The ISC Int'l Journal of Information Security

January 2015, Volume 7, Number 1 (pp. 75–81)
http://www.isecure-journal.org

Short Paper

Security-aware Register Placement to Hinder Malicious Hardware Updating and Improve Trojan Detectability

Mehrshad Vosoughi [1,*], and Ali Jahanian [2]
[1] *Electrical and Computer Engineering Department, Qazvin Branch of Islamic Azad University, Qazvin, Iran*
[2] *Electrical and Computer Engineering Department, Shahid Beheshti University, Tehran, Iran*

A R T I C L E I N F O.

Article history:
Received: 12 July 2014
First Revised: 9 December 2014
Last Revised: 20 May 2015
Accepted: 14 July 2015
Published Online: 17 July 2015

Keywords:
Clock Tree Construction, Hardware Security, Register Placement.

A B S T R A C T

Nowadays, bulk of the designers prefer to outsource some parts of their design and fabrication process to the third-part companies due to the reliability problems, manufacturing cost and time-to-market limitations. In this situation, there are a lot of opportunities for malicious alterations by the off-shore companies. In this paper, we proposed a new placement algorithm that hinders the hardware Trojan insertion or simplifies the detection process in existence of Trojans. Experimental results show that the proposed placement improves the Trojan detectability of the attempted benchmarks against Trojan insertion more than 20% in reasonable cost in delay and wire length.

1 Introduction

Currently, many of design houses are fab-less companies that outsource their fabrication process to offshore facilities known as foundries. [1]. In this situation, ICs are becoming increasingly vulnerable to malicious activities and alterations. The main concern of these vulnerabilities is in military systems, financial infrastructures and transportation security.

An adversary can alter the chip to disable or destroy a system at an arbitrary time or leak confidential information and secret keys. This alteration, which is called Hardware Trojan Horse (HTH), can be implemented in ASIC, commercial-off-the-shelf (COST) parts and any kind of microprocessors and microcontrollers. It can also be implemented as firmware modifications for examples in FPGA bit streams. This issue has been reported in some recent documents such as in IEEE Spectrum [1].

The design phase in IC fabrication process which is commonly designed by commercial CAD tool developers is safe and trusted. However, the IP blocks, models, and standard cells used by the designer during the design process and by the foundry during the post-design processes are untrusted. The fabrication phase is also untrusted. That is because the adversary can substitute or insert Trojan cells in the design during this phase to reach his goals.

Two different strategies can be made against the hardware Trojans: HTH detection and HTH prevention. Considerable contributions in the last few years are reported on HTH detection. Hardware Trojan detection methods can be categorized as either destructive, or non-destructive. Destructive methods are mostly reverse engineering methods that are very expensive and also we can not do this for every chip. Non-destructive methods can be categorized by two methods: automatic test pattern generation (ATPG),

* Corresponding author.
Email addresses: `vosoughi@qiau.ac.ir` (M. Vosoughi), `jahanian@sbu.ac.ir` (A. Jahanian).
ISSN: 2008-2045 © 2015 ISC. All rights reserved.

and side channel analysis.

In Automatic Test Pattern Generation, all possible input patterns test the chip and if the desirable output is not observed, we can consider that chip may have Hardware Trojan. Authors in [2–5] have provided some techniques in this scope. In the scope of side channel detection methods, contributions such as in [6–9] are reported.

Recent years significant contributions are reported on HTH prevention, too. Authors in [10] provide a new systematic Trojan prevention and RTL code hardening scheme based on ATPG, called Design-for-Trojan-Test (DFTT). DFTT methodology replaces Trojan-vulnerable source code with hardened Trojan prevention and detection code. In [11] a novel layout-aware scan-cell re-ordering method presented to localize design switching into a specific region and improve hardware Trojan detection. One strategy in prevention methods is to fill the threatening empty resource and spaces in all phase of design. Authors in [12] use this strategy in their proposed method. Their methods augment the schedule of all functional units and assignment of all registers such that all of them are in use in every cycles.

Proposed techniques are mainly concentrated on the HTH detection and security-aware hardware design is an ongoing research topic. Moreover, prevention of the chip from this kind of attacks by securing the design can be easier and also much chipper.

In this paper, we propose a new cell/register placement algorithm that places the cells/registers of design such that hinders the synchronous HTH insertion and also simplifies the HTH detection in existing of the malicious hardware manipulation. In other words, after the proposed placement, Trojan insertion requires increases the volume and hardness of the layout processing efforts required for HTH insertion that automatically increases the probability of Trojan detection in the manipulated hardware. The semantic of this technique is similar to Image Steganography where other data is modulated into the main picture for security or other application.

In other words, we propose a CAD flow in which inserting Trojan requires considerable change in the circuit. These degree of changes, make a circuit with different properties that can be easily marked as a Trojan-contained circuit. Therefore, our approach improves Trojan detectability. It is worth noting that combinational Trojans are not the target of this research, because combinational Trojans are permanently active and can be found easily. Therefore, real Trojans are designed as a state machine to be activated in a special state in order to hide it.

The rest of this paper is organized as follows. Section 2 describes the basic concepts and classification of hardware Trojans. Section 3 illustrates the proposed security-aware register/cell placement algorithm. Experimental results are discussed in Section 4 and finally, Section 5 concludes the paper.

2 Hardware Trojans

Hardware Trojan Horse (HTH) is any deliberate modification to original circuitry inserted by adversaries to exploit a system or to use hardware mechanisms to gain access to data or software running on the chips [13]. This modification can be insertion, substitution and deletion some circuitry or just change in size of wires and logic. To facilitate the development of Trojan mitigation, detection and protection techniques, it is necessary to systematically categorize the hardware Trojans. By this categorization and taxonomy, the study of characteristics of hardware Trojans is facilitated and detection or protection methods for each class can be developed.

Authors of [14] introduced a detailed taxonomy for hardware Trojans. They decomposed the Trojans based on three main parameters; physical characteristics, activation type and action of the Trojan. This classification is shown in Figure 1.

In Figure 1, Trojans are classified based on three main characteristics as follows [13]:

- Physical characteristics contains the physical features such as Trojan size, structure and geometrical distributions.
- Activation characteristics refer to the criteria that cause a Trojan to become active and carry out its disruptive function.
- Action characteristics identifies the Trojan based on its action when the Trojan is activated.

3 Proposed Algorithm

As mentioned before, we proposed a security-aware cell/register placement in this paper in order to prevent or harden the synchronous hardware Trojan insertion. In other words, using the proposed placement, Trojan insertion requires more layout processing efforts and also increases the probability of Trojan detection in a manipulated hardware. This section describes the proposed placement algorithm that is a modified Simulated Annealing (SA) algorithm.

Simulated annealing is a powerful technique that has been widely used to solve combinatorial optimization problems using a probabilistic hill-climbing algorithm. This technique has significant ability to escape from local optimums in the search space [16]. In real

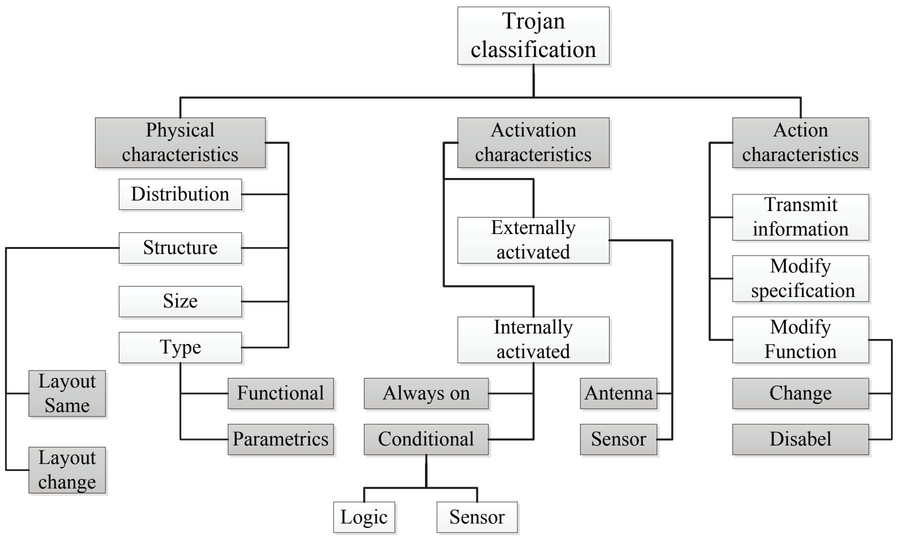

Figure 1. Detailed taxonomy showing physical, activation, and action characteristics of Trojans

annealing process, a material is lightly cooled to reach the lowest energy state. Slow cooling decreases random cells moves and makes it more stable. Slowing the cooling of the material allows each molecule to move into a place it feels most comfortable. As the material is kept at a high temperature, the molecules are able to move freely around, then when the material is cooled, the molecules are not able to move around as freely but still move limited distances [17].

Simulated annealing placement starts with a random initial placement. This placement is changed by exchanging location of two elements (Perturb). Then cost of this perturb is calculated. If the cost is decreasing then the move is accepted, but if the cost is not decreasing then the move is accepted with a probability. The probability of accepting an improper move decreases with falling the temperature. This technique helps the simulated annealing algorithm to climb out of local optimums in search for a global minimum [15]. We modified this algorithm to reach our mean.

Clock skew is an important concern in clock routing of high performance circuits. Clock skew is known as the maximum differences of clock arrival time to flip-flops. If the clock does not arrive to flip-flops at proper time, the chip would not work synchronously. We use the clock skew of the design as a signature to protect the design against hardware Trojan insertion. We presented a placement algorithm that places the flip-flops in the design in such a way that the insertion of extra flip-flops (as a part of HTH) requires many complicated layout processing and also changes the clock skew of the design, considerably. In this situation, delay characteristics of the design will be changed due to updating the design clock skew and Trojan detection will be simplified.

A widely used method for clock distribution is H-Tree construction. In this approach, clock tree is constructed such that delay from the clock source to each of the clock leaves is the same. Figure 2 shows a simplified binary H-shaped clock tree with eight leaves. We defined some regions of the chip as flip-flop feasible regions (FF_FRs) in this paper. FF_FRs are the area that inserting the flip-flops inside them, make feasible clock skew. In other words, locating the flip-flops inside the FF_FRs, guarantees that the clock skew will be lower than a specified value which is normally determined based on the timing closure specified by the designer. In Figure 2, clock tree is drawn by thin lies and FF_FRs are shown as dashed boxes. We planned the FF_FRs before placement and refine the canonical placement flow such that all the flip-flops are placed inside the FF_FRs. Algorithm 1, represents the proposed algorithm in more details.

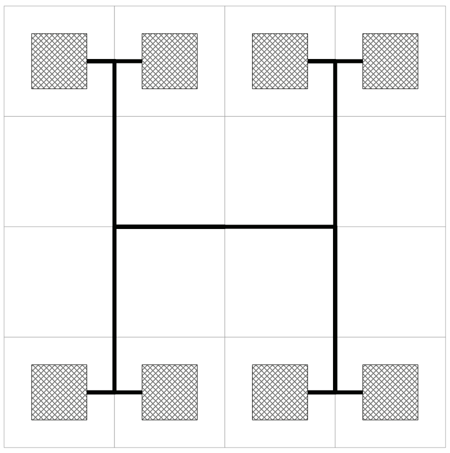

Figure 2. Canonical H-shaped clock tree

Algorithm 1 Proposed Placement Algorithm

```
Step1: Construct H-Tree on the die area and
determine the flip-flop feasible regions.
Step2: Initial placement
for all cells do
    if cell is Flip-Flop then
        Place it in a FF_FR.
    else
        Place it out of the FF_FRs.
    end if
end for
Step3: Modified SA-based placement
        Canonical SA with Modified Cost
        Modified Cost = TWL + FFD
Step4: Congestion Removal
for all cells do
    while # FF > q do
        Select a FF and Replace it to a FF_FR with
        least increase in TWL
    end while
end for
```

Step 1: In this step, the algorithm divides the die area to n equal regions by using the zero-skew H-tree clock routing algorithm. Each region represents a FF_FR. It is worth noting that total area inside the FF_FRs should be equal or larger than the total area of the design flip-flops.
Step 2: In this step, an initial random placement is done. All the flip-flops are placed inside the FF_FRs and other cells are located outside the FF_FRs.
Step 3: In this step, a modified SA (Simulate Annealing) placement algorithm is proposed to determine locations of the cells and flip flops. The main metric for SA placement algorithm is total wire length (TWL). However, we added a new cost (FFD) in our algorithm (total FF distance) representing the distance of the flip flops from the center of their corresponding FF_FR. FFD is calculated as:

$$FFD = \sum_{j=1}^{n} \sum_{j=1}^{F} d_{ij} \tag{1}$$

where n is the number of FF_FRs, F is the number of flip flops in FF_FRj and $di{,}j$ is the the distance between FFi and its corresponding FF_FRs (FF_FRj) center. Using the multi-objective cost function (consisting of total wire length and total FF distance), total wire length of the design is minimized and also flip flops are forced to place as close as to the center of their FF_FRs.
Step 4: After the SA-based placement, all the cells are placed as well as the flip flops and the flip flops are forced to be located closer to their FF_FRs as much as possible. This force causes to generate some congested FF_FRs in the design. In other words , many of the FF_FRs may contain more flip flops than some other FF_FRs. In this step we balance the congestion of all FF_FRs. At first, the maximum of allowed flip flops for each FF_FR is calculated as:

$$q = \lceil \frac{\#FF}{n} \rceil \tag{2}$$

where q is the maximum capacity of flip flops for each FF_FR, $\#FF$ is the number of flip flops in design and n is the number of FF_FRs. It is worth noting that by the use of H-Tree construction, n can be calculated as:

$$n = 2^i \quad for \quad (i \leq 2) \tag{3}$$

If the number of flip flops in a FF_FR is more than q, the algorithm selects a flip flop from this FF_FR and places it to a FF_FR in which the number of its containing flip flops are less than q. This replacement method should be done with aware of least increase in TWL.
After the proposed placement and congestion refinement, all the cells are placed and flip-flops are accumulated inside the FF_FRs. It should be noted that there is no white space between the flip-flops in the FF_FRs. After placement of the cells and flip flops according to the proposed algorithm, flip flops are concentrated to their FF_FRs and clock routing will result a good clock wire length under a skew constraint.

In this situation, if an adversary wants to insert HTH into the circuit, he/she should find some white spaces to place the HTH. As there is no white space in the FF_FRs, HTH should be placed outside of these regions. Therefore, the clock will arrive to HTH flip-flops with a different skew. If adversary wants to reach to the proper skew (for example zero skew) with any bounded skew clock routing algorithm (like BST/DME), he will face a high clock wire length. These differences in clock behavior, cause to facilitate the HTH detection.
We will show in our experiments that insertion of extra flip-flops will result a significant increase in clock wire length with a bounded skew (for example zero skew).

When the clock wire length increases significantly, the clock routing would be hardened and may be impossible. Even if the adversary wants to decrease the clock wire length, the clock skew for his flip-flops would be high and the HTH will not act properly. So, with this method of placement, we hardened the possibility of HTH insertion in design.

4 Experimental Results

We implemented the proposed placement algorithm in the EduCAD tool [18], a Linux-based educational physical design platform with ten benchmarks from IWLS benchmark suite [19] to evaluate the benefits and overheads of the proposed algorithm. Table 1 shows the statistical properties of the attempted benchmarks. In this table, columns *#cells*, *#nets* and *#pins* represent the number of cells, nets and pins, respectively.

Table 1. Statistical characteristics of attempted benchmarks

BM	#Cells	#FFs	#Nets	#Pins
s5378	1294	163	1333	88
spi	3227	229	3276	94
des_area	4881	128	5123	306
s38584	6724	1178	6743	294
tv80	7161	350	7180	48
systemcase	7959	654	8221	391
s38417	8278	1564	8310	138
mem_ctrl	11440	1082	11560	269
usb_funct	12808	1746	12955	251
dma	19118	1797	19807	950

We considered two different design flows. At the first flow, benchmarks are placed by EduCAD with canonical SA-based placer and the second flow utilizes the modified SA. Then the list of flip flops locations are extracted of the placed circuit in each flow and clock tree of each benchmark circuit is constructed. The goal of most of the current clock routing algorithms is on achieving zero-skew at the expense of longer wire length, resulting in high power dissipation. However in practice, circuits still operate correctly within a tolerable skew bound. Therefore, in order to reduce clock net power dissipation, authors in [20] believe that the clock routing algorithm should consider bounded-skew trees (BST) instead of zero-skew trees (ZST) and proposed a clock routing algorithm based on this claim which they named BST/DME. BST/DME minimizes total clock wire length under any given path-length skew bound.

We used BST/DME in order to construct the clock tree. After clock routing, clock skew and clock wire length is extracted. Then some flip flops are inserted to the design as Hardware Trojan in random locations in both design flows. The number of inserting HTH flip flop for each design is 5% of the number of original FF in design. These two flows are shown in Figure 3.

Table 2 represents the obtained clock tree wire length by using BST/DME tool with zero skew. Column *#FFs* shows the number flip flops and columns *Without HTH* and *With HTH* represent the clock wire length in two canonical and proposed design flows. As shown in Table 2, overhead of clock tree wire length in the proposed placement is about 20% more than the canonical placement. It means that after using the proposed placement, Trojan insertion requires considerable changes in clock tree wire length that highlights

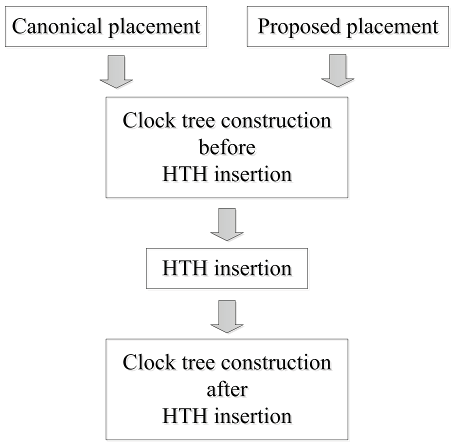
Canonical placement
Proposed placement
Clock tree construction before HTH insertion
HTH insertion
Clock tree construction after HTH insertion

Figure 3. Proposed design flow vs. canonical design flow

the Trojan-contained circuit. In this circuit, Trojan will be more detectable.

As we expect, increase of CLKWL by the existence of HTH FF in our modified SA is a significant percentage, but in canonical SA is negligible. So, if an adversary wants to insert HTH in a design placed with our proposed algorithm, increase in CLKWL is a big obstacle for him. Therefore, inserting HTH in a die placed by our proposed algorithm would be hardened.

Table 3 represents the wire length overhead of our modified SA placement algorithm. In this table, two first columns show the TWL in canonical SA and column three shows the TWL in our modified SA and the last column represents the TWL overhead of our modified SA placement algorithm in compare to canonical SA. We see that our modified SA cost is not a significant overhead.

Table 4 shows the delay overhead of our modified SA placement algorithm. In this table, we present the delay increase of our method by comparing with canonical SA. As can be seen in Table 4, the delay overhead of the proposed algorithm is not considerable.

It is noting that our placement is fixed-die. Therefore, chip area does not change during the placement.

5 Conclusion

In this paper, we proposed a new placement algorithm that hinders the hardware Trojan insertion or simplifies the detection process in existence of Trojans. In the presented algorithm, cells/registers of design are placed such that HTH insertion will be hard and also detection of the Trojans will be simplified when design is modified by attackers. Experimental results show that the proposed placement improves the immunity of attempted benchmarks against Trojan insertion. These improvements are gained in cost of 6.4% delay (on average) and 9.8% wire length (on average)

Table 2. Comparison of the proposed and canonical placement algorithms in terms of clock wire length

BM	Clock wirelength				Clock wirelength overhead (%)	
	Canonical placement		Proposed placement			
	Without HTH	With HTH	Without HTH	With HTH	Canonical placement	Proposed placement
s5378	4188	4238	2605	2852	1.19	9.48
spi	7236	7301	3883	4576	0.90	17.85
des_area	5677	5907	2343	2905	4.05	23.99
s38584	29266	30187	22141	24094	3.15	8.82
tv80	13164	13294	6459	8298	0.99	28.47
systemcase	20687	21420	13249	16735	3.54	26.31
s38417	37405	38431	29766	32608	2.74	9.55
mem_ctrl	31621	32877	20340	24509	3.97	20.50
usb_funct	45275	46637	35559	41286	3.01	16.11
dma	54995	56500	38207	54071	2.74	41.52
Average					**2.63%**	**20.26%**

Table 3. Experimental results in terms of total wire length overhead

BM	TWL in Canonical placement	TWL in Proposed placement	TWL Overhead (%)
s5378	116346	139858	20.2
spi	435216	477714	9.8
des_area	729872	758286	3.9
s38584	1462671	1647886	12.7
tv80	1391814	1454958	4.5
systemcase	1681310	1774713	5.6
s38417	2511486	2864596	14.1
mem_ctrl	2826223	3060943	8.3
usb_funct	3630536	4011246	10.5
dma	6909107	7468692	8.1
Average			**9.8%**

Table 4. Delay overhead of the proposed placement vs. canonical placement

BM	Delay of Canonical placement	Delay of proposed placement	Delay Overhead (%)
s5378	1669	1705	2.2
spi	3835	4270	11.3
des_area	5233	5200	-0.6
s38584	4654	4987	7.2
tv80	7153	7213	0.8
systemcase	8390	8181	-2.5
s38417	4833	5402	11.8
mem_ctrl	5320	5793	8.9
usb_funct	4309	4962	15.2
dma	7264	8010	10.3
Average			6.4%

overheads.

References

[1] S. Adee,“The Hunt for the Kill Switch, ”In *IEEE Spectrum*, vol. 45, no. 5, 2008, pp. 34-39

[2] S. Jha and S.K. Jha, “Randomization Based Probabilistic Approach to Detect Trojan Circuit, ”Proceedings 11th IEEE High Assurance Systems Engineering Symposium, 2008, pp. 117-124.

[3] F. Wolff, C. Papachristou, S. Bhunia and R. Chakraborty, “Towards Trojan Free Trusted ICs: Problem Analysis and Detection Scheme, ”In *Proceedings Design, Automation and Test in Europe Conf*, 2008, pp. 1362-1365.

[4] M. Banga and M. Hsiao, “A Region Based Approach for the Identification of Hardware Trojans, ”In *Proceedings IEEE International Workshop Hardware-Oriented Security and Trust*, 2008, pp. 40-47.

[5] M. Banga and M. Hsiao, “A Novel Sustained Vector Technique for the Detection of Hardware Trojans, ”Proceedings 22nd International Conf. VLSI Design, 2009, pp. 327-332.

[6] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi and B. Sunar, “Trojan Detection Using IC Fingerprinting, ”In *Proceedings IEEE Symposium*

Security and Privacy, 2007, pp. 296-310.

[7] X. Wang, H. Salmani, M. Tehranipoor and J. Plusquellic, “Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis, ”In *Proceedings IEEE International Symposium Defect and Fault Tolerance of VLSI Systems*, 2008, pp. 87-95.

[8] R. Rad, X. Wang, M. Tehranipoor and J. Plusquellic, “Power Supply Signal Calibration Techniques for Improving Detection Resolution to Hardware Trojans, ”In *Proceedings IEEE/ACM International Conference Computer-Aided Design*, 2008, pp. 632-639.

[9] J. Li and J. Lach, “At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection, ”In *Proceedings IEEE International Workshop Hardware-Oriented Security and Trust*, 2008, pp. 8-14.

[10] Y. Jin, N. Kupp and Y. Markis,“DFTT: design for Trojan test, ”In *Proceedings IEEE International Conference on Electronics, Circuits, and Systems*, 2010, pp. 1681-1171

[11] H. Salmani, M. Tehranipoor, and J. Plusquellic, ”Layout-aware scan-cell re-ordering for improving localized switching to detect hardware Trojans, In *Journal of Low Power Electronics (JOLPE)*, 2012

[12] M. Potkonjak, “Synthesis of Trustable ICs using Untrusted CAD Tools, ”In *Design Automation Conference*, 2010, pp. 633-634.

[13] M. Tehranipoor and F. Koushanfar, “A Survey of Hardware Trojan Taxonomy and Detection, ”In *IEEE Design & Test of Computer*, 2010, pp. 10-25.

[14] X. Wang, M. Tehranipoor, and J. Plusquellic, “Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions, ”In *Proceedings IEEE International Workshop Hardware-Oriented Security and Trust*, 2008, pp. 15-19.

[15] S. K. Lim, “Practical problems in VLSI physical design automation, ”In *Springer Pub.*, 2008.

[16] P. Banerjee, M.H. Jones and J.S. Sargent, “Parallel simulated annealing algorithm for cell placement on hypercube multiprocessors, ”In *”In IEEE Transactions on parallel and distributed systems*, 1990, Vol.I, No.1.

[17] J. A. Chandy, S. Kim, B. Ramkumar, S. Parkes, and P. Banerjee, “An evaluation of parallel simulated annealing strategies with application to standard cell placement, ”In *In International Parallel Processing Symposium and International Conference on VLSI Design*, 1996.

[18] S. Amanollahi and A. Jahanian, “EduCAD: an efficient, flexible and easily revisable physical design tool for educational purpose, ”In *Design, Automation and Test in Europe, University booth*, 2011.

[19] IWLS 2005 Benchmarks, Available on http://www.iwls.org/iwls2005, 2005.

[20] J. Cong and C. Koh, “Minimum-Cost Bounded-Skew Clock Routing, ”In *Proceedings IEEE International Symposium on Circuits and Systems*, 1995, pp. 215-218.

Mehrshad Vosoughi received his B.Sc. degree in computer engineering from Islamic Azad University, south Tehran Branch, Tehran, Iran in 2010, and the M.Sc. degree in computer system architecture from Azad University, Qazvin Branch, Qazvin, Iran in 2012. His current research interests are hardware security and automatic computer-aided design.

Ali Jahanian received B.Sc. degree in computer engineering from University of Tehran, Tehran, Iran in 1996 and the M.Sc. and Ph.D. degrees in computer engineering at Amirkabir University of Technology, Tehran, Iran in 1998 and 2008, respectively. He joined Shahid Beheshti University, Tehran, Iran in 2008. His current research interests are VLSI design automation and automotive embedded system design.