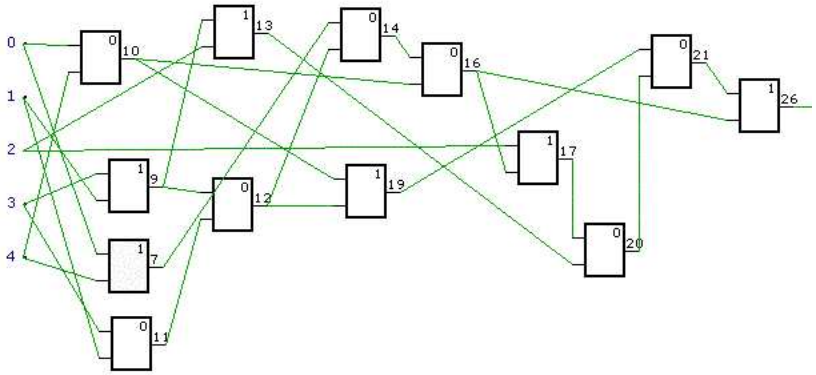

Figure 3: Polymorphic boolean symmetry – median circuit (the inputs: 0–4; gates: 0 – NAND/NOR, 1 – XOR/XOR). Label **0** denotes the first polymorphic gate and label **1** is the second gate of the two used. The bit 0 is LSB.

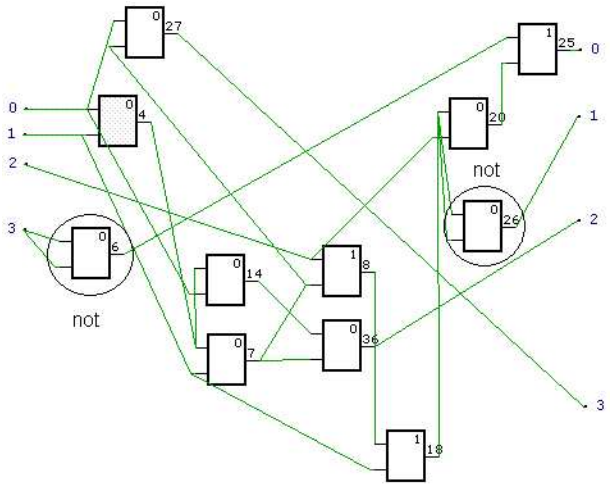

Figure 4: The evolved polymorphic plus1 – plus7 circuit (the inputs: 0–3; gates: 0 – NAND/NOR, 1 – XOR)

4.4 Polymorphic 4bit Plus1 vs. Plus7 Circuit

The Plus1 circuit accepts a four-bit input and adds 1 in order to create a four-bit output. We utilized CGP to find its implementation – it consists of 3 XOR, 1 NAND, 1 NOR and 2 NOT gates, i.e. of 42 transistors. The Plus7 circuit works similarly to Plus1 but it adds 7. It consists of 1 NOT, 1 NAND, 1 NOR and 3 XOR gates, i.e. of 40 transistors. By using three polymorphic multiplexers, we can create a polymorphic Plus1-Plus7 circuit, which costs $80 + 3C_{pmpx}$ transistors (the circuits have the identical output for the lowest bit, so only 3 multiplexers are needed). The circuit could by used in a counter to adaptively change its behavior. Figure 4 shows the polymorphic Plus1-Plus7 circuit evolved from scratch. It utilizes 6 NAND/NOR, 3 XOR and 2 NOT gates, i.e. 70 transistors. While it is probably possible to optimize the circuit which uses the polymorphic multiplexers, the evolved solution will still be more compact and cheaper.

4.5 Polymorphic 2bit Multiplier vs. 4bit Sorting Network

The 4-bit sorting network sorts a 4-bit input vector and requires 18 gates (9 ANDs, 9 ORs). The 2-bit multiplier multiplies two 2-bit operands and produces a 4-bit output which costs 7 gates (5 ANDs, 2 XORs). The cost of the polymorphic 2bit multiplier vs. 4bit sorting network circuit, which utilizes four polymorphic multiplexers is $158+4C_{pmpx}$ transistors. The same functionality (evolved from scratch) requires 150 transistors.

5 Conclusions

Three approaches were considered for designing polymorphic digital circuits at the gate level. We do not have any conventional approach to create non-trivial polymorphic circuits. While the approach based on polymorphic multiplexers is scalable, it does not produce compact polymorphic circuits. The evolutionary approach is not scalable; however the evolved circuits are compact. Note that we have not performed any transistor-level optimizations for the modules that are switched by polymorphic multiplexers. The selection of suitable gates is important for a particular polymorphic circuit. We performed the selection intuitively before each experiment; next research will be conducted to use CGP to accomplish this task. In most cases we are not able to understand the topology of the evolved circuits since implementations of required behaviors are entangled. Anyway, the area of polymorphic electronics is quite new and a methodology has to be established in order to build real-world applications routinely.

Acknowledgment: The research was performed with the GAČR 102/04/0737 *Modern Methods of Digital Systems Design.*

References

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

[2] Stoica, A. et al.: Taking Evolutionary Circuit Design From Experimentation to Implementation: Some Useful Techniques and a Silicon Demonstration. IEE Proc.-Comp. Digit. Tech. Vol. 151(4), 2004, p. 295–300

[3] 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), Innsbruck, Austria 2002

[4] Stoica, A., Zebulum, R., Keymeulen, D.: Polymorphic Electronics. In Proc. of International Conference on Evolvable Systems: From Biology to Hardware, LNCS 2210, Springer Verlag, 2001, p. 291–302

[5] Thompson, A., Layzell, P., Zebulum, R., S.: Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution. IEEE Transactions on Evolutionary Computation. Vol 3(3), 1999, p. 167–196

[6] Wakerly, J.: Digital Design: Principles and Practices. Prentice-Hall, London 2000