A Novel Polymorphic Gate Based Circuit Fingerprinting Technique

Tian Wang, Xiaoxin Cui, Dunshan Yu
Institute of Microelectronics, Peking University, Beijing, China
cuixx@pku.edu.cn

Omid Aramoon, Timothy Dunlap, Gang Qu
Department of Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, USA
gangqu@umd.edu

Xiaole Cui
Key Lab of Integrated Microsystems, Peking University Shenzhen Graduate School, Shenzhen, China
cuixl@pkusz.edu.cn

ABSTRACT

Polymorphic gates are reconfigurable devices that deliver multiple functionalities at different temperature, supply voltage or external inputs. Capable of working in different modes, polymorphic gate is a promising candidate for embedding secret information such as fingerprints. In this paper, we report five polymorphic gates whose functionality varies in response to specific control input and propose a circuit fingerprinting scheme based on these gates. The scheme selectively replaces standard logic cells by polymorphic gates whose functionality differs with the standard cells only on Satisfiability Don't Care conditions. Additional dummy fingerprint bits are also introduced to enhance the fingerprint's robustness against attacks such as fingerprint removal and modification. Experimental results on ISCAS and MCNC benchmark circuits demonstrate that our scheme introduces low overhead. More specifically, the average overhead in area, speed and power are 4.04%, 6.97% and 4.15% respectively when we embed 64-bit fingerprint that consists of 32 real fingerprint bits and 32 dummy bits. This is only half of the overhead of the other known approach when they create 32-bit fingerprints.

KEYWORDS

Polymorphic gate, fingerprinting, satisfiability don't care conditions

ACM Reference format:

Tian Wang, Xiaoxin Cui, Dunshan Yu, Omid Aramoon, Timothy Dunlap, Gang Qu, Xiaole Cui. 2018. A Novel Polymorphic Gate Based Circuit Fingerprinting Technique. In *GLSVLSI '18: 2018 Great Lakes Symposium on VLSI, May 23–25, 2018, Chicago, IL, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3194554.3194572

1 INTRODUCTION

Recent achievements in the field of digital circuit design brings a novel reconfigurable scheme based on polymorphic gates (or polymorphic circuits) that was first introduced by A. Stoica in 2001 [1]. With multiple functionalities integrated in one single structure, polymorphic gates/circuits can achieve function transformation in response to control factors such as temperature, supply voltage or external inputs, etc. One typical polymorphic gate that has been fabricated with HP 0.5um technology is a NAND/NOR gate proposed in [2]. It performs as a NAND gate when Vdd is 3.3V and when Vdd drops to 1.8V, this gate works as a NOR gate. Many follow-up works have been reported on designing various types of polymorphic gates based on evolutionary approach [3][4] and on building complex polymorphic circuits using polymorphic gates [5][6].

Polymorphic gates have been applied in many scenarios such as multifunctional adaptive systems [7][8], finite impulse response(FIR) filter [9], self-checking circuits [10,12] and reduction of test vector volume [11], where multiple working modes are supported, and each mode can be enabled with global control signals. Besides these applications, polymorphic gates have great potentials in hardware security, as the flexible built-in multi-functionality enables one or more conceived 'extra' functions in addition to the 'main' one, which makes it feasible to hide function or embed secret information, such as circuit watermark [21]. The embedded information will show up when the 'extra' function gets activated. One straightforward and convenient application is to embed circuit fingerprint, which allows the tracking of every individually sold IP. When a designer suspects IP piracy or counterfeiting, he can detect the embedded fingerprint to locate the source.

In this paper, we propose a circuit fingerprinting scheme with polymorphic gates controlled by external inputs. The scheme targets SDC (Satisfiability Don't Care) conditions that usually appear in non-trivial circuits and replaces the standard library cells holding the SDC conditions by polymorphic gates. The modified circuit delivers correct functionality and the configurations of the polymorphic gates constitute the circuit fingerprint. We also introduce additional replacements to those gate locations without SDC conditions. The control inputs of these polymorphic gates serve as the dummy fingerprints. Any malicious attempt to modify the dummy fingerprints will bring functional changes and make the fingerprinted circuits function incorrectly.

Our works and contributions are specified as follows,

- After a brief review of the existing design approaches of polymorphic gates, we employ the most widely used genetic algorithm. With this modified algorithm, we successfully construct five polymorphic gates whose function transition is controlled by external signals.
- We propose a polymorphic logic based justifiability checking method to determine the SDC conditions and potential locations where standard cells can be replaced by polymorphic gates to embed fingerprints.
- We perform security analysis by considering different attacking scenarios based on the attackers' capabilities, which demonstrates the robustness and reliability of the proposed scheme.
- We evaluate the area, delay and power overhead introduced by our proposed fingerprinting scheme on ISCAS 85 and MCNC benchmark circuits using 0.13um SMIC technology. Results demonstrate that, when we embed 32-bit real fingerprint and 32-bit dummy fingerprint, the benchmark circuits have an average of 4.04%, 6.97% and 4.15% overhead in delay, area and power, respectively. In other words, our scheme can provide sufficiently strong fingerprints (32 bits) with acceptable performance deterioration.

The rest of the paper is organized as follows: Section 2 gives the background and the design of polymorphic gates. Section 3 presents our fingerprinting scheme based on SDC conditions and the security analysis. In Section 4, we validate the proposed technique based on overhead evaluation, and Section 5 concludes.

2 DESIGN OF POLYMORPHIC GATES

2.1 Genetic Algorithm

Polymorphic gates can be implemented with FTPA (field programmable transistor array) [1], CMOS [2], emerging devices such as silicon nanowire and ambipolar devices [13] [14]. For better integration into the mainstream CMOS technology, we only focus on polymorphic gates consisting of CMOS transistors. Table 1 lists some representative polymorphic gates reported in the literature. These gates deliver different outputs for the same input vector depending on the operating environment such as temperature, supply voltage or specific input signals.

While standard logic gates adopt complementary topology, polymorphic gates employ rather unconventional structure at the transistor level. Due to their irregular topology, it is a challenge to find polymorphic gates. Evolutionary approach [3][15] is the most suitable method to search for potential designs that match perfectly with the required functionalities. Genetic algorithm is one of the most popular variants of evolutionary approach. In the general version of genetic algorithm [16], after the genotype (or gene) is mapped to an artificial system and the initial population of candidate individuals are created, a generative process ranks candidate solutions based on a fitness function which incorporates the desired criteria, and selects the fittest candidates for mutation and reproducing the next generation. This process repeats until an acceptable solution is found. In this paper, we tailor the generalized genetic algorithm for designing polymorphic gates, as shown in Algorithm 1. We assign an index to each terminal of transistors, the source, gate and drain. Genes refer to the width and length value of transistors and the index of the terminals that their source, gate or drain are connected to, which are all initialized in Step 1. The genes are mapped to candidate netlists which are later simulated and evaluated in Step 2. The function for fitness evaluation is the hamming distance between the outputs of the candidate solutions and the desired outputs. In step 3, only those with hamming distance less than a threshold are mutated for reproduction of the next generation.

Table 1: Examples of the Existing Polymorphic Gates.

Gate	Control	Control values	#Transistor
AND/OR[1]	T	27/125℃	6
AND/OR[1]	ext. input	3.3V/0V	6
AND/OR[1]	V_{dd}	3.3V/1.2V	8
AND/OR/XOR[1]	ext. input	3.3V/0V/1.5V	10
NAND/NOR[2]	V_{dd}	3.3V/1.8V	6
NAND/NOR[10]	V_{dd}	5V/3.3V	8
NAND/XOR[4]	ext.input	3.3V/0V	9

2.2 Evolved polymorphic gates

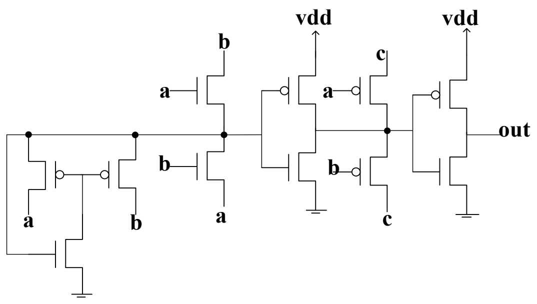
We have evolved several polymorphic gates whose function is selected by a specific control input. The functionalities of these gates and their size (in terms of the number of transistors) are summarized in Table 2. For example, the first row shows a polymorphic gate with 6 transistors that behaves as a NOR gate when the control input C=1, and changes to an inverter when C=0.

Table 2: Evolved Polymorphic Gates

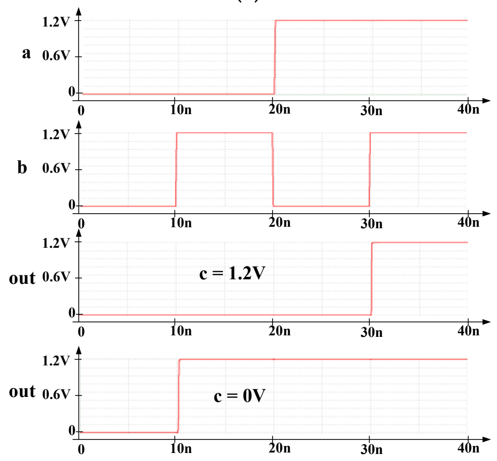
Gate functionalities	#Transistor
NOR(C=1) - INV (C=0)	6
NAND(C=1) - INV (C=0)	7
AND(C=1) – BUF (C=0)	9
AND(C=1) – OR (C=0)	9
NAND(C=1) – NOR (C=0)	9

Fig. 1 (a) presents the schematic of the polymorphic AND/OR gate in the second row from the bottom in Table 2. The nine transistors are connected in irregular topology and take unconventional parameters. The function of this gate is shown in Fig. 1 (b). When the input *c* is logic '1', this gate is an AND gate; when *c* reverses to logic '0', this gate functions as an OR gate.

```
Algorithm 1-Genetic algorithm for evolving polymorphic gates
Input:
Population_No - number of individuals in each generation
Generation_No- number of generations for evolution
f<c, a, b...> - truth table of the desired gate function, where
a,b.... are the input values and c is the external control
signal to transform the function
fitness_threshold – threshold of hamming distance between
the output of each individual and that of the desired gate
function
r - mutation rate of each generation
Output: Gate netlist that performs the desired function.
1.Gene initialization
cnt1 = 0;
  for (cnt2 = 0 ; cnt2 < Population_No; cnt2++)
    Initialize the genes and generate netlist[cnt2];
  end for
2.Simulation and fitness evaluation
for (cnt2 = 0; cnt2 < Population_No; cnt2 ++)
  for different c and every input combination of a,b...
    f' = Simulate(c,a,b,..netlist[cnt2]);
  end for
 fitness[cnt2]= matches(f, f');//Hmming distance calculation
end for
3.Selection and reproduction
  No_indiv= 0;
  for (cnt2 = 0; cnt2 < Population_No; cnt2 ++)
    if (fitness[cnt2] > fitness_threshold)
      Survival= Survival ∪ netlist(cnt2);No_indiv++;
  endif
 end for
 for every netlist in the set Survival
  for(cnt2 = 0; cnt2 <[Population_No/No_indiv];cnt2++)
   Change r% of genes in netlist[cnt2];
  end for
end for
4.While(cnt1 != Generation_No)
   Goto step 2~3; cnt1++;
end while
5. Report the netlists in the set Survival.
```

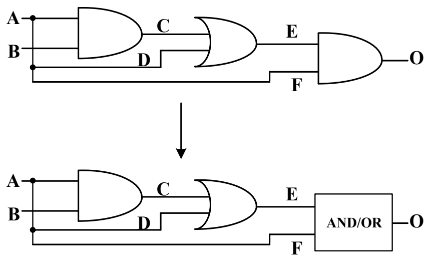


(a)



(b)

Figure 1: Polymorphic AND/OR gate. (a) Topology (b) Input and output waveforms (from top to bottom: input a, input b, output with c =1.2V, output with c = 0V).



(a)

A	B	C	D	E	F	AND	OR
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1

(b)

Figure 2: An example of SDC condition based fingerprinting. (a) The original circuit (top) and the one after the AND gate is replaced (bottom) (b) Truth tables of the internal signals and gates AND and OR.

3 POLYMOPHIC GATE BASED FINGERPRINTING WITH SATISFIABILITY DON'T CARE CONDITIONS

3.1 Satisfiability Don't Care(SDC) conditions

Satisfiability Don't Care conditions describe the logic combinations that will not occur in the internal nets given all the combinations that the primary inputs can take. For example, in Fig. 2, the AND gate in (a) cannot have input patterns (1, 0) or (0, 1) as shown in the truth table in (b).

As shown in Algorithm 2, SDC conditions can be determined by using the justification methodology in VLSI testing, which deduces the logic value from an internal net backwards to the primary inputs. If the justification of a logic pattern in a gate fails, there exists a SDC condition for this gate. For example, in Fig. 2(a), for the AND gate to have inputs E=0 and F=1, we have A=F=1, but this makes D=1 and the OR gate will make E=1. Therefore, input pattern (0,1) fails justification and thus it is a SDC condition.

Algorithm 2-Determining gate locations with SDC conditions

```
Function SDC_checking ( gate g, pattern <$v_1$,$v_2$, … ,$v_n$> )
for i = 0; i < n; i ++
    if ( justify ( g's ith input, $v_i$ ) = fail )
        return pattern<$v_1$,$v_2$, … ,$v_n$> is a SDC condition; // The SDC condition exist.
endfor
  return fail;// The SDC condition doesn't exist.
```

```
Function Justify (gate g, justification value v)
if (v != g's existing output value)// conflict occurs
   backtrace();
Endif
Else
  if ( g functions as AND && v == 1)
      for i = 0; i < number of inputs for g; i ++
          if ( justify ( g's ith input, 1) = fail )
              return fail;
      endfor
endif
if ( g functions as AND && v == 0)
      for i= 0; i < number of inputs for g; i ++
          if ( justify ( g's ith input, 0) = success )
              return success;
      endfor
endif
……
```

3.2 Fingerprinting scheme with polymorphic gates

A circuit fingerprinting technique needs to meet the following requirements [17][20]:

1. *Consistency*. The circuit embedded with fingerprints needs to function correctly.
2. *Uniqueness*. Each circuit should have distinct fingerprints so that it is feasible to differentiate different copies.
3. *Robustness*. To enable the trace of source, the fingerprints must remain unchanged in any illegally produced circuits and shouldn't be modified.

In the proposed scheme, we replace standard cells in the netlist with polymorphic gates whose either modes of operation maintain the correct functionality of the circuit. As shown in Fig. 2, by embedding such polymorphic gates, we can generate fingerprinted copies of the original circuits, where the configuration of each polymorphic gate represents one fingerprint bit.

3.2.1 SDC-based fingerprint

We derive the following principles when determining the potential locations for gate replacement.

P1. Only the gates which have the same logic as either modes of polymorphic gates can be potential locations for replacements. For example, if we want to replace standard cells with evolved AND/OR gate, we should only target the 2-input AND gates and 2-input OR gates.

P2. The SDC conditions of potential replacement locations which we aim to find should be the Differentiating Input Value of the polymorphic gates.

Differentiating Input Values (DIV) of a polymorphic gate are the input combinations for which the polymorphic gate produces different output when the control signal changes. For example, input combinations (1,0) and (0,1) are DIVs of the AND/OR gate in Fig. 1. The output of this gate reverses from 0 to 1 as the control signal changes from '1' to '0' when the two inputs are set as '1' and '0', respectively.

After these gates with such SDC conditions are replaced by polymorphic gates, the functionality of the original circuit remains unchanged no matter which configuration the embedded polymorphic gates take. This satisfies the consistency requirement. We assign a binary value to the control input of each polymorphic gates (e.g. the input *c* in Fig. 1) which serves as a one-bit fingerprint. By replacing *p* standard cells with polymorphic gates, we can obtain 2^p distinct fingerprints meeting the uniqueness requirement. In this scheme, we make this type of replacement with the first three polymorphic gates in Table 2. These gates deliver different outputs with only one input combination (1,0). We need to find the standard library gates with one SDC-condition (1,0), which is more likely to appear than the gates with two SDC conditions, (1,0) and (0,1) if we consider introducing NAND/NOR and AND/OR into the original design.

3.2.2 Adding 'dummy' fingerprinting.

Besides the gates with SDC conditions leading in, we also make some additional replacements for those gates without SDC conditions. To ensure the consistency of the circuit function after modification, the additionally inserted polymorphic gates should be configured to the same logic as the original cells. We take the function control input of these gates as the 'dummy' fingerprint bits as their values are constant and cannot be used for identification purposes. Any attempt to modify the dummy fingerprints will cause incorrect function of the circuit. Therefore, the robustness of the fingerprinted circuits against attacks can be enhanced. In this paper we add dummy fingerprint by replacing two-input logic gates with NAND/NOR and AND/OR gate.

The design flow to embed fingerprints is summarized in Fig. 3. After the locations for replacement are determined, we connect the function control inputs of polymorphic gates to the power supply or ground to configure the functions. Thus, the fingerprint values can be embedded in the circuit at the layout design phase. After the fingerprinted copies of the original circuit are fabricated, we can detect the embedded fingerprints by opening up the chip and identifying the configuration type of the polymorphic gates at each location to recover the corresponding bits in the fingerprints.

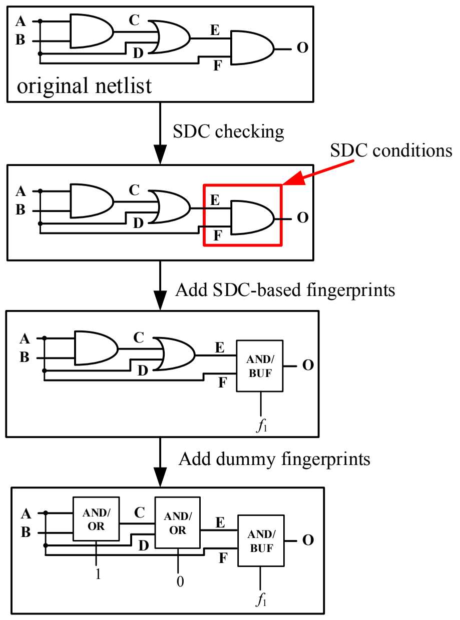


Figure 3: Design flow of the proposed scheme

3.3 Security Analysis

When a suspicious IP or circuit appears, the IP designer needs to recover the fingerprint to trace its source. Attackers who attempt to illegally build the circuit will try to alter the fingerprint to prevent it from being recovered. In this section, we will illustrate that our scheme is resilient against such adversaries and provides robust fingerprints in different attack scenarios.

3.3.1 Fingerprint modification.

Given a single copy of the fingerprinted circuit, the attacker can find the fingerprint locations by targeting the polymorphic gates. The attacker can modify the configuration of these polymorphic gates and change the fingerprint to one that has not been distributed by the vendor. However, this attack requires locating the dummy fingerprint bits as changing these bits would influence the function of the circuits. It is not feasible for the attackers to fully reverse engineer the netlist since the fingerprinted circuit is obfuscated with polymorphic gates. Therefore, the attacker cannot differentiate between the dummy and the real SDC-based fingerprint locations and this attack will become impractical.

3.3.2 Collision attack.

Collision attack is a powerful attack for all fingerprinting schemes where the attacker has access to multiple fingerprinted copies and can compare these copies to determine the fingerprint locations. More specifically, for our proposed scheme, the attacker may extract the fingerprints and compare their values corresponding to different gate locations. If there exists a certain location where the corresponding bit value differs in different copies, this location is a real SDC-based fingerprint location and can be modified. A countermeasure to thwart this attack would be encoding the fingerprint before applying it to the original circuit and checking the parity bits in the extracted fingerprint to help reveal any malicious modifications.

3.3.3 SAT-based attack

SAT-based attacks[18] are powerful adversaries for many logic encryption schemes. As the attacker obtains a gate-level netlist and a functional module of the circuit, he can use the correct input-output pairs to eliminate the wrong configurations of the embedded polymorphic gates for retaining dummy bits, recover the locations of dummy bits, and change them. To gain resilience against these attacks, we can combine schemes like SARLOCK[19], which compares inputs and fingerprint bits to generate a signal that flips the primary outputs. This modification will bring exponential complexity to retrieve the dummy bit locations.

4 SIMULATION VALIDATION AND EVALUATION

We selected several circuits in ISCAS 85 and MCNC benchmarks to perform overhead evaluation. The circuits are described in Verilog HDL and synthesized using 0.13um SMIC technology and the supply voltage is set as 1.2V. For simplicity, the RTL design is mapped to inverters and two-input/three-input/four-input NAND/NOR/OR/AND gates. A program written in C checks all the SDC conditions and replaces the standard cells with the evolved polymorphic gates automatically. The timing and power of the polymorphic gates are measured using Hspice and their layout area are estimated based on the transistor parameters. With these measurements, we incorporate the polymorphic gates into the SMIC 0.13um synthesis library as standard cells. We collect the area, delay and power of the original netlists and the fingerprinted netlists using Synopsys Design Compiler.

The maximum size of fingerprints utilizing SDC conditions and dummy fingerprint locations is listed in Table 3. For most benchmark circuits there are enough number of locations to embed polymorphic gates for our fingerprinting scheme.We set fixed fingerprint width which equals to 16bit and 32bit, respectively and evaluate the overhead. For both cases, we also add dummy fingerprints of equal width. The synthesis results of the fingerprinted netlists are summarized in Table 4 and Table 5. When we add 16-bit real fingerprint and 16-bit dummy fingerprint, the average overhead in delay, area and power is approximately 2%, 3.5% and 4% respectively, which is tolerable; for most benchmarks, performance deterioration is less than 5%. The average overheads rise to 4%, 7% and 5% when we add 16 more bits both in the real fingerprints and dummy fingerprints. Note that in this case, we have 64-bit fingerprints. Compared to the only other similar approach in [17] where their fingerprints are 32 bits, our average overhead is about only half. In addition, the use of dummy fingerprints makes our scheme more robust.

Table 3: Maximum SDC-based and Dummy Fingerprint Size

Circuit	#Gates	#SDC-based fingerprint	#Dummy fingerprint
C880	290	42	114
C1355	424	69	64
C1908	396	52	106
C3540	943	116	124
C5315	1428	47	548
dalu	1228	52	398
des	3483	70	1712
ex5	609	155	64
i8	1174	208	267
i10	1914	134	509

Table 4: Overhead with 16-bit Real Fingerprint and 16-bit Dummy Fingerprint

Circuit	∆delay(%)	∆area(%)	∆power(%)
C880	1.08	8.11	8.16
C1355	5.72	5.9	7.23
C1908	5.1	6.31	4.13
C3540	0.55	3	2.78
C5315	0	1.645	1.93
dalu	2.38	2.56	2.26
des	4.59	0.77	0.50
ex5	0	3.54	13.05
i8	0	1.845	1.79
i10	0.74	1.205	0.77
avg	2.02	3.485	4.26

Table 5: Overhead with 32-bit Real Fingerprint and 32-bit Dummy Fingerprint

Circuit	∆delay(%)	∆area(%)	∆power(%)
C880	1.08	16.31	11.90
C1355	11.74	12.275	8.02
C1908	7.05	13.17	5.09
C3540	1.11	5.835	4.72
C5315	6.8	3.355	2.05
dalu	5.71	4.77	2.49
des	4.59	1.455	0.55
ex5	1.65	6.485	13.26
i8	0	3.635	2.29
i10	0.74	2.4	0.98
avg	4.04	6.97	5.14

5 CONCLUSION

In this paper, we proposed a circuit fingerprinting scheme to resist IP overbuilding and piracy. The proposed scheme utilizes the SDC conditions and make replacements with the polymorphic gates whose functions are controlled by specific inputs into the gate-level netlist. Experiment results demonstrate that our scheme brings low overhead in performance, area and power. One important direction for future work would be investigating the implementation of evolved polymorphic gates and our proposed technique in fabricating real-life circuits.

ACKNOWLEDGMENTS

Omid Aramoon and Gang Qu were supported in part by the National Science Foundation under grant CNS1745466 and by a research agreement between the University of Maryland and the Laboratory for Physical Sciences.

REFERENCES

[1] A. Stoica, R. Zebulum, D. Keymeulen. Polymorphic electronics. *International Conference on Evolvable Systems*. Springer Berlin Heidelberg, 291-302, 2001.

[2] A. Stoica, R. Zebulum, X. Guo, D. Keymeulen, M.I.Ferguson & V.Duong. Taking evolutionary circuit design from experimentation to implementation: Some useful techniques and a silicon demonstration. *IEE Proceedings Computers and Digital Techniques*, 151(4), 295-300, 2004.

[3] L. Sekanina. Evolutionary design of gate-level polymorphic digital circuits. *Workshops on Applications of Evolutionary Computation*. Springer Berlin Heidelberg, 185-194, 2005.

[4] R. Ruzicka. On bifunctional polymorphic gates controlled by a special signal. *WSEAS Transactions on Circuits*, 7(3), 96-101, 2008.

[5] L. Sekanina. Design methods for polymorphic digital circuits. In *Proceedings of the 8th IEEE Design and Diagnostics of Electronic Circuits and Systems* (DDECS), 2005.

[6] W. Luo, Z. Zhang, X. Wang. Designing polymorphic circuits with polymorphic gates: a general design approach. *IET Circuits, Devices & Systems*, 1(6), 470-476, 2007.

[7] L. Sekanina, R. Ruzicka, Z. Vasicek, R. Prokop & L. Fujcik. Repomo32-new reconfigurable polymorphic integrated circuit for adaptive hardware. *IEEE Workshop on Evolvable and Adaptive Hardware*, 2009.

[8] L. Sekanina, L. Starecek, Z. Gajda, Z. Kotasek. Evolution of multifunctional combinational modules controlled by the power supply voltage. In *Proceeding of the 1st NASA/ESA Conference on Adaptive Hardware and Systems*, 86–193, 2006.

[9] L. Sekanina, R. Ruzicka, Z. Gajda. Polymorphic FIR filters with backup mode enabling power savings. *NASA/ESA Conference on Adaptive Hardware and Systems*, 2009.

[10] L. Sekanina, R. Ruzicka, R. Prokop. Physical demonstration of polymorphic self-checking circuits. *14th IEEE International On-Line Testing Symposium*, 2008.

[11] L. Sekanina, L. Starecek, Z. Kotasek, Z. Gajda. Polymorphic gates in design and test of digital circuits. *International Journal of Unconventional Computing*, 4(2), 125, 2008.

[12] L. Sekanina. Evolution of Polymorphic Self-Checking Circuits. In *Proceedings of the 7th Conference on Evolvable Systems: From Biology to Hardware*, 186–197, 2007.

[13] Y. Bi, K Shamsi, J.S. Yuan, P. E. Gaillardon, G.D. Micheli, X. Yin.Emerging technology-based design of primitives for hardware security. *ACM Journal on Emerging Technologies in Computing Systems*, 13(1), 2016

[14] R. Richard and T. Radek. Let's move polymorphism downwards: On the multifunctional logic based on ambipolar behaviour of semiconductor devices. *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era* (DTIS), 2016.

[15] J.F. Miller, J. Dominic and K.V. Vesselin. Principles in the evolutionary design of digital circuits—Part I. *Genetic programming and evolvable machines*, 1(1-2), 7-35, 2000.

[16] J.H.Holland. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992.

[17] C. DunBar, G. Qu. Satisfiability don't care condition based circuit fingerprinting techniques. 2015 *20th Asia and South Pacific Design Automation Conference* (ASP-DAC), IEEE, 815-820, 2015.

[18] P. Subramanyan, S. Ray, and S. Malik. Evaluatingthe Security of Logic Encryption Algorithms. *IEEE International Symposium on Hardware Oriented Security and Trust*, 137–143, 2015.

[19] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu. SARlock: SAT Attack Resistant Logic Locking. *IEEE International Symposium on Hardware Oriented Security and Trust*, 236–241, 2016.

[20] G. Qu and M. Potkonja, Fingerprinting intellectual property using constraint-addition. *Proceedings of the 37th annual design automation conference*, 587-592, 2000.

[21] T. Wang, X. Cui, D. Yu, O.Aramoon, T.Dunlap, G. Qu, et al. Polymorphic gate based IC watermarking techniques. *Asia and South Pacific Design Automation Conference*, 90-96, 2018.