Ročník 2017 electroscope Číslo I

Towards Design Flow for Space-Efficient Implementation of Polymorphic Circuits Based on Ambipolar Components

V. Simek[1], J. Nevoral[2], A. Crha[2], R. Ruzicka[1]
[1] IT4Innovations Centre of Excellence, Faculty of Information Technology, Brno University of Technology, Bozetechova 2, Brno, Czech Republic
[2] Department of Computer Systems, Faculty of Information Technology, Brno University of Technology, Bozetechova 2, Brno, Czech Republic
E-mail: simekv@fit.vutbr.cz, inevopral@fit.vutbr.cz, icrha@fit.vutbr.cz, ruzicka@fit.vutbr.cz

Anotace:
Hlavním cílem tohoto příspěvku představit ucelený návrhový postup, díky němuž je možno docílit efektivní implementace polymorfních obvodů. Především je zde využito technik na bázi evolučních algoritmů, které slouží k automatizovanému navrhování základních typů multifunkčních obvodových prvků (tj. logických hradel). V tomto případě se předpokládá uplatnění pokročilých materiálů či nanostruktur vykazujících tzv. ambipolární chování. Při návrhu vlastní struktury logických hradel je využito tranzistorů, u nichž lze řídit režim činnosti (tedy zda se chovají jako N- či P-kanálové prvky) řízením polarity napájecích větví. Bohužel konvenční návrhové metody a algoritmy není možné přímo využít pro efektivní návrh polymorfních obvodů, aniž by nebylo nutné se zabývat jejich podstatnou modifikací. Další z důležitých součástí prezentovaného způsobu návrhu vytváření polymorfních obvodů je tedy příslušná syntézní technika využívající specifických vlastností popisovaných multifunkčních hradel. Tento přístup k obvodové syntéze napomáhá dosažení prostorově efektivních výsledků zejména v případě komplexních polymorfních obvodů skládajících se ze stovek hradel. Klíčovým aspektem je v tomto případě využití principů Booleovského dělení a techniky tzv. kernellingu logických funkcí.

Abstract:
Main objective of this contribution is to present a unified design flow for an efficient implementation of polymorphic circuits. First of all, it employs an evolutionary inspired techniques that facilitates the creation of multifunctional circuit elements (i.e. logic gates) based on emerging materials and nano-structures exhibiting the ambipolar behavior. Those logic gates consists of individual transistors where the conduction mode (N- or P-channel) is controlled by switching the power rails. Unfortunately, conventional design methods and algorithms are not directly applicable for a design of polymorphic circuits without the need to face major changes. Hence the other important part of the suggested design flow is comprising the necessary circuit synthesis technique using those multifunctional logic gates. The presented circuit synthesis approach makes it feasible to achieve an area-efficient results in case of complex polymorphic circuit involving hundreds of gates. Its core is based on the utilization of Boolean division principles and function kernelling technique.

INTRODUCTION

The field of digital systems design has experienced an astonishingly vivid development throughout several previous decades. In fact, a closer look given to the technological side reveals their steadily growing complexity. The obvious challenge to keep pace with the growing functionality demands has resulted into the need to introduce large-scale heterogenous integration of miscellaneous physical features. However, as it clearly becomes apparent, the contemporary state-of-the-art conceptual paradigm, design approaches and fabrication procedures are inevitably getting closer to the ultimate edge depicted by the inherent technological constraints, which are naturally associated with the physical foundations of the widespread conventional CMOS process.

Substantial advancements reached along the way have gradually unlocked new directions and opportunities for digital circuits and systems implementation, and how to further increase their efficiency. However, the arrival of novel technologies also introduced in the same time a new set of obstacles to be surmounted. Researchers are facing more than ever before the necessity to grasp properly the benefits of those technological trends and turn them into an advantage when building digital logic gates, or other relevant circuit components.

It is assumed that a significant potential can be further unlocked thanks to the highly unorthodox computational or design approaches comprising e.g. reconfiguration of a digital circuit, exploitation of multifunctional circuit elements or even larger circuit structures taking an advantage of advanced emerging materials or nano-structures. A specific approach how to address these peculiar needs, at least in certain situations, could be also based upon the principles of so-called polymorphic or multifunctional electronics.

Polymorphic electronics is an approach that enables the design and implementation of multifunctional digital circuits **[1]**. Main idea behind the polymorphic electronics is connected with a circuit structure that is able to perform more than one intended function. It is

characteristic that the interconnection of the circuit components (gates) remains unchanged, which also represents a substantial contrast with the conventional electronics where the function must be explicitly selected at a given moment in time.
A closer look will reveal an important evidence that the approach based on polymorphic electronics paradigm carries one significant advantages in comparison with the conventional ways. In fact, that particular feature can be identified in its substantial technological independence. It means that the constantly emerging advanced materials or fabrication techniques might be considered at this place without major implications for the theoretical background of polymorphic electronics.
In recent years, various research activities have been focused on the utilization of so called post-silicon devices **[2]** for the construction of polymorphic circuit elements. It seems that some features of post-silicon devices, like the ambipolar charge carrier conductivity feature, may play an important role in the field of polymorphic electronics. The change of polymorphic gate function based on ambipolarity obviously leads to very efficient and neat implementation of logic gates, very close to the purity of ordinary CMOS logic gates. Parameters of such designed gates are also very promising, as the design is wholly digital – transistors operate as switches in the saturation mode.
Practical utilization of the polymorphic electronics concept is somewhat limited by the availability of suitable polymorphic logic gates. In the same time it becomes apparent that the aspects of designing polymorphic gates (and in particular more complex polymorphic circuits as such) are far from being a straightforward task for an ordinary human designer. This observation is true especially due to the fact that more than one function must be kept in mind while the structure of the gate is proposed. It is no wonder that most of the available polymorphic gates were created by means of using the evolutionary based design approaches. At the time of preparation of this paper, only three polymorphic gates based on ambipolar transistors were already reported in the literature.

Focus of the contribution

Main objective of this contribution is to present a unified design flow for an efficient implementation of polymorphic circuits. It employs an evolutionary inspired technique that facilitates the creation of multifunctional circuit elements (i.e. logic gates) based on emerging materials and nano-structures exhibiting the ambipolar behavior. Those logic gates consists of individual transistors where the conduction mode (N- or P-channel) is controlled by switching the power rails. In fact, the adoption of those reconfigurable transistors for construction of logic gates suggest the possibility to achieve a notable savings of the overall transistors count. Another advantage taking place within the context of the fabrication process is given by the fact that unlike the conventional CMOS technology (P-type MOS is mostly 2-3 times larger than its N-type MOS counterpart) all those transistors could be physically fabricated with the same dimensions and balanced switching characteristics.
The availability of multifunctional logic gates clearly opens a way towards the implementation of larger circuit structures. Unfortunately, conventional design methods and algorithms are not directly applicable for a design of polymorphic circuits without the need to face major changes. Therefore it is desirable to take into account this assumption for the design of a new, better and more efficient design methods of polymorphic circuit. Hence the other important part of the suggested design flow is comprising the necessary circuit synthesis technique using those multifunctional logic gates. The presented circuit synthesis approach makes it feasible to achieve an area-efficient results in case of complex polymorphic circuit involving hundreds of gates. Its core is based on the utilization of Boolean division principles and function kernelling technique.

PRINCIPLES OF AMBIPOLAR CONDUCTION

Among number of very interesting features associated with the emerging materials and nanoscale devices, especially the ambipolar conduction seems to be exposed to significant attention. The actual reason can be identified within the potential opportunity to enable a physical implementation of multifunctional circuits (these can be optionally referred to as reconfigurable) in a very efficient way.
From a technical point of view, fundamental principle behind the ambipolar mode of conduction is basically given by the mutual superposition of electron and hole currents. Physical devices built with this unique principle in mind offer an exceptional opportunity how to impose a direct control on electron-hole recombination taking place within the semiconductor channel. Ambipolar behavior that can provide both n- and p-channel performance in just a single device is very important due to its tremendous importance for manufacturing of complementary integrated circuits, where it basically eliminates the need to perform micropatterning of the individual p- and n-channel semiconductors. As a direct result of that, only a single type of an elementary switching device (let's say transistor) is sufficient in comparison with conventional CMOS fabrication technology.
Some of the advanced nanoscale devices provides transparent and reliable means how to take a precise control over this behavior and obtain significant benefits for digital-like circuits. Ambipolar mode of conduction has been already observed in many next-generation devices, e.g. comprising nanotubes, graphene, silicon nanowires, organic single crystals,

and organic semiconductor structures. As opposed to the unipolar silicon MOSFET device whose p-type or n-type behavior is unambiguously specified during fabrication, ambipolar devices can be switched from p-type to n-type, for example, by changing the gate bias intensity or drain-source polarity.

POLYMORPHIC ELECTRONICS

The purpose of this section is to provide a concise summary of the fundamental aspects relevant to the field of polymorphic electronics, which can be seen as a relatively new discipline in the field of electronic systems. In addition, several open problems are also specified in this context as well.

Within the domain of digital circuits and system, the notion of polymorphic electronics depicts a group of digital circuits that have the ability to perform more than one function, while the wiring of a given circuit remains still the same in all intended operating modes. This observation can be recognized as the most significant difference between polymorphic electronics and traditional approach to the realization of multifunctional circuits.

Selection of the corresponding function, which the circuit is going to execute, simply depends on the actual state of the target operating environment. Most importantly, the change of the polymorphic circuit function comes into the effect right away (without any eminent delay perceived) and sensitivity to the environments is naturally embedded into the circuit itself **[3]**.

It is important to point out that all the required circuit functions are designed intentionally, rather than, for example, as a fault condition caused by exceeding certain operating parameters of the circuit. The state of the environment can be accurately expressed through a physical quantity with a direct impact on the electrical properties of circuit building elements. Then, it is possible to clearly determine the actual function to be realized by that circuit according to the specific value of a relevant parameter **[1]**.

Such behaviour is useful for circuits that must adapt itself to unfriendly environment, e.g. by imposing restriction of power consumption **[4]** or heat dissipation **[5]** with preservation of essential functionality. Polymorphic electronics is also very beneficial for applications that are basically mono-functional, but need some additional feature. This might be helpful e.g. for embedded diagnostics **[6]**, security applications **[7]**, etc.

Open issues of polymorphic electronics

The field of polymorphic electronics, and especially the required multifunctional nature of its building components (e.g. logic gates, circuit blocks, etc.) which represent an important pillar of the whole paradigm, is surrounded with a number of still open problems that need to be addressed properly in order to successfully deploy this unconventional approach to digital circuit design, fully exploit its potential advantages and conceive practically feasible and efficient solution.

Some of the most important aspects, which deserve further attention in order to be resolved or further improved from the current level of advancement, are especially the following ones. The 1st one is the problem of an appropriate design methods for polymorphic circuits. One of the most common approaches of polymorphic circuits design is based on using some evolutionary methods. The 2nd issue is closely related to a search for convenient polymorphic components (gates). It is anticipated that especially the adoption of suitable emerging materials exhibiting so called ambipolar property may facilitate the implementation of space-efficient and reliable polymorphic gates.

Existing polymorphic gates

Polymorphic gate is described as an element which realizes elementary logic (boolean) function, whereas the function may vary in accordance with the particular state of the environment. It is possible to say that the function of the gate is controlled by environment. Such feature may be useful for variety of applications, may save chip area as well in terms of a total transistors count and, in the same time, reduce global interconnections significantly. If the gate exhibit e.g. NAND function for some range of the power supply voltage (V_{dd}) and e.g. NOR function for another range of the V_{dd}, the gate could be specified as a NAND/NOR gate controlled by V_{dd}. It is assumed that polymorphic gate may perform no more than one function with respect to any particular instant during the course of time.

Table 1 surveys the polymorphic gates reported in literature. For each polymorphic gate, the logic functions performed by the gate are given together with recommended setting of the control signal variable. The number of transistors characterizes the size of polymorphic gates only partially (transistors occupy different areas, gates were fabricated using different fabrication technology).

Tab. 1: A survey of existing CMOS-based polymorphic gates.

Gate	Cntrl.	Cntrl. Type	Transistors
NAND/NOR	3.3/1.8 V	V_{dd}	6
AND/OR	1.2/3.3 V	V_{dd}	8
NAND/NOR	5/3.3 V	V_{dd}	8
AND/OR	27/125 C	temp.	6
AND/OR	5/90 C	temp.	8
NAND/NOR	0/5 V	ext. V	10
NAND/NOR	5/0 V	ext. V	8
NAND/NOR	5/0 V	ext. V	10
NAND/XOR	5/0 V	ext. V	9
AND/OR	0/3.3 V	ext. V	6
AND/OR/XOR	3.3/1.5/0 V	ext. V	9
NAND/NOR	0/5 V	ext. V	10

Only two of the polymorphic gates have been physically fabricated so far; remaining polymorphic gates were either simulated or tested in a FPTA **[8]**. For instance, the 6-transistor NAND/NOR gate

controlled by Vdd was fabricated in a 0.5-micron HP technology **[9]**. Another NAND/NOR gate controlled by Vdd and introduced in **[10]** was utilized in the REPOMO chip **[3]**. Internal electrical interconnections of the designed gate are depicted on a transistor level in Figure 1 below. The gate was designed with the aim to achieve properties and criteria defined in the previous section.

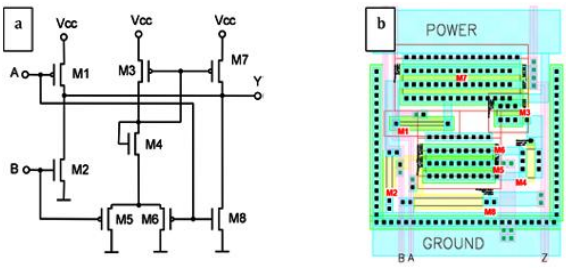

Fig. 1: Figure 3. Internal structure of polymorphic NAND/NOR gate on a transistor level (a) and its corresponding physical layout (b) using standard CMOS AMIS 0.7 um technological library. The resulting size of a single gate (b) is approximately 55.8 um x 68.2 um.

EVOLUTIONARY DESIGN OF POLYMORPHIC LOGIC GATES

Utilization of the polymorphic electronics concept, and therefore construction of more complex circuit arrangements, is somewhat limited by the availability of suitable polymorphic gates. In fact, the existence of several polymorphic gates employing so called ambipolar transistors has been already reported, e.g. in **[11]**. However, no systematic exploration of automated methods was conceived yet. Moreover, only a few papers were devoted to the application of evolutionary-inspired methods for the construction of small-scale digital circuits directly at a transistor level, which are utilizing in most cases standard p-MOS and n-MOS transistor devices.

In this section, key features behind the evolutionary method used for the design of polymorphic gates controlled by switching the power rails on a transistor level are outlined. In this case, four-terminal transistor with the ambipolar behavior are considered **[12]**. The approach reflects a different functionality of the ambipolar transistors and utilizes a novel view on circuit representation and simulation.

In order to verify the operation of a given circuit structure through the simulation performed in analog domain within a reasonable amount of time, i.e. the interconnection of individual transistors inside a polymorphic logic gate, which was obtained by means of using an evolutionary-based algorithm, Mrazek and Vasicek proposed a discrete simulator with a switch-level transistor model extended by a threshold drop degradation effect to achieve a fast simulation with convenient trade-off between accuracy and the overall time required for circuit evaluation **[13]**.

Circuit representation

In order to perform evolution of polymorphic circuits at the transistor level, a suitable representation that allows to encode the bidirectional graph structures containing junctions is needed. The method of choice utilized in the case is generally known as a Cartesian genetic programming (CGP), which was proposed by J. Miller **[14]**.

The circuit representation is derived from the CGP representation of gate-level circuit. Each polymorphic digital circuit is represented using an array of nodes and can be encoded by fixed length array of integers. Each node consists of three source terminals and one output terminal and can act as an ambipolar transistor or a junction. Ambipolar transistor uses all three source terminals, whereas the junction nodes two source terminals only. The utilized nodes are shown in Figure 2 below. Source terminals of each node can be independently connected to the output terminal of any node placed in previous columns or to one of the primary circuit inputs.

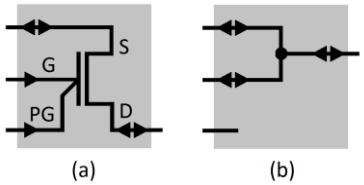

Fig. 2: Basic building blocks of transistor-level circuits: (a) ambipolar transistor and (b) junction. The arrows denote the possible directions of signal flow which have to be considered during the evaluation

The junction nodes combine two input signals and one output signal together. As a consequence of that, loops and multiple connections are natively supported.

Figure 3 demonstrates the utilized representation of a polymorphic inverter circuit which inverts the logical value of the input signal independently of power rails switching. The shown representation encodes a candidate circuit using four nodes. However, only three of them contribute to the phenotype and are active.

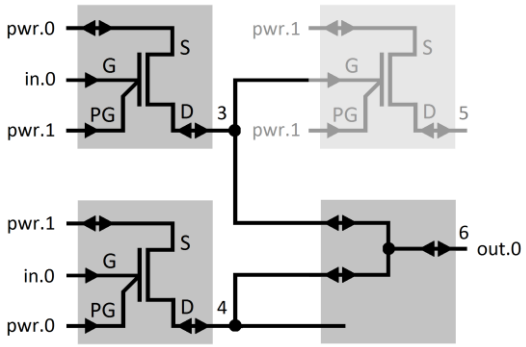

Fig. 3: Example of a candidate circuit implementing polymorphic inverter using two ambipolar transistors. Circuit has one input (in.0), one output signal (out.0) and two power rails (pwr.0, pwr.1).

Evaluation of the candidate solutions

The goal of the evaluation is to determine whether the candidate circuit meets the requirements, i.e. there is no violation of the specified constraints and the circuit itself is working correctly with regard to its functional definition. In fact, evaluation of the candidate solutions consists of two steps.

Firstly, set of active nodes is determined. This operation is performed due to speed optimization and because of skipping the short circuits in the unused part of the circuit. Only the active nodes represent (i.e. they are in a path from input nodes to outputs) the evaluated circuit. Inactive nodes are ignored.

Then, multi-level discrete event-driven simulator is utilized to determine the circuit response for each input signal combination. The advantage of this approach is that only necessary nodes are updated if there is a change of a value.

The research of ambipolar transistors started a few years ago. As a consequence of that, no exact models are still available. Therefore, we created several discrete models of ambipolar transistors with six discrete voltage levels according to the expected behavior, which are utilizes in the circuit simulation.

Search strategy

As a search algorithm, $1 + \Lambda$ evolutionary strategy is utilized **[14]**. The initial population is generated in a random manner. Every new population consists of the best individual and several offspring created using a point mutation operator which modifies randomly selected genes. The evolution is terminated when a predefined number of generations is reached.

Quality of each candidate solution is determined by the fitness function, which calculates the difference between expected circuits outputs and outputs delivered by the discrete simulation. Moreover, if a given simulation run exceeds the predefined number of steps or the occurrence of short-circuit is identified for some input signal voltage combination, a penalty is subtracted from the total fitness value.

As soon as a fully working solution is found, additional requirements (like e.g. high input impedance and low output impedance) for circuit properties are checked and the circuit is optimized to reduce the transistor count starts whereas quality of circuit outputs remains unchanged.

Evolved gates

An extensive library of building components for digital circuits based on ambipolar transistors has been already reported **[15]**. However, only three polymorphic circuit blocks utilizing these transistors and simultaneously controlled by switching the power rails were physically designed. Yang et al. presented NAND/NOR and XOR/XNOR gates **[11]**. Moreover, polymorphic inverter (labelled as NOT/NOT), which involves two ambipolar transistors connected similarly to classic MOS inverter, is generally well known.

Both XOR/XNOR and NAND/NOR mentioned gates use 4 transistors only. However, the first one expects the presence of both input signal negation. Therefore, in order to assemble the gate, 4 ambipolar transistors and 2 polymorphic inverters are needed (i.e. 8 ambipolar transistors in total).

In order to design new polymorphic gates based on ambipolar transistors, where their behavior is controlled by switching the power rails, evolutionary approach described above was utilized. Our goal was to design a set of polymorphic gates where each of them exhibits full voltage swing on the outputs.

Tab. 2: Size of the smallest solutions of selected polymorphic gates in number of ambipolar transistor being used

	W/O impedance constraints	**High input impedance & Low output impedance**
NOT/NOT	2	2
NAND/NOR	4	4
AND/OR	3	4
XOR/XNOR	4	5

All the evolved gates consist of equal or, in most cases, less ambipolar transistors compared to the currently known best circuits. Four transistors are needed to design the XOR/XNOR gate and even just three transistors are needed to design the AND/OR polymorphic gate. Table 2 summarizes the minimal transistor count for chosen evolved polymorphic gates. As an example, Figure 4 shows the AND/OR and XOR/XNOR polymorphic circuit gates with high input and low output impedance. Signal inputs of those gates are marked as in.0 and in.1, gate output as out and power rails are finally denoted as pwr0 and pwr1.

The electrical behavior of all the designed gates was subject to further analysis using HSPICE circuit-level simulator. As it was mentioned before, there do not exist any freely available, HSPICE compatible models applicable for simulation of behavior in case of four-terminal ambipolar transistors. Therefore, ambipolar behavior was emulated by a circuit composed of two MOSFET transistors, two transmission gates and one inverter. All the circuits were valid and operated correctly. Figure 5 shows the HSPICE simulation results of the gate depicted in Figure 4. Function of the polymorphic gates is changed every 40 ns – i.e. when the voltages on power input signals are switched.

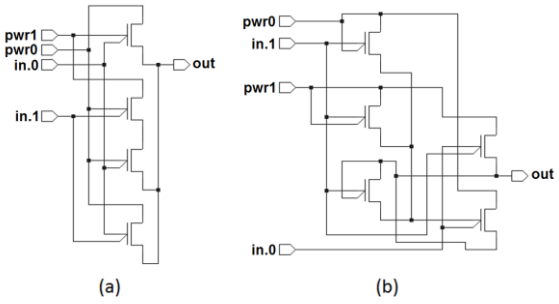

Fig. 4: AND/OR (a) and XOR/XNOR (b) polymorphic gates

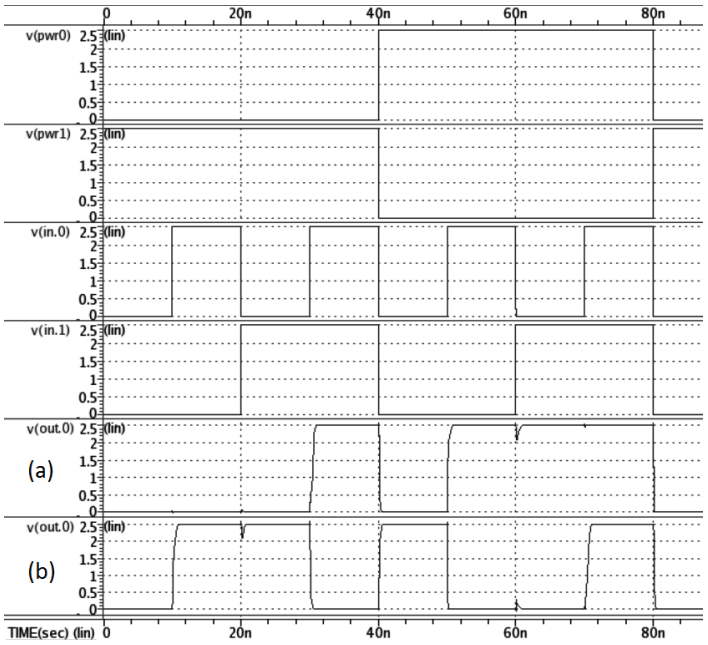

Fig. 5: AND/OR (a) and XOR/XNOR (b) polymorphic gates with high input and low output impedance

REVIEW OF POLYMORPHIC CIRCUITS SYNTHESIS METHODS

Synthesis methods of ordinary digital circuits have to solve the problem of finding interconnection graph G between individual functional elements (e.g. gates) which altogether make up just one particular function F. If a suitable canonical form of F is found, the structure of G can be easily inferred from it. For polymorphic circuits, this approach tends to exhibit higher complexity because just one graph needs to cover already several functions from the existing set $\Phi = \{F_1,..., F_n\}$, which makes up the given circuit and fulfil the demand of multifunctional operation. The task to find the same form for all the functions F_1 to F_n (with different elementary functions on the same position) is, therefore, not so trivial at all.

Nowadays, polymorphic circuits design is currently performed at a gate-level, while the individual gates are constructed at a transistors-level. Throughout wide range of experiments with the design of polymorphic circuits it became clear that design of circuits composed solely from polymorphic gates is less suitable. It seems appropriate to propose polymorphic circuits containing both polymorphic and static gates (the same behavior in both modes). It should be noted at this point that a number of static gates typically exceeds the number of polymorphic gates of developed circuit. In many cases it is also sufficient to use a single type of polymorphic gate only. This observation appears valid especially in case of the logic gate which implements logically complete behavior (e.g. NAND / NOR). If a wider set of polymorphic gates is used, it could eventually lead to more efficient solution. Nevertheless the level of complexity in case of a circuit design and synthesis would be inevitably facing a problem related to a significant state-space growth **[16]**.

Let us also note that polymorphic circuit synthesis methods do not aim at dealing with the question of environment intentionally and how it is physically involved in the circuit operation. This is the subject delegated to the chosen and employed polymorphic gates – building components of the circuit. Simple circuits could be obviously designed by hand but the growing complexity renders this approach virtually unfeasible. Proper synthesis techniques have to be obviously considered. As it turns out, direct usage of various conventional optimization methods targeted at the ordinary digital circuits fails to yield adequate results in this specific situation. Hence an alternative approach needs to be considered, e.g. the exploitation of evolutionary optimization methods, which might bring the solution **[17]**, **[18]**.

Selected evolutionary methods

Digital circuit synthesis and optimization techniques based on the exploitation of convenient evolutionary-inspired paradigms, as demonstrated by Sekanina **[19]** (and before initially suggested by Miller **[14]**, Koza **[20]** and Thompson **[21]**), could establish a way how to achieve a rather unconventional but, at the same time, interesting and useful solution. Needless to say, also the original concept of polymorphic electronics emerged virtually as a side effect of evolutionary design experiments **[1]**. Almost all polymorphic circuits, more complex than just a few gates, have been designed using Carthesian Genetic Programming (CGP) **[14]** till now.

In terms of CGP, the circuit structure is laid out as an array of **u** (columns) × **v** (rows) of programmable elements (gates). The number of circuit inputs, n_i, and outputs, n_o, is fixed and no feedback is allowed. Each

gate is programmed to perform one of the functions defined at the beginning of the experiment. The fitness function is constructed to minimize the Hamming distance between the output vectors of a candidate circuit and the required output vectors. Typically, all possible input vectors are applied to obtain the set of output vectors for the two required functions F_1 and F_2.

Selected conventional methods

One of the first examples of conventional design methods focused on polymorphic circuits was introduced by Gajda **[16]**. The first of these methods involves the so-called polymorphic multiplexing. This approach falls on the borderline between conventional and polymorphic digital circuits. For each function, a digital circuit is synthesized and the outputs of these circuits are then multiplexed by a polymorphic multiplexor. The structure of a circuit designed by this method shows a relatively low optimality. However, possible workaround towards the desirable improvement dwells in the partial sharing of some logic resources.
In addition to that, Gajda **[16]** proposed a method of polymorphic circuit synthesis utilizing binary decision diagrams (BDD). The method is called PolyBDD. Its core part is using Multi-terminal BDD (MTBDD), which is an extension of binary decision diagrams. For desired functions F_1 and F_2, a MTBDD is created. Then the MTBDD is converted into a circuit, where the nodes assume the role of multiplexers and the terminals are replaced by a proper polymorphic sub-circuit according to the number in a given leaf.

PROPOSED SYNTHESIS METHOD

Designing polymorphic circuits is undoubtedly a very difficult task. A designer must take into account two different digital circuits at the same time and advisedly design them to share common parts with aim to save resources, i.e. gates. Small amount of articles, relatively new technology and limited information resources about polymorphic synthesis confirm this fact. There were a few attempts to design polymorphic circuits, but most of them were at least partially suffering with various drawbacks.
The easiest method how to build a polymorphic circuit is to synthesize two different circuits by means of using conventional logic synthesis techniques and then switch their output with a polymorphic multiplexer element accordingly. An output function will be changed by environment state due to the polymorphic nature of a multiplexer being used, but sharing of resources is not met at this point. It is obvious that from the perspective of resource savings there is no improvement achieved at all **[16]**.
Due to all of these weaknesses mentioned in previous paragraphs, main target is to develop a synthesis methodology of polymorphic circuits which puts polymorphic gates directly inside the circuit. It requires a well-controlled design from beginning to the last stage of polymorphic synthesis. That is a reason why the design of the proposed method of polymorphic circuits design was designed completely from the scratch.

Principles of the synthesis method

The main idea behind the novel approach is based on the undeniable identification of common parts across the input circuits which are virtually shared between them as so-called common divisors by means of exploiting techniques of function kernelling **[21]**, **[23]** and Boolean division **[22]**.
The input for the proposed synthesis methodology is represented by specification of two different circuits – F1 and F2, see (1), (2) below. Their minimized notations are provided in DNF representation (Disjunctive Normal Form). Each function is further processed by the synthesis tool as a truth table in two-level PLA format.

$$F_1 = ab\bar{d} + b\bar{c}\bar{d} + \bar{b}\bar{c}d + a\bar{c} + \bar{a}\bar{b}c\bar{d} \quad (1)$$
$$F_2 = ab\bar{c} + a\bar{c}d + \bar{a}\bar{b} + \bar{b}c\bar{d} + \bar{a}c \quad (2)$$

From this starting point an intersection table of Sum-Of-Products of each circuit is derived. See table 3 for example of intersection table. A vertical line of a table is filled by SOP of a first circuit, a horizontal line of table is filled by SOP of a second circuit. Each cell fills the intersection of SOPs corresponding to row and column. When the table is completely filled in, it is possible to continue with a next step.
Then the first pass through the completed table is performed. The purpose is to identify those boxes that exhibit the mutual intersection of a maximum size, e.g. *minterm (1 | 1)*. The first minterm to be successfully recognized is then put at its place into the final expression. These minterms are basically common for both input functions and, thus, it is not required to deal with them in a polymorphic way. Once the minterm is registered in the final expression, corresponding row and column are eliminated from the table.
Next, the second pass through the table 3 is commenced. This time, the task is to find the largest intersection. The box fulfilling this requirement is then rewritten into the final expression, the whole row and column with this particular box are eliminated from the table. However, it is important not to put aside the remaining literals which are specific for the first and second function alternatively. These literals will be isolated by suitable polymorphic element.
A polymorphic multiplexer **[24]** and polymorphic inverter are intended to be used at this place. When the difference between literals is originating only from the negation, the polymorphic inverter is used, otherwise when the rest of literals are actually distinguished in literal names, a polymorphic multiplexer is applied. Now, the necessary step to be

taken is to cross out the column and row in the table, because these product terms are already covered.
Algorithm itself continues with an iterative walk through the table and is trying to find a maximum intersection until coverage of the whole table is effectively achieved. However, it is not as easy task as it might look like on a first sight. There exist a number of specific situations which need a particular attention:

1) No intersection: When no intersection is found and any column and row rests, hard 1 is used as the intersection.
   Example: $f = 1(f_1 literals \mid f_2 literals)$.

2) Different number of product terms: When this situation occurs, is possible to deploy one polymorphic multiplexer switching between hard 1 and rest product terms of a function.
   Example: $f = 1(1 \mid f_2 product\ term_1 + \ldots + product\_term_n)$.

The algorithm concept is essential, however an automated tool performing this algorithm is very indispensable. That fact has resulted into the creation of specific software tool for the suggested methodology, which is briefly discussed in the following section.

Notes on software tool

With regards to the basis of the methodology discussed in the previous section, there has been prepared a software tool performing an automated synthesis of two polymorphic circuits defined by PLA input files. A main purpose of creating the synthesis tool is a substantial automation of the whole procedure. In fact, it is perfectly feasible to synthesize small circuit ”on the paper”, but more complex circuits require a considerable level of automation. The synthesis software tools is console application, a GUI is not necessary for this purposes.
The synthesis tool itself has been divided into three parts. The first part performs loading of a PLA file(s), second part is responsible for the polymorphic synthesis based on identification of common parts among the product terms and the third part finally collects the statistic data and prepares their output for further analysis or visualisation.
At first, names of input and output files are given. Then the PLA file type check is carried out with the specified files. Then, next step involves loading of the PLA data to a special internal structure of the synthesis tool which forms a table, the table of intersections. Each cell in a row or column is based on unsigned integer type that means a one product term is represented by one unsigned integer. This solution allows very fast and bit-wise operations on the intersection table.
As soon as the intersection table is created from the input PLA file, the tool can proceed with the synthesis. A main task of this part is going through the intersection table, searching for the maximum intersection between two different circuits and generating output formula describing a target polymorphic circuit. This particular step is executed until the intersection table is fully covered. It is important to notice that all operations with table are bit-wise, so it significantly contributes to the overall efficiency. This part also solves special cases like no intersection and different number of product terms. During this process, a statistics are gathered into the statistic data structure.

EXPERIMENTAL RESULTS

The proposed software synthesis tool for a design of polymorphic circuits has been tested on several real circuits defined by a truth table in two-level PLA format. Circuits have been chosen with respect to the same number of input for one synthesis run. All circuits are taken from MCNC benchmarks.
Basic specification of these circuits can be found in table 4 below. A circuits in the table have original names with brackets notation. Letters in brackets denotes which outputs are synthesized (all noted letters) and capital letters tell us which output is active while circuit works in mode one, or in mode two respectively. When there are no brackets, two different circuits are synthesized and polymorphism is responsible for switching between circuit function one or circuit function two.
Finally, results provided by polymorphic synthesis tool are shown and compared with results from conventional synthesis tool SIS **[25]**. With the aim of straightforward comparison, all circuits were built from two input gates only. The only exception in this context is an inverter. We have chosen a number of actually deployed two-input gates as the main parameter for comparison. Percentage improvement over the conventional solution is noted in the last column of the table 4 as the number of used gates in polymorphic solution versus convectional solution.

CONCLUSIONS

A unified design flow for an efficient implementation of polymorphic circuits was presented in this contribution. In order to design unique polymorphic gates an evolutionary approach based on Cartesian genetic programming was utilized. The purpose of the evolutionary algorithm was to effort to achieve minimization of the overall number of transistors being used for each one of the evolved gates. Those logic gates consisted of individual transistors where the conduction mode (N- or P-channel) was controlled by switching the power rails. The achieved results clearly shows significant transistor savings compared to the currently best known conventional logic gates.
The evolved set contains polymorphic gates with high input impedance and low output impedance as well as

various discrete switch-level ambipolar transistor models extended by taking into account the threshold voltage drop degradation effect were used. Functionality of the proposed gates was verified by HSPICE simulation. It appears that those gates bring a significant advantage for space-efficient synthesis of polymorphic circuits and suggest the opportunity how to considerably reduce the target size of complex polymorphic circuits.

Unfortunately, conventional design methods and algorithms are not directly applicable for a design of polymorphic circuits without the need to face major changes. Hence the other important part of the suggested design flow included the necessary circuit synthesis technique using those multifunctional logic gates. Its core is based on the utilization of Boolean division principles and function kernelling technique. A set of real experiments with complex circuits, where in the one case it was possible to achieve almost 40% gates saving to our previous results, was performed in order to evaluate the proposed synthesis tool. Then, an average improvement on real benchmark MCNC circuits is about 20%.

In order to further increase the synthesis efficiency of polymorphic circuits further steps will explore, for example, the applicability of AIG graphs and structural hashing for better identification of circuit parts that can be shared between two (or even more) functions subjected to the synthesis process. Potential advantage could be also exploited in connection with more efficient circuit elements (e.g. logic gates) based on so called ambipolar transistors created using silicon nanowires hetero-structures or by means of using the hybrid integration of silicon-based chip structure with deposition of ambipolar semiconductor material for the active channel layer of a transistor.

ACKNOWLEDGEMENTS

This work was generously supported by the national COST grant Unconventional Design Techniques for Intrinsic Reconfiguration of Digital Circuits: From Materials to Implementation (no. LD14055). Another support was also provided by The Ministry of Education, Youth and Sports of the Czech Republic from the National Program of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602.

REFERENCES

[1] Stoica, A., Zebulum, R., Keymeulen, D.: Polymorphic electronics. In: Proceedings of Evolvable Systems: From Biology to Hardware Conference, vol. 2210 of LNCS. Berlin-Heidelberg: Springer, 2001, pp. 291–302.

[2] Rabaey, J. M., Malik, S.: Challenges and solutions for late- and post-silicon design. In: IEEE Design Test of Computers, vol. 25, no. 4, pp. 296–302, July 2008.

[3] Sekanina, L., Růžička, R., Vašíček, Z., Prokop, R., Fujcik, L.: REPOMO32 - New Reconfigurable Polymorphic Integrated Circuit for Adaptive Hardware. Proc. of the 2009 IEEE Symposium Series on Computational Intelligence - Workshop on Evolvable and Adaptive Hardware, Nashville, IEEE CIS, 2009, pp. 39 – 46.

[4] Růžička, R.: Gracefully Degrading Circuit Controllers Based on Polytronics. Proc. of 13th Euromicro Conference on Digital System Design, IEEE CS, 2010, pp. 809 – 812.

[5] Ruzicka, R., Simek, V.: Chip temperature selfregulation for digital circuits using polymorphic electronics principles. In: Proc. of 14th Euromicro Conference on Digital System Design. Institute of Electrical and Electronics Engineers, 2011, pp. 205–212.

[6] Sekanina, L., Stareček, L., Kotásek, Z., Gajda, Z.: Polymorphic Gates in Design and Test of Digital Circuits. International Journal of Unconventional Computing, 4(2), 2008, Philadelphia, pp. 125 – 142, ISSN 1548-7199.

[7] Sekanina, L., Růžička, R., Vašíček, Z., Šimek, V., Hanáček, P.: Implementing a Unique Chip ID on a Reconfigurable Polymorphic Circuit, In: Information Technology And Control, 42(1), 2013, pp. 7-14.

[8] Zebulum, R., Stoica, A., Keymeulen, D.: A Flexible Model of a CMOS Field Programmable Transistor Array Targeted for Hardware Evolution. In: Third Int. Conference on Evolvable Systems: From Biology to Hardware (ICES2000), Edinburgh, 2000, pp. 274 – 283.

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

[10] Růžička, R., Sekanina, L., Prokop, R.: Physical demonstration of Polymorphic Self-checking Circuits. Proc. of the 14th IEEE IOLTS, IEEE CS, 2008, pp. 31 – 36.

[11] Yang, X., Mohanram, K.: Ambipolar electronics, 2010.

[12] Harada, N. et al.: A polarity-controllable graphene inverter. In: Applied Physics Letters, 96(1): 012102 -012102-3, 2010.

[13] Mrazek, V., Vasicek, Z.: Evolutionary design of transistor level digital circuits using discrete simulation. In: Proc. of European Conf. on Genetic Programming, vol. 9025 of LNCS. Berlin-Heidelberg: Springer, 2015, pp. 66–77.

[14] Miller, J., Thomson, P.: Cartesian Genetic Programming. In: Proc. of the 3rd European Conference on Genetic Programming, EuroGP 2000, vol. 1802 of LNCS. Berlin-Heidelberg: Springer, 2000, pp. 121-132. ISSN 0302-9743.

[15] Ben-Jamaa, M. H., Mohanram, K., Micheli, G. D.: An efficient gate library for ambipolar CNTFET logic. In: IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, 2011, vol. 30, no. 2, pp. 242–255.

[16] Gajda, Z., Sekanina, L.: On Evolutionary Synthesis of Compact Polymorphic Combinational Circuits, In: Journal of Multiple-Valued Logic and Soft Computing, vol. 17, no. 6, 2011, Philadelphia, US, pp. 607-631.

[17] Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York, Oxford, 1996.

[18] Gajda, Z., Sekanina, L.: Reducing the Number of Transistors in Digital Circuits Using Gate-Level Evolutionary Design. 2007 Genetic and Evolutionary Computation Conference, New York, ACM, 2007, pp. 245 – 252.

[19] Sekanina, L.: Ubiquity symposium: Evolutionary computation and the processes of life: evolutionary computation in physical world. Ubiquity. 2013, vol. 2013, no. 2, pp. 1-7. ISSN 1530-2180.

[20] Koza, J. R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992, 840 p., ISBN 0-262-11170-5.

[21] Thompson, A.: Silicon Evolution. Proc. of Genetic Programming, GP'96, MIT Press, 1996, pp. 444 – 452.

[22] Hachtel, G. D., Somenzi, F.: Logic Synthesis and Verification Algorithms, Kluwer Academic Pub, 1996, 564 p.

[23] Brayton, R. K., McMullen, C.: The Decomposition and Factorization of Boolean Expressions, ISCAS Proceedings, April 1982.

[24] Crha, A., Ruzicka, R., Simek, V.: On the Synthesis of Multifunctional Logic Circuits. In: Abstracts Proceedings of International FLASH Conference. Brno: Fakulta elektrotechniky a komunikačních technologií VUT v Brně, 2015, pp. 52-53. ISBN 978-80-214-5270-1.

[25] Ellen, S., Luciano, L., Alexander, S., Robert, K. B.: SIS: A System for Sequential Circuit Synthesis. In: Electronics Research Laboratory Memorandum No. UCBERL M92/41, 1992.

Tab. 3: Intersection table for two input functions F1 and F2.

F_1 \ F_2	$ab\bar{c}$	$a\bar{c}d$	$\bar{a}\bar{b}$	$\bar{b}c\bar{d}$	$\bar{a}c$
abd	$ab(d\|\bar{c})$	$a(bd\|\bar{c}d)$	$\emptyset$	$\bar{d}(ab\|\bar{b}c)$	$\emptyset$
$b\bar{c}d$	$b\bar{c}(d\|a)$	$\bar{c}(bd\|ad)$	$\emptyset$	$\bar{d}(b\bar{c}\|\bar{b}c)$	$\emptyset$
$\bar{b}\bar{c}d$	$\bar{c}(bd\|ab)$	$\bar{c}d(b\|a)$	$\bar{b}(\bar{c}d\|\bar{a})$	$\bar{b}(\bar{c}d\|cd)$	$\emptyset$
$a\bar{c}$	$a\bar{c}(1\|b)$	$a\bar{c}(1\|d)$	$\emptyset$	$\emptyset$	$\emptyset$
$\bar{a}\bar{b}cd$	$\emptyset$	$\emptyset$	$\bar{a}\bar{b}(cd\|1)$	$\bar{b}cd(\bar{a}\|1)$	$\bar{a}c(\bar{b}d\|1)$

Tab. 4: Comparison of the results achieved with the proposed synthesis flow and conventional SIS tool applied on a set of test circuits.

no.			Polymoprhic synthesis tool						SIS 1.3.6 tool				Improvement
#	Circuit 1	Circuit 2	INV	2-AND	2-OR	P-MUX	P-INV	SUM	INV	2-AND	2-OR	SUM	[%]
1	rd84.pla (W)	rd84.pla (X)	8	896	92	93	8	1097	10	218	1232	1460	24.86
2	rd84.pla (X)	rd84.pla (Y)	8	896	1	1	2	908	9	518	512	1039	12.61
3	rd84.pla (W)	rd84.pla (Z)	8	600	91	72	0	771	10	364	574	948	18.67
4	rd84.pla (WxYz)	rd84.pla (wXyZ)	8	1448	92	75	10	1633	11	509	1302	1822	10.37
5	newtpla1.pla-X	newtpla1-Y.pla	4	15	1	1	1	22	5	26	5	36	38.89
6	9sym.pla	Z9sym.pla	9	447	85	77	11	629	9	860	85	954	34.07
7	t481.pla	ryy6.pla	16	4293	112	113	9	4543	17	4382	992	5391	15.73
8	ryy6.pla	newtag.pla	5	34	7	7	5	58	10	39	19	68	14.71
9	max46.pla	9sym.pla	9	549	46	47	8	659	18	779	130	927	28.91
10	rd73.pla	sqn.pla	7	749	65	49	9	879	19	293	669	981	10.40
11	sao2.pla (xWYz)	sao2.pla (XwyZ)	10	333	49	33	8	433	14	258	178	450	3.78