Polymorphic Gate based IC Watermarking Techniques

Tian Wang, Xiaoxin Cui, Dunshan Yu

Institute of Microelectronics, Peking University
Beijing, China 100871
e-mail: 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
e-mail: gangqu@umd.edu

Xiaole Cui

Key Lab of Integrated Microsystems
Shenzhen, China 518055
e-mail: cuixl@pkusz.edu.cn

Abstract - Polymorphic gates are reconfigurable devices whose functionality may vary in response to the change of execution environment such as temperature, supply voltage or external control signals. This feature makes them a perfect candidate for circuit watermarking. However, polymorphic gates are hard to find because they do not exhibit the traditional structure. In this paper, we report four dual-function polymorphic gates that we have discovered using an evolutionary approach. With these gates, we propose a circuit watermarking scheme that selectively replaces certain standard logic gates with the polymorphic gates. Experimental results on ISCAS and MCNC benchmark circuits demonstrate that this scheme introduces low overhead. More specifically, the average overhead in area, speed and power are 4.10%, 2.08% and 1.17% respectively when we embed 30-bit watermark sequences. These overheads increase to 6.36%, 4.75% and 2.08% respectively when 10% of the gates in the original circuits are replaced to embed watermark up to more than 300 bits.

I Introduction

The advances in integrated circuit(IC) technology has led to demands for efficient ways to implement increasingly complex electronic systems. Multifunction or reconfiguration schemes are promising solutions to this problem as they aim to expand the potential scope and utility of electronic devices through simplified and minimal number of components [1]. Traditional multifunctional systems are implemented by multiplexing between multiple stand-alone conventional subsystems. This straightforward approach satisfies the required functionality at the cost of larger implementation overheads.

Recent achievements in the field of digital circuit design brings another concept – so called polymorphic gates (or polymorphic circuits), which was introduced by A. Stoica as a novel type of reconfigurable scheme [2]. Different from the conventional reconfigurable circuits, polymorphic circuits need no reconfiguration switch, as the multiple-functionality is inherently embedded in them. Function transitions are triggered by changes in temperature, supply voltage or external signals, etc. For example, one typical polymorphic gate that has been fabricated using HP 0.5um technology is a NAND/NOR gate proposed in [3]. 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 [4][5] and on the design of polymorphic circuits using polymorphic gates [6][7].

With two or more functions built in one single compact structure, polymorphic circuits have found many applications where a few predefined functions have to be implemented and a global control signal selects the function, such as multifunctional adaptive systems [8,13], finite impulse response(FIR) filter [9], self-checking circuits [10], reduction of test vector volume [11,15], etc. For these scenarios, two modes are supported in polymorphic circuits, where one mode is considered the main operation mode and the other mode is only for special occasions. Notice that the special mode remains invisible when the circuits delivers normal function. Therefore, it is feasible to hide secret information and activate the special mode with specific control factors to reveal the secret.

In this paper we aim to dig out the potentials of polymorphic circuits in hardware security and trust related applications, which hasn't been comprehensively researched in previous works. One straightforward and convenient application of polymorphic gates is to embed circuit watermark, which is one of the first studied hardware security problems [12]. In this scheme, the circuit delivers correct functionality in the normal mode; when it's necessary to demonstrate the watermark, the circuit is transitioned to the special mode by activating the external control so that the circuit can change its functionality and produce different outputs. In this case, the hidden "secret" is the hardware-level watermark, which proves the ownership of the circuits and gives the circuits legal protection against piracy, overbuilding and counterfeiting.

Based on above considerations, we have put forward a watermarking scheme by replacing the standard logic gates in the design netlist with polymorphic gates. Our works and contributions are specified as follows:

- After making a second visit to the design approaches of polymorphic gates, we optimize the most widely used genetic algorithm and implement an automatic design tool; with this tool, we have successfully constructed four polymorphic gates.
- We propose a polymorphic logic based circuit watermarking scheme by replacing standard library cells with polymorphic gates. The scheme features easy detectability [14] which is achieved by a justifiability and observability checking algorithm.
- We evaluate the area, delay and power overhead introduced by our proposed watermarking schemes on ISCAS 85 and MCNC benchmark circuits using 0.13um SMIC technology.

Results demonstrate that with 10% of the gates in the circuits being replaced by polymorphic gates, we are able to embed watermarks up to 300 bits, the circuits embedded with watermarks have an average of 6.36%, 4.75%, 2.08% overhead in delay, area and power, respectively. In other words, our scheme can provide sufficiently strong watermarks with acceptable performance deterioration. Furthermore, the proposed watermarking scheme delivers near perfect solution on fairness [20, 21].

The rest of the paper is organized as follows: Section II gives the background on polymorphic gates and their design methodology, the evolutionary approach. Section III presents our design tool and the polymorphic gates we have evolved. In Section IV, we propose our watermarking technique based on the evolved polymorphic gates, and Section V concludes

II. Background

As the basic building unit of polymorphic circuits, polymorphic gates can be implemented with FTPA (field programmable transistor array) [2], CMOS [3], emerging devices such as silicon nanowire and ambipolar devices [16] [17]. Table I lists some representative polymorphic gates reported in the literature. Different from conventional logic gates, these gates will deliver different outputs for the same input vector depending on the operating environment such as temperature, supply voltage or external signals. So far there are only two fabricated polymorphic gates. The rest of the gates are either simulated or tested with FPTA. For better integration in the mainstream CMOS technology, we only focus on polymorphic gates consisting of CMOS transistors.

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 [4][18] is the most suitable method to search for potential gate designs that match perfectly with the required multiple functionalities [13]. As long as the gate specifications (usually takes the form of truth table) are given, candidates can be evaluated and ranked. Genetic algorithm is one of the most popular variants of evolutionary approach. In the generalized version of Genetic algorithm [19], 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. We assign an index for the source, gate and drain terminals of each transistor respectively. Genes refer to the index of the terminals that the source, gate or drain are connected to and the width and length values of each transistor. The fitness function is the hamming distance between the outputs of the candidate solutions and the desired outputs.

TABLE I
Examples of the existing polymorphic gates.

Gate	Control	Control values	Transistors
AND/OR[2]	temperature	27/125C	6
AND/OR[2]	ext. voltage	3.3V/0V	6
AND/OR[1]	V_{dd}	3.3V/1.2V	8
AND/OR/XOR[2]	ext. voltage	3.3V/0V/1.5V	10
NAND/NOR[3]	V_{dd}	3.3V/1.8V	6
NAND/NOR[10]	V_{dd}	5V/3.3V	8
NAND/XOR[5]	ext. voltage	3.3V/0V	9

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<*t*, a, b…> - truth table of the desired gate function, where *a*,*b*…. are the input values and *t* is the external control signal to transform the function
HD_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 t and every input combination of a,b…
      f' = Simulate(t,a,b,..netlist[cnt2]);
  end for
      HD[cnt2] = hamming distance (f,f');
end for
```

3. Selection and reproduction

```
No_indiv= 0;
for (cnt2 = 0; cnt2 < Population_No; cnt2 ++)
    if (HD[cnt2] == 0)
      Report(netlist[cnt2]);
    endif
    elseif (0<HD(cnt2) < HD_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
```

Notice that although the genes are randomly initialized and modified, there are some practical constraints that needs to be followed, which are specified as follows.

C1. At least one terminal should be connected to power, ground, output, inputs, respectively.

C2. No floating nodes should appear in the circuits.

C1 is to ensure that each design will be a complete gate. To meet C2, we force the source, drain and gate terminals of one transistor to be connected to one terminal of other transistors so that every terminal is in the path from power to the ground. The modified genetic algorithm is shown as in Algorithm 1.

III. Designing logic gates with evolutionary approach

A. Experimental setup

We target two-input one-output gate and consider temperature as the external control signal, which ranges from -25°C to 150°C. The 130nm SMIC technology is adopted and the supply voltage is set as 1.2V.

As shown in Fig.1, we have developed an automatic design platform that integrates a netlist generation module, Hspice simulator and function evaluation module based on the genetic algorithm specified above. The netlist generation module mutates the genes of each individual in every generation to reproduce the next generation. The genes are translated into the .sp netlists that are later fed into Hspice for simulation. The measurements in the output files (.lis) of Hspice simulator are extracted by the function evaluation module so that the fitness function value (here refers to the Hamming distance) of each candidate can be calculated. Both the netlist generation module and the fitness evaluation module are written in C.

B. Evolved polymorphic gates

With the design tool, we have evolved four novel polymorphic gates for the first time. The function of these gates are listed in Table II.

Fig.2(a) presents the schematic of the polymorphic NOR/INV gate. These transistors are connected in irregular topology and takes unconventional parameters. The function of this gate is shown in Fig.2(b). At room temperature, this gate is a NOR gate; when temperature rises from to 125C, this gate inverts the second input.

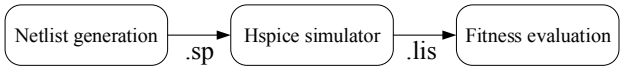
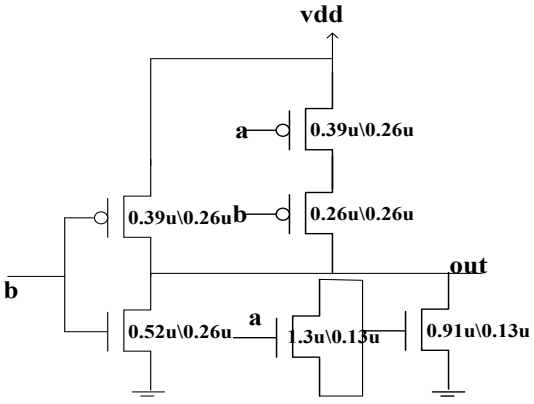


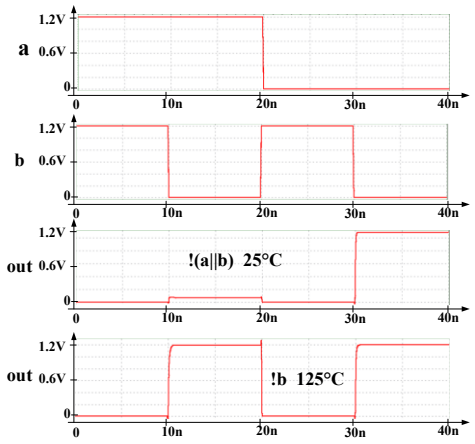
Fig. 1. Platform for implementing the genetic algorithm.

TABLE II
Evolved polymorphic gates controlled by temperature.

Gate function	Transistor
OR(25°C)-AND (125°C)	6
NOR (25°C)-INV(125°C)	6
NAND (25°C)-INV (125°C)	6
AND (25°C)-BUF (125°C)	7



(a)



(b)

Fig.2 Polymorphic NOR/INV gate. (a) Topology (b) Input and output waveforms.

IV. Proposed circuit watermarking scheme with evolved polymorphic gates

In this section we propose a circuit watermarking scheme based on embedding the polymorphic gates we've evolved into the original design.

A. Polymorphic logic based justifiability and observability checking algorithm

We embed the evolved polymorphic gates into the design by replacing the standard cells in the gate-level netlist with polymorphic gates. Every cell in the original netlist represents a possible location for gate replacement. However, the suitable locations for replacement should ensure:

1. The circuit functions correctly at normal operation mode, after the gates in these locations are replaced.

2. The functionality of the modified circuit in normal mode and special mode can be differentiated by observing the primary outputs of the circuit.

To find the suitable locations to embed polymorphic gate, we address three principles that needs to follow. These

principles ensure the correct functionality of the circuit and facilitate the easy detectability of the embedded watermark [12].

P1. Only the gates that have the same functionality as the polymorphic gates in normal mode could be potential locations for watermark embedding.

P2. There should be at least one pattern for primary inputs of the circuit that can 'activate' the polymorphic gate. An input pattern can '**activate**' a polymorphic gate if it can set the inputs of the gate to a Differentiating Input Value.

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 combination (1,0) is DIVs of the NOR/INV gate in Fig.2, for the output of this gate reverses from 0 to 1 as temperature changes from 25C to 125C when the two inputs are set as '1' and '0'.

Moreover, this primary input pattern should also '**propagate**' the output of this gate to the primary output so that the function transformation can be observed.

To find such input patterns, we adopted the justification methodology in VLSI testing to deduce the logic value from the input of the candidate gate backwards to the primary input. Only a gate that is justifiable can be a potential replacement location.

Besides justifiability, we also need to check the observability of this gate location. We propagate the output of one gate by justifying the other input(s) of its successor gate as non-controlling values. Take the circuit in Fig.3 as an example. To propagate the output of gate 1, the first input of its successor gate 4 should be justified as '1' so that the output of gate 4 is determined by the output of gate 1. Similarly, we advance the output one gate at a time until it reaches the primary output.

P3. The changes in the functionality of one polymorphic gate upon changing the temperature shouldn't influence the activation and propagation of other polymorphic gates.

To cater for P3, we need to incorporate the following criteria when checking the justifiability and observability of a certain location.

First, according to **P1**, every gate that has the same function with one of the polymorphic gates could be possible locations for gate replacement. When checking the justifiability or observability of a gate, we may need to justify other gates that will be replaced by polymorphic gates. For these gates, we should assign DIV with least priority. For example, if we want to check whether a NOR gate (gate A) could be replaced by a NOR/NAND and we need to set the output of another two-input NOR gate (gate B) as '0', we will try the input patterns '11' first for B and then '01'or'10' if '11' fails. As '11' gives the same output for NOR and NAND, even if gate B is replaced by a polymorphic gate, the transition of its output when control signal changes will not influence the activation or propagation of gate A.

Second, when checking the observability of a gate location we need to make sure that even if the successor gates are replaced by polymorphic gates, the gate output can still propagate in special mode. We give a case-by-case solution based on analyzing the truth-table of the four polymorphic gates.

Algorithm 2- Justifiability and observability checking algorithm

Input: netlist of the original design, polymorphic gate g_p whose function changes from $f1$ to $f2$ when temperature changes.

Output: locations that can be replaced by polymorphic gates

```
for every gate g in the gate netlist
  if(g functions as f1&& Activate(g) = success && Propagate(g) = success)
    Report g is a possible location to embed polymorphic gate
  endif
end for
```

```
Function Activate (gate g)
    for i= 0; i < number of inputs for g; i ++
        if (justify (g's ith input, v_i) = fail)
            return fail;
        endif
    endfor      // f1(v_1, v_2, …) != f2(v_1, v_2, …)
return success;
```

```
Function Propagate (gate g)
  if (g is output) return success;
  for i = 0; i< number of fanout gates for g; i ++
    g_f = g's ith fanout gate;
    n_i = number of inputs for g_f;
    if ((g_f is NAND or g_f is AND)&& n_i ==2)
      if (justify (g_f's 2nd input, 1) == success)
        return success; endif
    endif
    if (g_f is NOR && n_i ==2)
      if (justify (g_f's 1st input, 0) == success)
        return success; endif
    endif
    if (n_i !=2)
      for j = 0; j < n_i; j++
        if (justify (g_f's jth input, v_non_con) == success)
          return success; endif
      endfor
    endif
  endfor
return fail;
```

```
Function Justify (gate g, justification value v)
  if (v != g's existing output value)// conflict occurs
      backtrace();
  Endif
  Elseif (g functions as f1&& v = v_d )
    for every input patterns giving same output for f1 and f2
      for i= 0; i < number of inputs for g; i ++
        if (justify (g's ith input, v_i') = fail)
          break; // f1(v_1', v_2,'…) = f2(v_1', v_2', …) = v_d
      endfor
    endfor
    for i= 0; i < number of inputs for g; i ++
      if (justify (g's ith input, v_i) = fail)
        return fail; // f1(v_1, v_2, …) = v_d != f2(v_1, v_2, …)
    endfor
  endif
    …
```

Case 1. The successor gate is a two-input NOR gate. Suppose this gate is replaced by NOR/INV gate. If the second input of this gate is justified as '0' and the first input is connected to the signal that needs to propagate, when temperature rises, the gate inverts the second input and the output of this gate is always '1' so the propagation fails. If the two inputs are swapped (the first input is justified as '0'), the gate delivers the complementary of the second input so that the propagation can proceed.

Similarly, if the successor gate is two-input NAND (or AND) gate, it may be replaced by the NAND/INV (or AND/BUF) that inverts (or outputs) the first input. In this case the second input should be justified as '1' and the signal to propagate is connected to the first input.

Case 2. The successor gate is a two-input OR gate. If one input of this OR gate is justified as '0' and this gate gets replaced by a polymorphic OR/AND gate, when temperature rises, this gate turns into an AND gate. Since one input of this gate is '0', the output will be '0' no matter what value the other input (which is the signal we want to propagate) takes, which makes the propagation fail. Therefore, we will avoid propagating to two-input OR gates.

Based on these principles, we have come up with a justifiability and observability checking algorithm to find out the possible locations to insert polymorphic gates. The algorithm is specified as in Algorithm 2.

Fig. 3 gives a motivational example on how we run this algorithm. Suppose that we choose the evolved NOR/INV gate. For the given circuit, there are three NOR gates that can be replaced. As discussed before, the two inputs of the NOR gate should be justified as '0' and '1' so that the function transition can be observed if the gate is replaced. For gate 1, i4 and i5 are set as '1' and '0', respectively. As marked in Fig.3, by propagating the output of g1 forward to the primary output, we derive i1= '0', i2= '0' and i3= '1'. Thus, the output of gate 1 can be observed at the primary output. Similarly, we derive 'x10xx' (x is the don't care value) for gate 3. By changing the temperature, we can observe the output to tell whether gate 1/gate 3 is a standard NOR gate or a polymorphic NOR/INV gate. For gate 2, such input pattern doesn't exist, so gate 2 is not an ideal location to insert polymorphic gate.

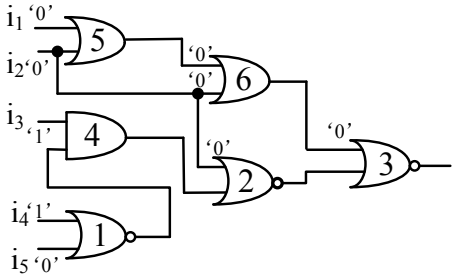


Fig.3 Motivation example of Algorithm 2

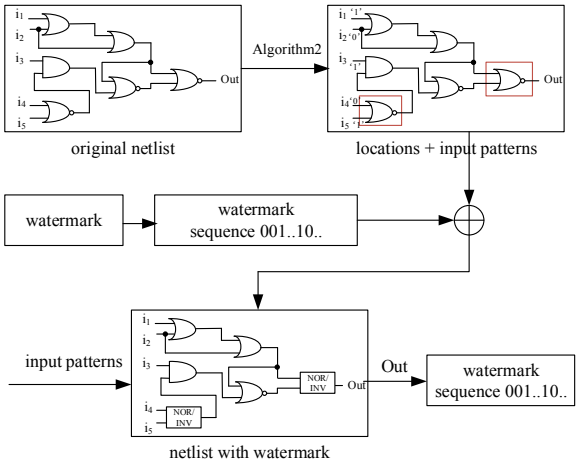


Fig. 4 Design flow of the proposed watermarking scheme

B. *Watermarking scheme with polymorphic gates*

The watermarking scheme with polymorphic gates involve the following steps in the design phase as illustrated in Fig. 4.

- Determine the locations (e.g. gate 1 and 3 in Fig. 3) to insert polymorphic gates with the Algorithm 2 in IV.A. The input vectors for detecting the watermark are also obtained at this step ('00110' for gate 1 and 'x10xx' for gate 3).
- Generate the watermark sequence. Bit '1' ('0') in this binary pattern means a standard logic gate in a certain location will (will not) be replaced by a polymorphic gate.
- Replace standard logic gates with polymorphic gates. Thus the watermark is embedded in the 'hidden' function when the circuit works at special mode.

To detect the watermark, the users need to feed the input vectors provided by the designers ('00110' and 'x10xx') into the circuits and compare the primary outputs gathered at normal mode and special mode. If a difference is observed at a certain primary output, it indicates that the circuits has employed a polymorphic gate at the corresponding location, which means a bit '1' of the watermark sequence is detected; and vice versa. Therefore, the watermark sequence can be retrieved, which proves the ownership of the design.

C. *Overhead evaluation of the scheme*

We selected several circuits in ISCAS 85 and MCNC benchmarks to perform overhead evaluation. The circuits are described in Verilog HDL and synthesized using 130nm SMIC

TABLE III
Area, performance and power overhead of the proposed scheme (small watermark sequence length)

Circuit	No. of gates	No. of possible locations	Δ delay (%)	Δ Area (%)	Δ Power (%)
C880	290	94	0	0.66	0.43
C1355	424	32	6.62	0.47	0.24
C1908	396	114	1.70	0.12	1.11
C3540	943	83	0.18	0.19	0.47
C5315	1428	556	0	0.50	0.46
dalu	1228	131	0	0.23	0.44
des	3483	1084	0	0.07	0.53
ex5	609	48	0	0.37	0.54
i8	1174	277	0	0.16	0.51
i10	1914	526	0.74	0.09	0.27
vda	483	131	0	0.35	0.006
Avg	-	-	0.84	0.29	0.45

technology and the supply voltage is set as 1.2V. For simplicity, the behavioral design is mapped to invertors and two-input/three-input/four-input NAND/NOR/OR/AND gates. The evolved gates are replaced in the gate-level netlist automatically with a netlist modifier written in C. 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 measure the area, delay and power of the original netlist and the modified netlist after polymorphic gates are embedded using Synopsys Design Compiler. The overheads are evaluated in three different scenarios:

1. We start from the simplest case where only a small size watermark is needed. Choose a small number (say 4) of possible gate locations to embed polymorphic gates. We traverse all the 16 cases where the watermark sequence ranges from "0000" to "1111"; for each watermark sequence, we replace the gate at a certain location with polymorphic gate if the corresponding bit in the sequence is '1'. Therefore, we obtain 16 netlists embedded with 16 different watermarks and measure their maximum path delay, area and power using Design Compiler. The average overhead of the 16 modified netlists is reported in Table III. From this, we can see that the proposed watermarking scheme can achieve very good fairness [20, 21] except on the delay metric for one circuit C1355.

2. In real-world practices, a large number of unique watermarks are needed, therefore, the watermark sequence should be long enough. We set a fixed length of watermark sequence and choose a relatively large number of (i.e. 20 or 30) potential gate locations. Since it is not practical to exhaust all the possible 2^{20} or 2^{30} cases, we randomly generate 100 20-bit (30-bit) watermark sequences, embed polymorphic gates and obtain 100 modified netlists. The average overhead of these netlists is reported in Table IV.

3. In this scenario, we set the size of watermark proportional to that of the circuit it secures. We replace 5% and 10% of the total number of gates with polymorphic gates, which ensures the space of candidate watermarks for most benchmark circuits. Similarly, we randomly generate 100 watermark sequences and report the average overhead of the modified netlists in Table V.

From Table III we can tell that for most benchmark circuits there are enough number of locations to embed polymorphic gates for our watermarking scheme. A small size watermark (4-bit watermark sequence) introduces little overhead, which is an average of less than 1% increase in delay, area and power, respectively. Table IV shows that with a fixed-length watermark sequence, the watermarking scheme incurs an average of less than 5% overhead. In Table V, when replacing 10% of the gate with polymorphic gates, the average overhead may rise to 6%. For large size benchmark circuits, the watermark sequence length equals 5% and 10% of the total gate number, which far exceeds that in Table IV. A long watermark sequence means on average more gates are replaced by polymorphic gates and contributes to a larger overhead in Table V.

TABLE IV
Area, performance and power overhead of the proposed scheme (fixed watermark sequence length)

Circuit	30 gates to be replaced			20 gates to be replaced		
	Δ delay (%)	Δ Area (%)	Δ Power (%)	Δ delay (%)	Δ Area (%)	Δ Power (%)
C880	7.88	4.05	5.54	0.54	2.77	3.97
C1355	12.34	3.46	0.89	11.44	2.22	0.70
C1908	4.62	4.15	0.47	6.08	2.52	0.76
C3540	7.24	1.53	1.44	8.17	0.93	0.48
C5315	0	0.47	0.75	0	0.14	0.61
dalu	0	1.78	0.49	0.47	1.17	0.14
des	0	0.48	0.44	0.28	0.28	0.49
ex5	5.96	2.44	1.33	0	1.63	1.40
i8	0.30	1.21	0.67	0.30	0.83	0.58
i10	1.79	0.71	0.43	3.13	0.47	0.26
vda	4.96	2.59	0.42	3.19	1.97	0.32
Avg	4.10	2.08	1.17	3.05	1.36	0.88

TABLE V
Area, performance and power overhead of the proposed scheme (varying watermark sequence length)

Circuit	10% of the gates to be replaced			5% of the gates to be replaced		
	Δ delay (%)	Δ Area (%)	Δ Power (%)	Δ delay (%)	Δ Area (%)	Δ Power (%)
C880	4.89	3.74	4.70	2.44	1.80	2.73
C1355	12.04	3.59	0.95	11.44	2.36	0.69
C1908	6.56	5.18	1.27	7.54	2.02	0.10
C3540	11.15	4.47	3.68	5.94	2.55	1.52
C5315	5.95	4.67	2.24	5.31	1.86	1.31
dalu	0.71	7.12	3.37	0.23	3.52	1.45
des	8.90	5.77	0.63	8.04	2.79	0.98
ex5	9.60	4.01	3.35	0.33	2.64	1.97
i8	1.85	4.63	1.24	0.30	2.36	0.71
i10	2.23	4.97	1.23	2.08	2.48	0.54
vda	6.02	4.06	0.22	2.48	2.11	0.06
Avg	6.36	4.75	2.08	4.19	2.41	0.92

V. Summary and Conclusions

In this paper we proposed a circuit watermarking scheme to resist overbuilding and piracy. The proposed scheme is based on embedding polymorphic gates that have been evolved with the genetic algorithm 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 techniques in fabricating real-life circuits.

Acknowledgement: We thank Dr. Bill Johnson and Dr. Doug Ketchum for their valuable discussion and suggestion. This work is supported in part by the National Science Foundation under grant CNS1745466 and by the Department of Defense.

References

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

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

[3] 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, vol.151, no.4, pp.295-300,2004.

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

[5] R. Ruzicka, "On bifunctional polymorphic gates controlled by a special signal," WSEAS Transactions on Circuits, vol.7, no.3, pp. 96-101, 2008.

[6] L. Sekanina, "Design methods for polymorphic digital circuits," Proc. of the 8th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop DDECS, 2005.

[7] W. Luo, Z. Zhang, X. Wang, "Designing polymorphic circuits with polymorphic gates: a general design approach," IET Circuits, Devices & Systems, vol.1, no.6, pp. 470-476, 2007.

[8] 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.

[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, vol. 4, no.2, pp.125, 2008.

[12] G. Qu and M. Potkonjak, "Intellectual Property Protection in VLSI Designs: Theory and Practice", Kluwer Academic Publishers, 2003.

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

[14] G. Qu, "Publicly Detectable Watermarking for Intellectual Property Authentication in VLSI Design", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 11, pp. 1363-1368, Nov, 2002.

[15] L. Sekanina, "Evolution of Polymorphic Self-Checking Circuits," Proc. of the 7th Conference on Evolvable Systems: From Biology to Hardware, pp.186–197, 2007.

[16] Y. Bi, K Shamsi, J.S. Yuan, P. E. Gaillardon, G.D. Micheli, X. Yin et al., "Emerging technology-based design of primitives for hardware security," ACM Journal on Emerging Technologies in Computing Systems, vol. 13, no.1, pp.3, 2016.

[17] 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), pp.1-5, 2016.

[18] J.F. Miller, J. Dominic and K.V. Vesselin, "Principles in the evolutionary design of digital circuits—Part I," Genetic programming and evolvable machines.vol.1, no.1-2, pp.7-35, 2000.

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

[20] G. Qu, J.L. Wong, and M. Potkonjak, "Fair Watermarking Techniques", Proceedings of the 2000 Asia and South Pacific Design Automation Conference, pp. 55-60, 2000.

[21] J.L. Wong, G. Qu, and M. Potkonjak, "Optimization-intensive Watermarking Techniques for Decision Problems", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 1, pp. 119-127, Jan, 2004.