Designing Polymorphic Circuits with Periodical Weight Adjustment

Houjun Liang[1], Rui Xie[2]
[1]Department of Computer Science, [2]Department of Accounting
Anhui University of Finance and Economics
Bengbu, Anhui, China
lhjabc45@sina.com

Liang Chen
Department of Transportation Service
Automobile NCO Academy
Bengbu, Anhui, China
aufecl1@163.com

***Abstract*—A polymorphic circuit can perform two or more functions under different conditions without the need of extra components. Those functions can be activated by environmental signals, such as temperature, power supply voltage, illumination, and so on. So far, existent evolutionary algorithms can only evolve polymorphic circuits at lower success rate. In this paper, the periodical weight adjustment method is proposed for the Evolutionary Strategy to evolve polymorphic circuits. The experimental results demonstrate that the periodical weight adjustment method can perform better than existent evolutionary techniques for polymorphic circuits, and it has the potential to evolve more complex polymorphic circuits, e.g., by this method a three-state polymorphic circuit is evolved for the first time.**

I. Introduction

The concept of polymorphic electronics was proposed by Stoica et al [1]. Unlike traditional multifunctional electronics, which are based on switches or multiplexers, the polymorphic circuit does not need any extra components to switch its different functions, but by some external factors, such as power supply voltage, temperature and light [2-4].

The Evolutionary Algorithms (EAs) for designing the polymorphic circuits are firstly proposed by Sekanina and his colleagues [5-7]. The advantages of evolutionary design approaches are: i) the circuits designed by the evolutionary design approach often need fewer gates. ii) The evolutionary design approach can find novel and creative topologies for the circuits.

Gajda and Sekanina have done a lot of important work on designing polymorphic circuits with evolutionary algorithms [5, 8, 9]. Especially in [8], two-state polymorphic circuits with 8-inputs/8-outputs and 13-inputs/1-output have being evolved with the aid of the binary decision diagrams (BDD) and multiplexer. However, designing polymorphic circuits purely by evolutionary algorithms is still a hard problem. In [10], an improved Evolutionary Strategy (ES) based on weighted sum method was proposed to design polymorphic circuits. Nevertheless, this improved algorithm need to set weight factors manually, and the success rates are still relatively low. So, more effective design method should be developed.

Because a polymorphic circuit has two or more different functions, it is necessary to balance the search preferences among different functions. The objective of the periodical weight adjustment method in this paper is to balance the search preference among different functions automatically. The experimental results demonstrate that the success rates are improved. The ES based on periodical weight adjustment can evolve polymorphic circuits more effectively and has the potential to evolve more complex polymorphic circuits, e.g. three-state polymorphic circuits.

This paper proposes a method for designing polymorphic circuits by periodically adjusting weights, the rest is is organized as follows. Section II introduces some related work. Section III considers the existent problems. Section IV describes the Periodical Weight Adjustment Method (PWAM) in detail. Section V gives the experimental results. Finally, a brief conclusion is given in section VI.

II. Related Work

A. Cartesian Genetic Programming

The evolutionary algorithm adopted in this paper is based on Cartesian Genetic Programming (CGP) [11-14], which was originally introduced into Evolvable Hardware (EHW) field by Miller et al. [15]. CGP has already demonstrated its efficiency and is widely used for the evolutionary design of digital circuits [8, 9, 14]. The CGP model is described as follows.

- *Representation*: CGP uses an array of c (columns) × r (rows) of Programmable Elements (PEs) to represent the desired circuit. The programmable elements can be traditional primary logic gates (e.g. AND, OR, NOT), or polymorphic gates (e.g. AND/OR, AND/AND), or high-level functional units. If r=1, it means the number of candidate PEs is same with c.
- *Connection*: Each Programmable Element has its own inputs and outputs. The inputs of each PE can only be

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.212

connected to the previous columns or the circuit inputs, but cannot be connected to the latter columns. The interconnectivity of each PE is defined by the levels back parameter L, which determines how many previous columns can be connected to the current one. In this paper, $L = c$, and this means that all previous columns can be connected. It is noted that the PEs in the first column can only connect to the primary inputs of the circuit.

In order that the expected circuit can be evolved under the control of EAs, the candidate circuits must be mapped into chromosomes. A chromosome is a string of parameters (genes) that are joined together. It represents a potential solution to a given problem and defines the topology of a candidate circuit. Although the CGP model is often adopted in evolvable hardware community, the exact encoding methods are often with minor differences. The encoding method used in this paper is exactly the same as that in [10]. The initial population is generated randomly.

B. Evolutionary Algorithms and Existent Results

The evolutionary algorithm chosen for designing polymorphic circuits is (μ, λ)-ES [5, 10]. In this algorithm, each individual of the population represents a potential solution to the desired polymorphic circuit.

Generally, the procedure of the fitness evaluation of an individual can be described as that in Fig. 1. For a circuit in the ith mode with n_i inputs and n_o outputs, the maximum fitness value of Fit_i is $2^{n_i} \times n_o$. When both weight factors and are set to 1, it means the circuit in each mode is treated with equal emphasis. For convenience, this kind of ES is called as Conventional ES (CES). Using this method, several kinds of polymorphic circuits using polymorphic gates as primary building blocks were evolved [5]. However, the success rate of evolutionary design of polymorphic circuits with CES is very low. In [10], a Weighted Sum Method (WSM) is proposed to improve the evolutionary process, where w_1 and w_2 are set according to the evolutionary difficulties under different modes.

```
/* Suppose the expected polymorphic circuit behaves as a function $f_1$ in mode 1 and as a function $f_2$ in mode 2, and the weight factors for two functions are denoted by $w_1$ and $w_2$, respectively.*/
1. Initialize the fitness value $Fit = 0$ and temporary variables $Fit_1 = 0$, $Fit_2 = 0$.
2. For each input-output combination do
   2.1 Set the circuit into mode 1 and calculate corresponding fitness $Fit_1$;
   2.2 Set the circuit into mode 2 and calculate corresponding fitness $Fit_2$;
3. Calculate the fitness of the complete polymorphic circuit as
   $Fit = w_1 \times Fit_1 + w_2 \times Fit_2$.
```

Fig. 1. The procedure of the fitness evaluation of an individual.

The typical results of both CES and WSM are listed in TABLE I. It can be seen that the scale of the evolved polymorphic circuits is small, and the success rate is still relatively low.

All the six kinds of polymorphic circuits are also adopted as the benchmark circuits in this paper.

TABLE I. THE RESULTS OF EXISTENT EVOLUTIONARY TECHNIQUES

Circuits	**Gates Used**	***Num.Gates****		**Results [5]**	**Results [10]**				
				$w_1 = w_2 = 1$	$w_1 = w_2 = 1$		**Best Results**		
		[5]	**[10]**	***Suc.Rate****	***Suc.Rate***	***Ave.* (*Std.*)**	***Suc.Rate***	***Ave.* (*Std.*)**	w_1 / w_2
5b-parity-median-1	NAND/NOR XOR/XOR	24	50	0.080	0.65	181654 (106159)	0.74	152915 (102429)	1/2000
5b-parity-median-2	NAND/XOR XOR/NOR	20	40	0.053	0.3	248480 (91685)	0.73	145926 (114312)	1/8192
Mult2b-sn4b-1	NAND/NOR AND/AND	40	40	0.008	0.57	204399 (104416)	0.75	169612 (85035)	6/1
Mult2b-sn4b-2	$(a \wedge \bar{b})$/XOR XOR/$(a \wedge \bar{b})$	40	60	0.020	0.34	251233 (117700)	0.47	215805 (105613)	1/256
2b-mult-add-1	NAND/NOR OR/XOR	40	40	0.055	0.59	207557 (106129)	0.84	123375 (84513)	48/1
2b-mult-add-2	NAND/NOR AND/AND	40	40	0.025	0.57	203779 (105874)	0.83	137054 (81271)	1/16

**Num.Gates* denotes the number of total gates in each chromosome.

***Suc.Rate* denotes the success rate.

III. Problem and Strategy

As descriptions in the former section, the success rate of existent Evolutionary Algorithms (EAs) is relatively low, even for small-scale two-state polymorphic circuits.

The typical approaches for improving the evolutionary ability is to design improved versions of to EAs [16], or to introduce high-level functional components as the building blocks instead of primary gates (i.e. function-level evolution) [7], or to develop the Divide-and-Conquer techniques [17]. For polymorphic circuits, due to their intrinsic complexity, neither effective function-level evolution approaches nor Divide-and-Conquer approaches have been proposed, so far.

Compared to the CES, though the WSM can effectively improve the success rate and decrease required evolutionary generations for most polymorphic circuits. However, the WSM needs to set up the appropriate weight values. In this paper, an evolutionary algorithm based on periodical weight adjustment is proposed. It can change its weights periodically according to a predefined function, without the need of efforts to do a number of experiments to obtain appropriate weights.

IV. Evolution Strategy with Periodical Weight Adjustment

For a polymorphic circuit with n functions, the fitness value can be calculated as (1).

$$Fit(x) = \sum_{i=1}^{n} w_i \times Fit_i(x) \qquad (1)$$

Where $Fit_i(x)$ denotes the fitness of the chromosome x under mode i, and w_i denotes the corresponding non-negative weight factor.

Generally, the weight factor is important and will affect the efficiency of EAs. In [5] , both weight factors are actually set to 1. In [10], the weight factors are set in advance, and during the evolutionary process, the weight factors don't change, while in this study, the weight factors are set to change periodically. For convenience, it is named as PWAM.

In the PWAM method, the weight factors are set to change dynamically and periodically according to the sinusoid function, which is expressed as (2).

$$w_i = \sin(2\pi t_i) + 1 \qquad 1 \le i \le n \qquad (2)$$

Where n means the number of functions included in a polymorphic circuit, and t_i is calculated according to (3). Additionally, each weight factor is added by 1 in order to avoid being negative.

$$t_i = \frac{(CG + (i-1)Pd/Den) \bmod Pd}{Pd} \qquad 1 \le i \le n \qquad (3)$$

The objective of (3) is to map the number of current generation into interval [0, 1), where CG means the current evolutionary generation in the evolutionary process. Pd means period, i.e. the changing period of the weight values, for example, if Pd=100, it means after 100 evolutionary generations t_i will be same with its original value. Generally, Pd is set smaller than the maximum number of evolutionary generations.

Due to the sinusoidal function in itself is periodical, (3) can be simplified into (4).

$$t_i = \frac{CG + (i-1)Pd / Den}{Pd} \qquad 1 \le i \le n \qquad (4)$$

Den is a positive number, which can be set to n or other designated values. Its main function is to generate phase difference.

For convenience, a new variable *PhaseDifference* is defined as follows.

$$PhaseDifference = 2\pi \times \frac{[i-(i-1)]Pd/Den}{Pd} = \frac{2\pi}{Den} \qquad (5)$$

The unit for *PhaseDifference* is radian. For example, for a 3-state polymorphic circuit, generally *Den* can be set to 3, and the phase difference between weights will be $2\pi/3$; if *Den* is set to 5, the phase difference between weights will be $2\pi/5$.

Among all the experiments, the case where *PhaseDifference*=0 should be paid more attention. It does not mean $Den = \infty$. In this situation, (4) should be rewritten as (6). This means all weights are changing with evolutionary generations but still keep same with each other.

$$t_i = \frac{CG \bmod Pd}{Pd} \qquad 1 \le i \le n \qquad (6)$$

V. Experiments

In this section, experiments are done to test the performance of PWAM.

A. Parameter Settings

In TABLE II, the parameters of Evolution Strategy are given. The number of runs is 100, which means that each polymorphic circuit is evolved 100 times independently. In all the experiments, the mutation rate is 0.05.

TABLE II. Parameters Used in the Experiments

Number of runs	100
Mutation rate	0.05
Circuit layout	$1 \times Num.Gates$
Maximum number of evolutionary generations	300,000
$\boldsymbol{\mu}$	4
$\boldsymbol{\lambda}$	128

The gate array of the CGP model is $1 \times Num.Gates$. That is to say, r=1, and the number of columns c is equal to the number of gates (*Num.Gates*). Since the number of candidate gates used to evolve different polymorphic circuits is not same, *c* is not a fixed number. For fair comparisons, the number of the gates used for each polymorphic circuit are the same as those in [10] (see TABLE I).

Performance indicators are *Suc.Rate* and *Ave. Suc.Rate* means average success rate, and *Ave.* means the average evolutionary generations. As for *Ave.*, when one run cannot find a functionally correct circuit, the maximum number of evolutionary generations is counted

B. *Comparisons of Different Weight Adjustment Methods*

In this section, the experimental results of the evolutionary design of two-state polymorphic circuits are presented. The intention of these experiments is to test how well the periodical weight adjustment method behaves. For this group of experiments *PhaseDifference*= π .

TABLE III. EXPERIMENTAL RESULTS OF DIFFERENT WEIGHT ADJUSTMENT METHODS

Circuit	**WSM (The best results)**		**PWAM (Period(*Pd*)=10000)**	
	Suc.Rate	***Ave.*(*Std.*)**	***Suc.Rate***	***Ave.*(*Std.*)**
5b-parity-median-1	0.74	152915 (102429)	0.91	75513 (88219)
5b-parity-median-2	0.73	145926 (114312)	0.59	179030 (115166)
2b-mult-add-1	0.84	123375 (84513)	0.91	94486 (90565)
2b-mult-add-2	0.83	137054 (81271)	0.84	123773 (104959)
Mult2b-sn4b-1	0.75	169612 (85035)	0.98	72706 (59359)
Mult2b-sn4b-2	0.47	215805 (105613)	0.89	139953 (89029)

The data in parentheses are standard deviations. As is shown in TABLE III, PWAM works well than the WSM. It can be observed that, for the circuit 5b-parity-median-2, the performance of the PWAM is worse than that of the WSM. For the circuit 2b-mult-add-2, the success rates of WSM and PWAM are similar. For other four circuits, the PWAM is much better than the WSM. Especially, for the circuit Mult2b-sn4b-2, the success rate of the PWAM is higher than that of WSM by 42%.

Due to the PWAM is better, it will be further analyzed in the following sections.

C. *The Impact of Parameters*

1) *Period and PhaseDifference*

In this section, how the parameters Period (*Pd*) and *PhaseDifference* will affect the performance of the PWAM are evaluated.

Fig. 2 demonstrates how the *Period* affects the success rate. It can be observed that, when the *Period* is too small or too large, the success rate is relatively low. The ideal value for the *Period* is between 2000 and 100000. Within this range, the *Period* is not a sensitive parameter for the evolutionary

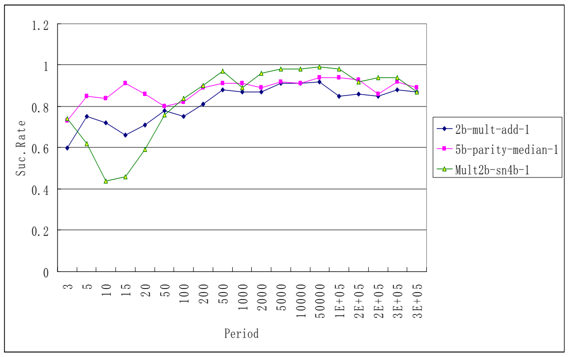

Fig. 2. The relationship between the period and the success rate

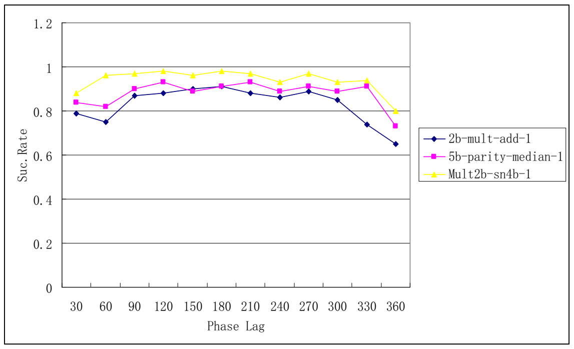

Fig. 3. The relationship between the phase lag and the success rate

algorithm. In the following experiments, the parameter *Period* is fixed at 10000.

Fig. 3 reveals how the phase lag (i.e. *PhaseDifference*) affects the success rate. Similar to the parameter *Period*, the *PhaseDifference* is also not a sensitive value in a certain range. The ideal value for *PhaseDifference* can be chosen from 60^o ($2\pi/6$) to 330^o ($2\pi\times 11/12$). When the phase lag is set to 0^o (0) or 360^o (2π), it means there are no phase differences among the weight factors.

2) *The Number of Candidate Gates*

This section explore the relationship between the number of candidate gates and the success rate. The two-state polymorphic circuits 5b-parity-median-2 and 2b-mult-add-1 are taken as a test workbench. Since the trend is evident, only the test parameters of 5b-parity-median-2 are given in details in TABLE IV.

In TABLE IV, when *Period*=10000, with the number of candidate gates increases from 40 to 60, the success rate increases from 59% to 72%, and the average generation needed for obtaining a solution decreases. It seems the number of candidate gates has much impact on the evolution process. Since the selected CGP array consists of one row, the number of columns *c* is same with the number of candidate gates.

TABLE IV. THE RELATIONSHIP BETWEEN THE NUMBER OF CANDIDATE GATES AND THE SUCCESS RATE

5b-parity-median-2	Period(*Pd*)=2000		Period(*Pd*)=10000	
	Suc.Rate	***Ave.*(*Std.*)**	***Suc.Rate***	***Ave.*(*Std.*)**
Num.Gates=40 (i.e. *c*=40)	0.56	196087 (113462)	0.59	179030 (115166)
Num.Gates=60 (i.e. *c*=60)	0.71	148324 (117044)	0.72	145594 (114527)

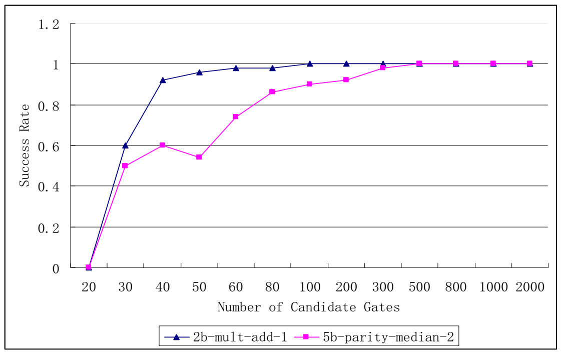

Fig. 4. The number of candidate gates and the success rate

In order to reveal how the number of candidate gates affects the performance of the PWAM, more experiments on 5b-parity-median-2 and 2b-mult-add-1 are conducted, and experimental results are given in Fig. 4.

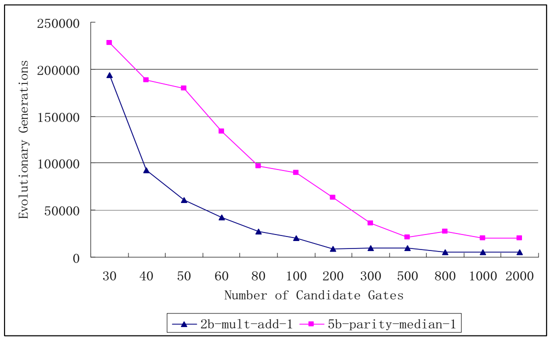

Fig. 5. The number of candidate gates and the average evolutionary generation

From Fig. 4, it can be observed that the success rate increases with the increment of the number of candidate gates. When the number of candidate gates is large enough, the success rate is close to 100%. As can be observed from Fig. 5, for circuits 5b-parity-median-2 and 2b-mult-add-1, the average evolutionary generation decreases gradually at first. However, when the number of candidate gates exceeds 500, the difference of average evolutionary generations is no longer obvious.

D. *Evolving a Three-State Polymorphic Circuit with the PWAM*

In order to compare the performance of the CES, in which each weight factor is 1, with that of the PWAM, experiments on a 3-state polymorphic circuit are conducted. The 3-state polymorphic circuit is composed of a 2-bit multiplier, a 2-bit adder and a 4-bit sort net. In this experiment, *Num.Gates*=80, *Period*=10000.

TABLE V. EXPERIMENTAL RESULTS OF A 3-STATE POLYMORPHIC CIRCUIT

The style of weight adjustment	*Suc.Rate*	*Ave.*(*Std.*)
$w_1 = w_2 = w_3 = 1$	0.46	220995(98169)
PhaseDifference$=2\pi/3$	0.92	98287(83570)
$w_1 = w_2 = w_3$ **(same but changing)**	0.69	174362(105860)

From TABLE V, it can be observed that, when all weight factors are set at 1 (i.e. the CES is used), the success rate is only 42%. While with the PWAM (*PhaseDifference*$=2\pi/3$), the success rate is 92%. It is higher by 50%. The case in which all weight factors are equal but changing simultaneously is also given in TABLE V. Its performance is between those of the former two cases.

Except for two-state polymorphic circuits, the PWAM can be used to evolve more complex circuits. An evolved solution for the 3-state polymorphic circuit is shown in Fig. 6.

The difference of success rate caused by *PhaseDifference* on the evolution design of a 3-state polymorphic circuit is given in TABLE VI. It is obvious that when the *PhaseDifference* is not close to 0^o or 360^o the result is acceptable. No matter which value is chosen from [60^o, 300^o] for *PhaseDifference*, the performance of the algorithm will not change greatly.

TABLE VI. THE RELATIONSHIP BETWEEN PHASEDIFFERENCE AND THE PERFORMANCE OF THE PWAM

Statistics	***PhaseDifference*** **(in Degree)**					
	60	**120**	**180**	**240**	**300**	**360**
Suc.Rate	0.91	0.92	0.89	0.92	0.91	0.69
Ave. (Std.)	97872 (94321)	98287 (83570)	105107 (89125)	103468 (92527)	95128 (90799)	174362 (105860)

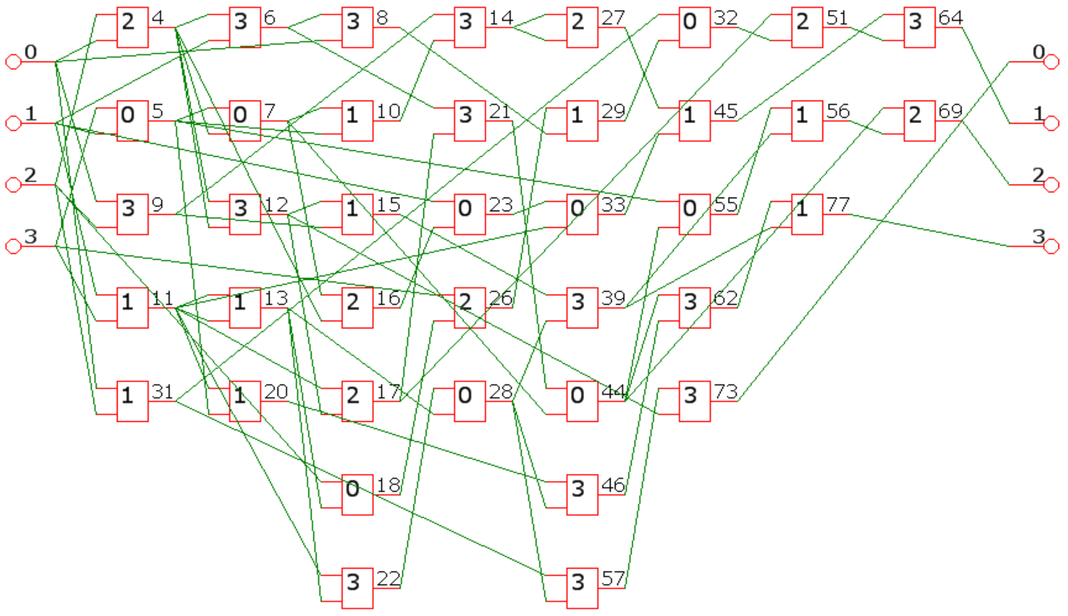

Fig. 6. A three-state polymorphic circuit 2b-mult-add-sn4b. The inputs: A (0-1), B (2-3); the outputs: 0-3 (0-2 in case of adder); gates: 0-nand/nor/or, 1-or/xor/nand, 2-and/nand/and, 3-nor/and/nor

VI. Discussion and Conclusions

In order to improve the ability of the evolutionary design of polymorphic circuits, the periodical weight adjustment method for Evolutionary Strategy, i.e. the PWAM is proposed. The weight factors are made to change dynamically during the evolutionary process. Experiments on six 2-state circuits demonstrate that the PWAM is better than the CES and the WSM in most cases.

A 3-state polymorphic circuit is evolved for the first time. The PWAM shows much better performance. However, for a polymorphic circuit with a bit large scale (e.g. a 3-bit multiplier/3-bit adder); the PWAM still can not design it easily. Therefore, in the future, the scalability is still an important issue to be resolved and why PWAM is better than CES and WSM should be further analyzed.

References

[1] Adrian Stoica, Ricardo Zebulum and Didier Keymeulen, "Polymorphic Electronics," in the 4th International Conference on Evolvable Systems: From Biology to Hardware (ICES'01), Tokyo, Japan, 2001, pp. 291-301.

[2] Richard Ruzicka and Lukas Sekanina, "Physical Demonstration of Polymorphic Self-Checking Circuits," in IEEE International On-Line Testing Symposium, 2008, pp. 31-36.

[3] Adrian Stoica, Ricardo Zebulum and Didier Keymeulen, "Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration," IEE Proceedings on Computers and Digital Techniques, vol. 151, pp. 295 - 300, 18 July 2004.

[4] Adrian Stoica, Ricardo Zebulum and Didier Keymeulen, "On Polymorphic Circuits and Their Design using Evolutionary Algorithms," in IASTED International Conference on Applied Informatics(AI2002), Innsbruck, Austrilia, 2002.

[5] Lukas Sekanina, "Evolutionary Design of Gate-Level Polymorphic Digital Circuits," in Applications on Evolutionary Computing: EvoWorkkshops 2005, 2005, pp. 185-194.

[6] Lukas Sekanina and Zdenek Vasicek, "A SAT-based Fitness Function for Evolutionary Optimization of Polymorphic Circuits " in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, Singapore, 2012, pp. 96-103.

[7] Lukas Sekanina, Vojtech Salajka and Zdenek Vasicek, "Two-Step Evolution of Polymorphic Circuits for Image Multi-Filtering," in WCCI 2012 IEEE World Congress on Computational Intelligence, Brisbane, Australia, 2012, pp. 1-8.

[8] Z. Gajda and L. Sekanina, "On Evolutionary Synthesis of Compact Polymorphic Combinational Circuits," Multiple-Valued Logic and Soft Computing, pp. 607-631, 2011.

[9] Z. Gajda and L. Sekanina, "Gate-level optimization of polymorphic circuits using Cartesian Genetic Programming," in IEEE Congress on Evolutionary Computation, Trondheim, Norway, 2009, pp. 1599-1604.

[10] Houjun Liang, Wenjian Luo and Xufa Wang, "Designing Polymorphic Circuits with Evolutionary Algorithm Based on Weighted Sum Method," in the 7th International Conference on Evolvable Systems: From Biology to Hardware (ICES'07), Wuhan, China, 2007, pp. 331-342.

[11] Julian F. Miller, Cartisian genetic programming: Springer Berlin Heidelberg, 2011.

[12] P. Burian, "Reduction of Fitness Calculations in Cartesian Genetic Programming," in 2013 International Conference on Applied Electronics (AE), Pilsen, 2013, pp. 1-6.

[13] P. Burian, "Compact Version of Cartesian Genetic Programming " in 2014 International Conference on Applied Electronics (AE), Pilsen, 2014, pp. 63-66.

[14] B.W. Goldman and W.F. Punch, "Analysis of Cartesian Genetic Programming's Evolutionary Mechanisms," IEEE Transactions on Evolutionary Computation, vol. 19, pp. 359-373, 2015.

[15] Julian Miller and Dominic Job, "Principles in the Evolutionary Design of Digital Circuits - Part I," in Genetic Programming and Evolvable Machines. vol. 1, 2000, pp. 7-35.

[16] Michal Bidlo and Zdenek Vasicek, "Cellular Automata-Based Development of Combinational and Polymorphic Circuits: A Comparative Study," in International Conference on Evolvable Systems: From Biology to Hardware (ICES'08), 2008, pp. 106-117.

[17] Emanuele Stomeo, Tatiana Kalganova and Cyrille Lambert, "Generalized disjunction decomposition for evolvable hardware," IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 36, pp. 1024-1043, 2006.