A Low Cost Acceleration Method for Hardware Trojan Detection Based on Fan-out Cone Analysis

Bin Zhou[1,3], Wei Zhang[2], Srikanthan Thambipillai[1], J.K.J.Teo[1]

[1]School of Computer Engineering, Nanyang Technological University, Singapore {bzhou,astsrikan}@ntu.edu.sg

[2]Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China wei.zhang@ust.hk

[3]Research Center of Basic Space Science, Harbin Institute of Technology, China {zbhit}@hit.edu.cn

ABSTRACT

Fabless semiconductor industry and government agencies have raised serious concerns about tampering with inserting hardware Trojans in an integrated circuit supply chain in recent years. In this paper, a low hardware overhead acceleration method of the detection of HTs based on the insertion of 2-to-1 MUXs as test points is proposed. In the proposed method, the fact that one logical gate has a significant impact on the transition probability of the logical gates in its logical fan-out cone is utilized to optimize the number of the insertion MUXs. The nets which have smaller transition probability than the threshold value set by the user and minimal logical depth from the primary inputs are selected as the candidate nets. As for each candidate net, only its input net with smallest signal probability is required to be inserted the MUXs based test points until the minimal transition probability of the entire circuit is no smaller than the threshold value. Experiment results on ISCAS'89 benchmark circuits show that our proposed method can achieve remarkable improvement of transition probability with small overhead penalty.

Categories and Subject Descriptors

B.7.m [**Integrated Circuits**]: Miscellaneous; B.6.2 [**Reliability and Testing**]: Testability.

General Terms

Algorithms, Design, Experimentation, Security

Keywords

Hardware Trojan; Fan-out cone; Low cost; Signal probability; Transition probability

1. INTRODUCTION

Outsourcing design and fabrication process of the integrated circuit (IC) has given opportunities for some adversary within the supply chain to tamper the IC design by maliciously implanting extra logic as hardware Trojan circuitry into an IC [1]. The hardware Trojan (HT) circuitry can be inserted during the design, fabrication, and manufacturing process in the form of additional gates or resized gates compared with the design specification [2], which enables an adversary to control, monitor, spy contents and communications, or to remotely activate/disable parts of the IC. Since the ICs form the core computing and communication kernels for the high security areas, such as avionics, defense, communications, industrial and so on, ensuring IC trust in the presence of an untrusted design flow is of paramount importance.

However, since HTs are most likely connected to the nets with low controllability and/or observability that they can be only triggered in rare conditions, the detection of HTs becomes an extremely challenging problem and the existing Automatic Test Pattern Generation (ATPG) algorithms are not effective for the detection of HTs. Moreover, it can be expected that the inputs of HTs are supplied by the nets with low transition probabilities, which will result in small impacts on circuit side-channel signals such as power and delay [1, 3]. Hence, it is also challenging to detect HTs through side-channel analysis. Nonetheless, in order to ensure the hardware security, efforts have been made through different approaches proposed in recent years. These methods can be generally characterized into two categories: side-channel analysis methods [4-9] and logic testing [1, 10-12].

Side-channel signal analysis method tries to detect hardware Trojans by measuring circuit parameters, such as power (transient and leakage current) and delay, and compare them with the correct results. This method faces great challenge that due to the low activity nature of HTs, the impacts of HTs may be masked by the process variation and environmental noise, and hence, hard to be detected by the side-channel analysis. The approach of logic testing tries to apply test patterns to activate HTs and compare the responses with the correct output results. As we have mentioned, since HTs can only be triggered in rare conditions, the random test patterns or the deterministic test patterns generated by the traditional ATPG tool do not provide high detection coverage.

However, there is an efficient way to enhance the detection probability of HTs through increasing the transition probability of each net by inserting dSFF/test point [1]. However, as the techniques are based on the insertion of special circuitry, e.g., dSFFs/test points, the circuit delay, power, and area will be deteriorated. Therefore, how to minimize the overhead of test point insertion while achieving the best improvement of activity probability of HTs is the key problem to the techniques.

In this paper, we propose a low-cost yet efficient technique to increase the net transition probability and accelerate the HT

ESWEEK'14, October 12 - 17 2014, New Delhi, India
Copyright 2014 ACM 978-1-4503-3051-0/14/10…$15.00.
http://dx.doi.org/10.1145/2656075.2656077

detection based on the insertion of 2-to-1 MUXs as test points. In the proposed method, we utilize the fact that the transition probability of one logical gate remarkably influences the transition probability of the logical gates in its logical fan-out cone to select the best candidate nets which cover the most nets with minimal transition probability for test point insertion. The insertion point is then added to the input nets of the candidate nets to best improve the transition probability of the candidate nets and its fan-out nets. Compared to the previous method, our method is capable to more efficiently improve the transition probability of the circuit and minimize the number of insertion points. Moreover, through using MUX based test point, our test architecture is simplified to further reduce the layout complexity and hardware overhead. In addition, we also applied the weighted test pattern technique which can not only further optimize the number of the inserted points, but also prevent the adversary from guessing out the input pattern. Our simulation results on large benchmarks demonstrate that the proposed method is capable to improve the transition probability of the entire circuit significantly with neglectable delay, and area overhead.

The left of this paper is organized as follows. Section 2 discusses the related works in HT detection. The background of the proposed method is introduced in Section 3. In Section 4, a motivation example is presented to illustrate the basic idea of our approach and show the difference between our work and previous work. The proposed method including the test architecture and selection algorithm is discussed in detail in Section 5. Section 6 presents the simulation results for the proposed method, and finally Section 7 concludes the work.

2. RELATED WORKS

Side-channel analysis has been widely used in HT detection. The methods developed in [4-6] use path delay as a side-channel signal to detect HTs. In [4], maximizing the resolution of each path delay measurement is executed to increase a HT's impact. It generates a vector that sensitizes the shortest path passing via the HT's site. In [5], path delays of nominal chips are collected to form a series of fingerprints, each one representing one aspect of the total characteristics of a genuine design. By comparing chips' delay parameters to the fingerprints, HTs can be detected. In [6], clock sweeping is used to obtain the critical and noncritical path delay and then generate signatures for ICs for the purpose of detecting HTs. The methods developed in [7-9] use transient power as a side-channel signal to detect Trojans. In [7], the authors claim HTs detection resolution depends on the Trojan activity directly and circuit activity reversely. The proposed method reorders scan cells based on their placement to reduce circuit switching activity by limiting it into a specific region. It helps magnify HTs contribution to the total circuit transient power, and improve the efficiency of power-based side-channel signal analysis techniques for HTs detection. In [8], a power supply transient signal analysis method based on the analysis of multiple power port signals is proposed to detect HTs. [9] proposed a sustained vector methodology based on repeating each vector for multiple times. In addition, gate-level characteristics [13-15] can be used to detect hardware Trojans. Delay and power characteristics of each gate of a design subjected to process variations can be individually extracted. Linear programming is used to solve a system of equations created using non-destructive measurements of power and delays.

Logic testing is another common approach to detect HTs by applying test patterns to activate the hidden HTs. In [10], the authors analyze and formulate the Trojan detection problem based on a frequency analysis under rare trigger values and provides procedures to generate input trigger vectors and Trojan test vectors to detect Trojan effects. In [11], a test pattern generation technique based on multiple excitation of rare logic conditions at internal nets is proposed. A design methodology called On-Demand Transparency was proposed in [12] where special circuitry, like state machine was embedded in an IC to improve the controllability and observability of internal nets during "transparent mode". Recently, the work in [1] proposed an efficient method which inserts a dummy scan flip-flop (dSFF) in some candidate nets with low transition probabilities to shorten transition generation time. The method successfully increases the low transition probability nets to above threshold, which in turn improves the probability to detect HTs. However, the number of insertion points required is still high, incurring relatively large overhead.

3. BACKGROUND

Increasing the transition probability (*tp*) of each net in the circuit is an efficient way to enable the detection of HTs due to two folds of reasons: (1) It can improve the power-based side-channel signal analysis methods by increasing the number of transitions in Trojans. (2) It provides an opportunity for fully activating a Trojan circuit and observing the erroneous response at the circuit's output.

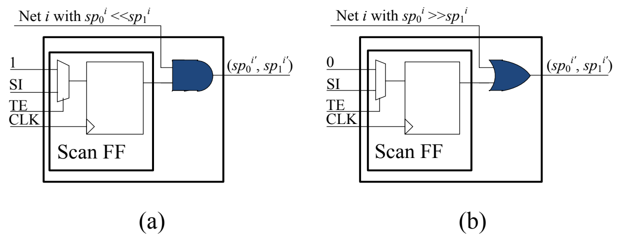


Fig. 1 The dSFF structures when (a) $sp_0^i << sp_1^i$ and (b) $sp_0^i >> sp_1^i$

Table 1 The computation rule of the transition probability

Gate Type	**Computation Rule**
AND/NAND	$tp^{out} = \prod_{i=1}^{n} sp_1^{in^i} \cdot \left(1 - \prod_{i=1}^{n} sp_1^{in^i}\right)$
OR/NOR	$tp^{out} = \prod_{i=1}^{n} sp_0^{in^i} \cdot \left(1 - \prod_{i=1}^{n} sp_0^{in^i}\right)$
MUX with 2 inputs	$sp_1^{out} = sp_1^{in^1} \cdot sp_0^{in^3} + sp_1^{in^2} \cdot sp_1^{in^3}$ $sp_0^{out} = 1 - sp_1^{out}$ $tp^{out} = sp_1^{out} \cdot sp_0^{out}$
NOT	$tp^{out} = sp_0^{in} \cdot sp_1^{in}$

In [1], a dummy scan flip-flop shown in Fig.1 is inserted in the candidate nets to improve the transition probability of its fan-out nets and in turn reduce the total number of the nets with low transition probability. If signal probability of "0" on candidate net Net$_i$, sp_0^i is less than its signal probability of "1", sp_1^i, an AND gate is connected to the output of the scan flip-flop and net Net$_i$ to increase sp_0^i, as depicted in Fig. 1(a). Similarly, if sp_1^i is less than sp_0^i, an OR gate is used to increase sp_1^i, as in Fig. 1(b). When TE is active (test mode), the output of scan flip-flop is supplied by SI. If random test patterns are applied, the signal probabilities of "1" and "0" at the output of scan flip-flop are 0.5. In this way, $sp_1^{i'}$ is reduced up to $0.5\times sp_1^i$ for the case shown in Fig. 1(a). Similarly, $sp_0^{i'}$ is reduced up to $0.5\times sp_0^i$ for the case shown in Fig. 1(b). Thus, after dSFF insertion, the transition probability of the fan-out nets can be improved.

In order to identify the candidate nets to insert a dSFF test point, the work in [1] also proposed an insertion algorithm. The algorithm divided all the nets into two groups: Nets with transition probability higher than the threshold and nets with transition probability lower than the threshold. In order to increase the transition probability of the nets in the second group, the nets in the first group are first sorted and then one net is selected to insert dSFF in an iteration. Transition probability of nets after dSFF insertion is calculated again to obtain the number of low transition nets. If the new number is less than the number before this round of dSFF insertion, the inserted dSFF is kept. Otherwise, the dSFF would be ignored. The procedure continues until there is no net in the second group (*i.e.*, there is no net with lower transition probability than the transition probability threshold), or the number of nets in the second group remains unchanged during iterations (*i.e.*, adding insertion points in the first group cannot further improve the transition probability of the nets in the second group). The computation rule of the transition probability is shown in Table 1. Herein $sp_0^{in^i}$ and $sp_1^{in^i}$ represent the signal probability being "1" of the *i*th input and the signal probability being "0" of the *i*th input, respectively. According to the computation rule, we can obtain the largest transition probability for one signal net, i.e. 0.25, only when its signal probabilities being "0" and "1" are equal. In other words, improving the transition probability for one signal net corresponds to making its signal probability be close to 0.5.

4. MOTIVATION

The method in [1] showed an effective improvement on the transition probability of the nets in the benchmark circuits. However, there is great potential to further improve this HT detection acceleration method in terms of efficiency and overhead. Observing that in the original method in [1], the candidate net selected to insert the dSFF point is based on its own transition probability without considering the logical connection between the candidate net to the nets with low transition probability, the insertion of dSFF points in the candidate nets does not guarantee the largest reduction of the number of nets in the second group. Hence, it may result in large number of required insertion points, which in turn causes large overhead on area and delay. In addition, for large TP_{th}, the number of the nets in the first group with high transition probability will be very large. Traversing the first group with an evaluation for each trial insertion of dSFF point will lead long simulation time.

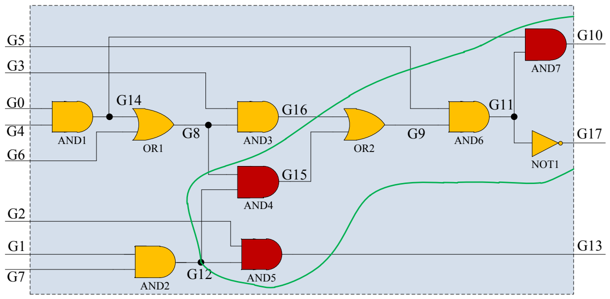


Fig.2 An example circuit

Next, we will illustrate our basic idea of proposed method through a simple example. Fig. 2 shows an example circuit. Assume sp_1 of each primary input is 0.5, we can get the signal probability of each net according to the computation rules of logical gates. The computation results are shown as Table 2. From Table 2, the nets G10, G13, and G15 have minimal transition probability when the threshold of transition probability TP_{th} is set to 0.15 by user. According to the dSFF insertion method in [1], the net group with transition probability higher than transition probability threshold consists of {G11,G17,G14,G12,G16,G8,G9} and the net group with transition probability smaller than transition probability threshold consists of {G10,G13,G15}. Considering the method in [1], two possible orders can be applied to sort the nets in the first group. When the ascending order is applied, two dSFFs are required to insert on the nets G14 and G15 after five iterations to improve the transition probability of the nets G15 and G10. However, as for the net G13, although insertion dSFF on the net G12 can improve its transition probability, the transition probability of the net G9 become smaller than TP_{th}. Therefore, the insertion dSFF on the net G12 will be discarded according to [1]. In other words, there is no existing insertion position to improve the transition probability of the net G13. When the descending order is applied, two dSFFs are required to insert on the nets G12 and G14 after eleven iterations to improve the transition probability of the nets G15, G13, and G10. We can see that the running time is long and the optimal solution cannot be found in some cases.

Table 2 The experimental results of the example circuit after inserting dSFFs

Net	Original tp_0	tp_1 after inserting 1st dSFF	tp_2 after inserting 2nd dSFF	$\frac{tp_2-tp_0}{tp_0}\%$
G10	0.049735	0.067356	0.172257	246.35
G13	0.109375	0.214844	0.214844	96.43
G15	0.131836	0.238037	0.249939	89.58
G11	0.165877	0.206121	0.228649	37.84
G17	0.165877	0.206121	0.228649	37.84
G14	0.187500	0.187500	0.187500	0.00
G12	0.187500	0.187500	0.187500	0.00
G16	0.214844	0.214844	0.241211	12.27
G8	0.234375	0.234375	0.152344	-35.00
G9	0.243587	0.243430	0.206834	-15.09

Our method proposes to identify the insertion point based on fan-out cone analysis. From the above example, we can find out the nets G10, G13 and G15 locates in the logical fan-out cone of G12. If changing the transition probability of the net G12 must lead to variations of transition probability of the nets located in its logical fan-out cone. That is to say, the transition probability of G12 has influence on those of G10, G13, and G15. Based on this analysis, we insert one dSFF on the net G12. The transition probabilities of G10, G13 and G15 are improved up to 0.067356, 0.214844 and 0.238037 from 0.049735, 0.109375 and 0.131836, respectively. Now only G10 has smaller transition probability than TP_{th}, so another dSFF is required. Observing the gate AND7 corresponding to G10, we can see that changing the transition probability of one input net (G11 and G14) of G10 must lead to some variations of transition probability of G10. Because the low transition probability of G10 results from the low signal probability of "1", correspondingly, we need to reduce its probability at "1". As for AND7, we can get two insertion points G14 and G11, selecting the net with smaller signal probability being "1" as the insertion point will lead to better results. We will see the detail proof of selection rules in the next Section. In this way, only two dSFFs are inserted on the net G12 and G14 after two iterations.

The comparison of transition probability before and after the dSFF insertion is shown in Table 2. There is an improvement ranging from 0% to 246.35% for all candidate nets except for G8 and G9. However, the transition probabilities of G8 and G9 are still larger than TP_{th}. Comparing with the method in [1], the runtime of our proposed method based on fan-out analysis can be significantly reduced due to the reduction of number of iterations. More importantly, as our proposed method employs fan-out analysis to optimize the selection of insertion nets, the number of the inserted dSFFs is minimized, which in turn minimize the hardware, delay, and power overhead as shown in the experiments.

5. PROPOSED METHOD

Based on the simple example described in Section 4, we can discuss our proposed method from two parts: the proposed test architecture and the selection algorithm. In the test architecture subsection, we will give an implementation of detection of HTs based on scan test techniques. Following it, we will theoretically prove why our test architecture works and describe the optimal selection algorithm which manages to add the minimal set of 2-to-1 MUXs to achieve the maximum improvement of transition probability.

5.1 Test Architecture

In the aspect of the test architecture, the original structure of dSFF in Fig. 1 does not have a uniform structure and the structure of dSFF must be selected according to the relationship between the signal probabilities being "1" and "0". One input of the 2-to-1 MUX in dSFF must be connected to power or ground line to avoid changing the function of the circuit. These requirements may incur the extra layout cost.

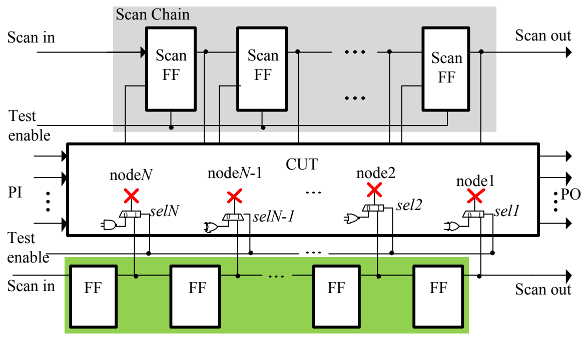


Fig.3 The proposed test architecture for detection of HTs

Fig.3 shows our proposed test architecture for detection of HTs. Assume that the scan techniques are applied to implement the structural/functional testing and all scan flip-flops are configured into one scan chain. In order to detect HTs, we will insert one 2-to-1 MUX on one input net of each candidate net to control the logical value of the candidate net. Hence for each candidate net, a 2-to-1 MUX is required. One flip-flop is added to provide the control logical value for each candidate net as shown in the green area of Fig.3.

When the test enable (TE) signal is low, the circuit is set to the normal functional mode. Unlike the method of inserting dSFFs in [1], where the output of the inserted scan flip-flop must be supplied by the deterministic logical value depending on the gate type at the output to avoid changing the circuit functionality in normal functional mode, here the output of the inserted flip-flop can be supplied by any logical value without any modification of the circuit functionality. When the test enable (TE) signal is high, the circuit is set to the test mode. The test patterns are applied to the circuit through the scan chain. In this way, the transition probability of the candidate nets can be improved during the test mode, and the function of the circuit will not be affected during the normal functional mode. It is obvious that the insertion of 2-to-1 MUXs into the logical paths may result in the delay, and area overhead. In the following subsections, we will describe the optimal selection algorithm which manages to add the minimal set of 2-to-1 MUXs while achieving the maximum improvement of transition probability.

5.2 Analysis of improvement of transition probability

In this subsection, we first give a theoretical analysis on why inserting a 2-to-1 MUX in the input nets of a candidate net can improve the transition probability of the candidate net.

In order to improve the transition probability of candidate net Net_i, whose transition probability is lower than the transition probability TP_{th}, the difference between its signal probability being "1" and "0" should be reduced. In order to achieve the goal, its input net which has largest influence on the transition probability of the candidate net should be selected to insert a 2-to-1 MUX. Assume one candidate net Net_i has N inputs. We select one of its input to insert a 2-to-1 MUX based on the selection rule shown in Table 3. Here we assume all the gates in the circuits have been decomposed into the basic gate type.

Table 3 Selection rules of input net with smallest signal probability

Gate type	$sp_1^i >> sp_0^i$, $sp_1^i << sp_0^i$
AND/ NAND	Input net with smallest sp_1
OR/ NOR	Input net with smallest sp_0

Next, we will take AND gate type as an example to demonstrate the improvement of the transition probability after applying our proposed method (see Fig.4). As for the other gate types, the derivation process is similar. Assume that the signal probability on the kth input of one candidate net Net_i are $sp_0{}^k$ and $sp_1{}^k$. The signal probability being "1" of the candidate net Net_i before applying our proposed method can be computed by

$$sp_1{}^i = \prod_{k=1}^{N} sp_1{}^k \qquad (1)$$

Assuming the jth input of the candidate net Net_i has the smallest sp_1, after applying insertion of 2-to-1 MUXs on the jth input of the candidate net, the signal probabilities on the jth input, both $sp_0{}^{j'}$ and $sp_1{}^{j'}$, are equal to 0.5 because of applying the random test patterns. In this way, the updated signal probability of the candidate net Net_i can be computed by

$$sp_1{}^{i'} = sp_1{}^{j'} \times \prod_{k \neq j}^{N} sp_1{}^k \qquad (2)$$

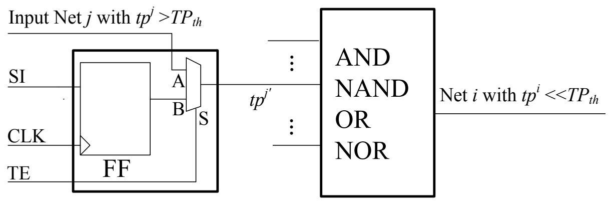

Input Net j with $tp^j > TP_{th}$
SI
CLK
TE
FF
A
B
S
$tp^{j'}$
AND
NAND
OR
NOR
Net i with $tp^i << TP_{th}$

Fig.4 The inserted circuit in the proposed method

Since the candidate net Net_i has original transition probability smaller than TP_{th}, there are two cases needed to be considered. Case 1: $sp_1{}^i >> sp_0{}^i$. $sp_1{}^i >> sp_0{}^i$ indicates that each input net $sp_1{}^k >> 0.5$. That is to say, $sp_1{}^j > sp_1{}^{j'}$. According to (1) and (2), we can get $sp_1^{i'} < sp_1^i$, which means sp_1^i is reduced after applying our proposed method. Also, according to our selection rule, sp_1^j is the smallest among all the inputs. Hence, other inputs are in the range of $1>sp_1^k>>0.5$. So $\prod_{k\neq j}^{N} sp_1^k$ is made closest to 1 and $sp_1^{i'}$ closest to 0.5 where we can get the highest transition probability at Net_i. Case 2: $sp_1^i << sp_0^i$. Same rule can be applied that we make the input net with smallest sp_1 to be 0.5 by inserting a 2-to-1 MUX. Then $sp_1^{i'}$ is closest to 0.5 and corresponding $sp_0^{i'}$ is reduced to near 0.5.

```
Input: Net-list and $TP_{th}$
Output: Updated Net-list after inserting MUXs

SELECTION(Net-list, $TP_{th}$){
 1: Logical_Depth ← ComputeLogicalDepth(Net-list)
 2: SP ← ComputeSignalProbability(Net-list)
 3: TP ← ComputeTransitionProbability (SP)
 4: Nets ← ReorderNet(Net-list,TP,ascend)
   /* Reorder nets in the ascending order of transition probability */
 5: MINTP ←SelectNetWithTransitionProbability(Nets, $TP_{th}$)
   /*Select nets with transition probabilities smaller than $TP_{th}$ to form
    MINTP*/
 6: I_MINTP←GetInputNet(MINTP, Net-list, SP)    /* Get the input of
    each net in MINTP with minimal signal probability to form I_MINTP
    */
 7: I_MINTP←ReorderNet(I_MINTP, Logical_Depth, Ascend)     /*
    Reorder nets in I_MINTP in the ascending order of logical depth
    from PI */
 8: MINLD ←GetNetMinLogicalDepth(I_MINTP, Logical_Depth)
    /*Get the nets which have the smallest logical depth.
 If #MINLD>1
    foreach net $n_i \in$ MINLD
       8: $\varphi_{ni}$ ←GetFanoutCone(Net-list, $n_i$)
       9: $V_{ni}$ ←GetCoverVector(Net-list, $\varphi_{ni}$)
          /*Constitute one cover vector representing whether one net is
          located in the logical fan-out cone of one net $n_i$ or not
          according to the set $\varphi_{ni}$*/
       10: $NO_{ni}$ ←ComputeNumberCoveredNet($v_{ni}$)
          /*Compute the number of nets of MINTP covered by the cover
          vector of net $n_i$*/
    11: $n_k$ ←GetTargetNet1(MINLD, NO)
       /*Get $n_k$ whose cover vector includes the largest number of nets
       in MINTP */
 Else
    12: $n_k$ ←GetTargetNet2(MINLD)
 13: Insert2-to-1MUX($n_k$)
 14: Net-list ←UpdateNetlist(Net-list)
}
```

Fig.5 The proposed selection algorithm

The similar analysis can be applied to other basic type of gate. Based on the above analysis, we can conclude that our proposed method based on the insertion of 2-to-1 MUXs on the input net of the candidate net can guarantee that the transition probability of the candidate net is improved. Note that if a test point is directly inserted into the candidate net, there will be problem that the candidate net is broken into two pieces. Only the second half net after the insertion point will obtain high transition probability, while there always exists a first half net remaining with low transition probability.

5.3 Selection Algorithm

In this subsection, we will describe our proposed MUXs insertion algorithm in detail. The objective of the insertion algorithm is to add the minimal set of 2-to-1 MUXs to achieve the maximum improvement of transition probability.

The proposed selection algorithm is shown in Fig.5. It takes the circuit net-list composed of basic gate type and threshold of transition probability TP_{th} as input and generates the updated circuit net-list with insertion points as output. As can be seen from the proof described in the last subsection, inserting a 2-to-1 MUX on the input net of a candidate net with small transition probability can directly improve its transition probability, all nets with transition probability lower than TP_{th} are selected as the candidate nets and form the set *MINTP*. Based on our selection rule for input net insertion, for each net in the set *MINTP*, its input net with smallest signal probability is selected to construct the set *I_MINTP*, and the nets in *I_MINTP* is sorted in the ascending order of logical depth from the primary. We have known that inserting a test point in a net will improve the transition probability of its fan-out cone. In order to minimize the number of required total insertion points, in every iteration, we select the first net in *I_MINTP* with minimum logic depth as the best candidate net to insert a 2-to-1 MUX, because it does not locate in the fan-out cone of any other nets and hence, one insertion point is required to increase the transition probability of its output net. Moreover, inserting in the net with small logic depth in *I_MINTP* will potentially influence more nets of *MINTP*. When there are multiple nets in *I_MINTP* with same logic depths, the net whose logical fan-out cone includes the largest number of nets of *MINTP* is selected. To realize this step, the fan-out cone of each net is first identified and then a vector is constructed to cover the minimal set. Finally, the net with largest cover is selected as the target net to insert the test point. After the test point insertion, the new transition probability of the logic cone is re-evaluated and the iteration repeats from the beginning of the algorithm until the transition probability of all the nets are improved above $TP_{th.}$ Note that there may be the case that the selected input net of one candidate net is primary input. Then the input pattern of the net can be altered to meet the requirement of transition probability.

		G10	G13	G15	G11	G17	G14	G12	G16	G8	G9
Loop1	G12	1	1	1	1	1	0	1	0	0	1
	G14	1	0	1	1	0	1	0	1	1	1

		G10	G13	G15	G11	G17	G14	G12	G16	G8	G9
Loop2	G14	1	0	1	1	0	1	0	1	1	1

Fig.6 The cover vectors representing whether one net is located in the logical fan-out cone of one logical gate

In order to better explain the insertion algorithm, the example in Fig. 2 is used here. When the threshold of transition probability TP_{th} is set to 0.17, we can get the net set *MINTP*={G10,G13,G15,G11,G17} in the first loop. Then the input net of each net in *MINTP* is identified and reordered in the ascending order of logical depth from the primary inputs. Here we get *I_MINTP*={G12,G14}. Since G12 and G14 have same logic depth, the one with maximum coverage of *MINTP* needs to be identified. Hence, for each net n_i in *I_MINTP*, one cover vector representing whether one net of *MINTP* is located in the logical fan-out cone of n_i is generated as shown in Fig.6. From the figure, we can see that the cover vector of G12 can cover the entire set *MINTP*, so only one 2-to-1 MUX is required to insert in G12. After insertion, the circuit is updated and the procedure goes back to line 1 and start the next loop. The algorithm continues until there is no net with the transition probability smaller than 0.17.

In a conclusion, the proposed method can insert *n* MUXs in at most *n* loops with simple sorting in each loop, because there is no rejection of insertion process like in [1], which in turn guarantees the short runtime of the proposed method. The algorithm complexity only comes from the case when logic cone searching and vector covering need to be performed, which is proportional to $O(N^2)$, where *N* is the total number of nets in the circuits. More importantly, the proposed method always tries to select one optimal net as the target insertion point, which guarantees the significant reduction of required insertion points.

5.4 Determining threshold of transition probability TP_{th}

We have seen that the selection of TP_{th} determines how many insertion points are required to best detect the HTs. In this section, we discuss that how to set TP_{th} to guarantee the detection of HTs in the worst case. Here we assume that the hardware Trojan circuits can be inserted into IC during GDSII generation, mask generation, and fabrication process. For traditional Trojan circuit, it is obvious that each Trojan circuit consists of two parts: Trigger and Payload. Only when the output of the trigger circuit upset the data input through the pay load, the Trojan circuit is activated (refer to Fig. 7).

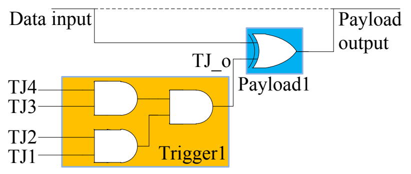


Fig.7 an example of 4-input Trojan circuit

After applying our proposed method, the minimal transition probability must be no smaller than TP_{th}. In order to consider the worst case, we assume the nets with minimal transition probability TP_{th} are connected with the inputs of the Trojan circuit. As for a net with the transition probability of TP_{th}, its signal probability can be computed by

$$sp \times (1 - sp) = TP_{th} \tag{3}$$

Solving the equation (3), the signal probability can be got as

$$sp = \frac{1 \pm (1-4TP_{th})^{\frac{1}{2}}}{2} \tag{4}$$

Assume the trigger circuit of one Trojan circuit is composed of all AND/OR gates, in order to get the low bound of the threshold of transition probability TP_{th}, the nets with minimal signal probability being "1" are connected to the inputs driving AND gates of the Trojan circuit, and the nets with minimal signal probability being "0" are connected to the inputs driving OR gates of the Trojan circuit. As for a Trojan circuit with *n* inputs, its output transition probability of the trigger circuit of the Trojan circuit can be written by

$$TP_{out} = \left(\frac{1-(1-4TP_{th})^{\frac{1}{2}}}{2}\right)^n \times \left(1 - \left(\frac{1-(1-4TP_{th})^{\frac{1}{2}}}{2}\right)^n\right) \tag{5}$$

In [16], the transition probability is modeled using geometric distribution (GD) and is used to estimate number of clock cycles required to generate a transition on a net. Based on (5), the number of required clock cycles No_{clk} to generate a transition at the output of the trigger circuit of the Trojan circuit with *n* inputs can be computed by

$$No_{clk} = \frac{1}{\left(\frac{1-(1-4TP_{th})^{\frac{1}{2}}}{2}\right)^n - \left(\frac{1-(1-4TP_{th})^{\frac{1}{2}}}{2}\right)^{2n}} - 1 \tag{6}$$

According to (6), we can get the maximal number of required clock cycles to generate a transition at the output of the trigger circuit of the Trojan circuit with *n* inputs. Similarly, TP_{th} can be represented as a function of No_{CLK}.

$$TP_{th} = \left(\frac{1 \pm \left(1-\frac{4}{1+No_{CLK}}\right)^{\frac{1}{2}}}{2}\right)^{\frac{1}{n}} - \left(\frac{1 \pm \left(1-\frac{4}{1+No_{CLK}}\right)^{\frac{1}{2}}}{2}\right)^{\frac{2}{n}} \tag{7}$$

Next, we will take a Trojan circuit shown in Fig.7 as an example to describe the relationship between TP_{th} and *No_clk*. As for the Trojan circuit, only when all four inputs are set to "1"s, the logical value on the data input can be upsetted by the Payload circuit. In order to achieve the stealthy nature of HTs, we only consider the worst case that all inputs are connected to the nets with minimal signal probability being "1". When TP_{th}=0.1, the minimal signal probability being "1" sp_1^{min} after applying our proposed method can be computed by (4), we can get sp_1^{min} =1.1270E-1. As for the 4-input Trojan circuit, the signal probability being "1" of the output *TJ_o* can be computed by $sp_1^{min}(TJ\_o) = (sp_1^{min})^4$=1.6133E-4. According to (3), the transition probability of *TJ_o* is 1.6131E-4. Therefore, the number of required clock cycles No_{clk} to generate a transition at *TJ_o* is 6198.4 according to (6). That is to say, the Trojan circuit may be activated after applying at least 6199 random test patterns. Table 4 shows No_{clk} under the condition of the different number of inputs of trigger circuit and the different threshold of the transition probability TP_{th}. As can be seen from this table, when TP_{th} is fixed, No_{clk} increases in proportion to $1/TP_{th}$ with the increment of the number of inputs of trigger circuit. When *n* is fixed, No_{clk} increases in proportion to 10^n with the increment of TP_{th}.

Table 4 No_{clk} under the different number of inputs of trigger circuit and TP_{th}

No_{clk}	TP_{th}=1.0E-1	TP_{th}=1.0E-2	TP_{th}=1.0E-3
n=2	79	9799	997999
n=3	699	969999	996999999
n=4	6199	96019999	996001999999
n=5	54999	9504999999	995005000000119

6. EXPERIMENTAL RESULTS

In order to validate the efficiency of the proposed method, we implemented the proposed algorithm described in Section 5 in C language and performed experiments using the combinational benchmarks in ISCAS'89 benchmark suite. We carried out the experiments on a computer with Intel(R) Core(TM) i7@2.93 GHz processor and 8GB memory. We used a STM 65nm standard cell library to map the circuit net-list during circuit synthesis. The signal probability being "1" of each primary input is set to 0.5. In order to evaluate the efficiency of our proposed method, several experiments are implemented.

6.1 Improvement of transition probability

In the first experiment, we take the large benchmark circuit s38417 as an example, and compare the transition probability after applying our proposed method with those of the original circuit as shown in Fig. 8 and Fig.9. We can see from Fig. 8 that after insertion, the transition probability of all the nets are above threshold. From Fig.9, it shows that the transition probability depends on the logical depth from the primary inputs, *i.e.,* with the increment of the logical depth from the primary inputs, the transition probability reduces exponentially. After applying the proposed method, the transition probability can be increased up to the threshold of the transition probability. Although our proposed method may cause the reduction of the transition probability of some nets, the number of nets with the reduced transition probability is very small. What's more, the transition probability of every net after applying our proposed method is larger than the threshold of the transition probability TP_{th}, which in turn validates the efficiency of the proposed method. We can also see that with the reduction of the transition probability TP_{th}, the nets which are affected by the inserted test points have larger logical depth from the primary inputs. It is because that the net with less logic depth typically has larger transition probability. Hence, for smaller threshold, only the net with large logic depth will be selected for insertion to increase its transition probability.

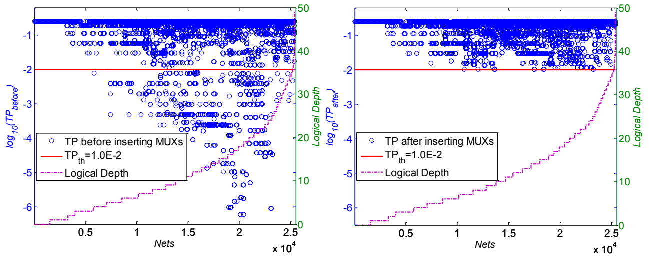
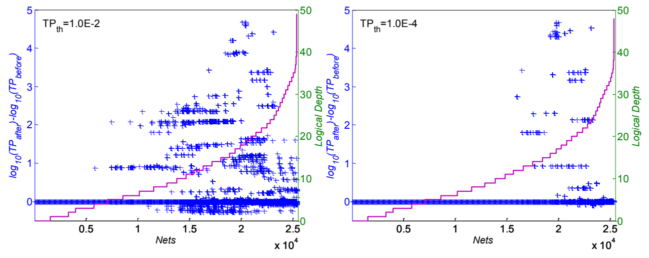


Fig.8 The distribution of transition probability



a) TP_{th}=1.0E-2 b) TP_{th}=1.0E-4

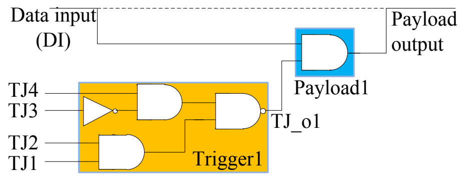
Fig.9 Comparison result of the transition probability after applying our proposed method with those of the original circuit of S38417

6.2 Activation of Trojan circuits

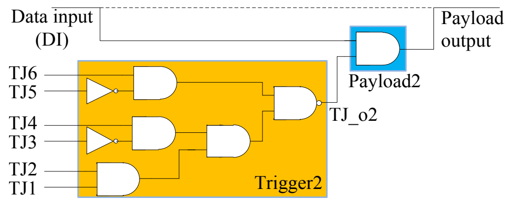
In the second experiment, two Trojan circuits similar to [1] shown in Fig.10 are inserted into the large benchmark circuit s5378. As for the Trojan circuit 1, only when the inputs of the trigger circuit are 1101 and the data input is 1, the data input will be upsetted by the payload gate. Similarly, only when the inputs of the trigger circuit are 110101 and the data input is 1 for the Trojan circuit 2, the data input can be upsetted by the payload gate. It is obvious that we can get remarkably different activation probability if the inputs of the Trojan circuit are connected to the nets with different transition probability.

In order to validate the improvement of our proposed method, three groups of experiments with different input connection combinations are considered. The selected nets connected with the inputs of Trojan circuit and their transition probability before and after inserting MUXs are listed in Table 5. The parameters TP_{before} and TP_{after} represent the transition probability of the selected nets before and after inserting MUXs, respectively. Simultaneously, in order to make the inserted Trojan circuits have the stealthy nature, the nets with smallest signal probability being "1" are selected to connect to the inputs of the Trojan circuit (TJ1, TJ2, TJ4 and TJ6), and the nets with smallest signal probability being "0" are selected to connect to the inputs of the Trojan circuit (TJ3 and TJ5).

In order to directly exhibit the efficiency of the proposed method, in case 1, we choose the nets with smallest transition probability before insertion to connect to the inputs of the Trojan circuits. Since the transition probability of these nets will be significantly improved, we can see the great improvement of the HT activation rate. However, in practical scenario, the adversary usually insert Trojan circuits in the final chip with MUX insertion already. Hence, the attacker will select the nets with smallest transition probability after MUX insertion. Therefore, we use case 2 and 3 to validate the efficiency of our method under this practical scenario. As for case 2(3), TP_{th} is set to be 1.0E-1(1.0E-2). Hence, all the nets in the chip will have transition probability larger than 1.0E-1(1.0E-2). Among all the nets, the nets smallest transition probability are selected to connect to the inputs of the Trojan circuits. The experimental results after inserting the Trojan circuits are shown in Table 6. In these experiments, 10000 random test patterns are applied and the number of the transition of each net is recorded.



a) Trojan 1



b) Trojan 2

Fig.10 Two Trojan circuits

For Case 1, in the original circuits, the inputs for TJ1 to TJ6 are fixed at 001010. Hence, output TJ_o does not have any transition. That is to say, the Trojan circuit must not be activated after the 10000 applied random test patterns. When applying our proposed method, the number of the transitions of the inputs of Trojan circuit are significantly increased, which may increase the probability of activation of the Trojan circuit. For example, when TP_{th}=1.0E-1, the Trojan circuit 2 may be activated for 56 times because the number of "0" at the output TJ_o of the trigger circuit is 56. Similarly, when TP_{th}=1.0E-2 and 1.0E-3, the Trojan circuit 2 may be activated for 39 and 82 times, respectively. With the reduction of TP_{th}, some inputs of the candidate nets with smallest transition probability before MUXs insertion have higher possibility of being directly selected as the insertion points, which in turn must result in the large increment of transition probability of the candidate nets. This is why the results for TP_{th} = 1.0E-3 are better than those of TP_{th} = 1.0E-1 and 1.0E-2.

For case 2, after MUX insertion, all the nets have transition probability larger than 1.0E-1. The attacker will still find the net

Table 5 The experimental results of the number of the transitions in s5378

Trojan inputs	Case 1(TP_{th}=1.0E-1, 1.0E-2, 1.0E-3)					Case 2 (TP_{th}=1.0E-1)			Case 3 (TP_{th}=1.0E-2)		
	Net	TP_{before}	TP_{th}=1.0E-1	TP_{th}=1.0E-2	TP_{th}=1.0E-3	Net	TP_{before}	TP_{after}	Net	TP_{before}	TP_{after}
			TP_{after}	TP_{after}	TP_{after}						
DI	I4774	0.2088	0.1132	0.1754	0.2088	I4774	0.2088	0.1118	I4774	0.2088	0.1881
TJ1	n2184gat	0.0000	0.1881	0.1903	0.1903	n971gat	0.0016	0.1057	n870gat	0.0014	0.0112
TJ2	n2328gat	0.0000	0.1877	0.1864	0.1864	n2148gat	0.0022	0.1035	n969gat	0.0018	0.0113
TJ3	n1885gat	0.0000	0.1880	0.1883	0.1883	n1648gat	0.0309	0.1061	n188gat	0.0142	0.0142
TJ4	I3923	0.0000	0.1313	0.1082	0.1874	n196gat	0.0020	0.1871	n1647gat	0.0000	0.0118
TJ5	n1384gat	0.0000	0.1881	0.1903	0.1903	n1307gat	0.0270	0.1068	I4780	0.0148	0.0148
TJ6	I4566	0.0000	0.1057	0.1106	0.1106	n78gat	0.0030	0.2345	n446gat	0.0014	0.0116

Table 6 The experimental results after inserting the Trojan circuits

	Output of Trigger	Original		TP_{th}=1.0E-1		TP_{th}=1.0E-2		TP_{th}=1.0E-3	
		No. zeros	No. transitions	No. zeros	No. transitions	No. zeros	No. transitions	No. zeros	No. transitions
Case 1	TJ_o1	0	0	100	199	81	161	163	325
	TJ_o2	0	0	56	113	39	78	82	165
Case 2	TJ_o1	0	0	11	23	-	-	-	-
	TJ_o2	0	0	4	9	-	-	-	-
Case 3	TJ_o1	0	0	-	-	0	0	-	-
	TJ_o2	0	0	-	-	0	0	-	-

with the minimum transition probability to insert the Trojan. We list such nets in the Table 5 and show the transition probability change before and after MUX insertion. We can see that for this case, the Trojan cannot be activated in the original circuit. With the transition probability increased larger than 1.0E-1, Trojan circuit 1 and Trojan circuit 2 may be activated for 14 and 2 times, respectively. Note that according to equation (6), in the worst case for transition probability 1.0E-1, we need spend at most 487998 random test patterns to generate a transition at the output of the trigger circuit 2. In fact, after applying 10000 random test patterns, the Trojan circuit 2 may be activated for 2 times. It shows that the normal case can be much better than the worst case estimation and it is also due to the larger transition probability than the threshold 1.0E-1.

As for case 3, the threshold is set to 1.0E-2 and the transition probability of all the nets are increased to above 1.0E-2. According to equation (6), in the worst case, we need spend at most 9.41E+11 random test patterns to generate a transition at the output of the trigger circuit 2. In fact, the Trojan circuit 2 can't be activated after applying 10000 random test patterns because the number of the applied test patterns is far smaller than that of required test patterns. However, although Trojan circuits can't be directly activated for case 3, the insertion of 2-to-1 MUXs is helpful to increases the switching activity of the hardware Trojan circuit, which will be beneficial to the Trojan detection techniques based on power analysis as shown in the next section.

In a summary, based on the above three cases, we effectively testify that the proposed method can guarantee the transition probability to be enlarged up to the threshold of the transition probability TP_{th}, which implies that the probability of activation of the Trojan circuit could be improved. Case 1 shows a relatively good case that when the Trojan is connect to the points with larger transition probability after MUX insertion, the activation rate is pretty high. Case 2 and 3 show that even in the case the adversary chooses the nets with minimum transition probability to connect the Trojan, it can still possibly be activated or our method will ease the Trojan detection in another way as discussed next.

6.3 Power Analysis

As can be seen from the previous sections, the insertion of 2-to-1 MUXs is helpful to activate the hardware Trojan circuit, which in turn increases the power of the hardware Trojan circuit. That is to say, the techniques based on power analysis may benefit from our proposed method. In order to analyze the effectiveness of insertion of 2-to-1 MUXs in the techniques based on power analysis, s5378 benchmark is taken as an example. Two designs are generated: 1) design without the insertion of 2-to-1 MUXs and 2) design with the insertion of 2-to-1 MUXs considering TP_{th}=1.0E-2. The connection relationship (Case 3) shown in Table 5 is utilized to get the designs with the hardware Trojan circuit. To analyze contribution of Trojan circuit on design power consumption, the ratio, *Ratio_PW*, is defined as the power of the design without Trojan circuit to the power of design with Trojan circuit:

$$Ratio\_PW = \frac{Power_{without}}{Power_{with}} \quad (8)$$

Where $Power_{without}$ and $Power_{with}$ denote the power of the design without and with hardware Trojan circuit, respectively. The power is estimated by the number of switching activities. Fig.11 shows the comparison results of *Ratio_PW* for Trojan circuits 1 and 2. As can be seen Fig.11, the Trojan impact on the power is significantly magnified after MUXs insertion in some cases when compared with the design without the insertion of MUXs, which in turn testify that our proposed method is beneficial to the techniques based on power analysis.

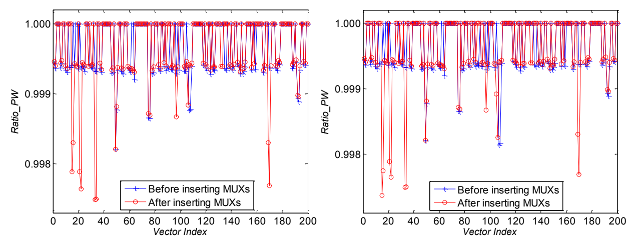


Fig.11 Ratio of power before MUX insertion and after MUX insertion when TP_{th}=1.0E-2

6.4 Area, delay, and power overhead

Finally, we apply our proposed method to some large ISCAS89 benchmark circuits, and extract the overhead including power, delay and area of the proposed method as shown in Table 7. Herein, the power overhead is estimated by the number of switching activities, the delay overhead is computed by the delay of the most critical path, and the hardware overhead is computed by the total area of the logical gates. Unlike the method in [1], where a dSFF including a flip-flop circuit and a 2-input AND/OR is required to insert on a selected net, here our proposed method requires to insert a 2-to-1 MUX and flip-flop on a selected net. In fact, when the number of the insertion points is small, the flip-flops can be removed and only one 2-to-1 MUX is inserted for a selected net. In this way, two primary inputs are required for B and S ports of one inserted MUX. Therefore, we give the area overheads of only inserted MUXs and both MUXs and flip-flops. As can be seen from this table, the power, delay and area increase with the increment of the threshold of transition probability, which implies that the smaller threshold of transition probability TP_{th} might be utilized for the high performance circuits. Note that although the power overhead for large TP_{th} can be around 30%, it is only consumed in the testing mode. When circuit is in real function, the MUX will not switch and there is only small leakage power consumed. The total area overheads for large TP_{th} are about 40% for smaller benchmark circuit and 30% for larger benchmark circuit, respectively.

It is well-known that a common 2-to-1 MUX usually consists of two 2-input ANDs, one 2-input OR and NOT, so the area of one 2-to-1 MUX must be larger than that of 2-input AND/OR. However, if the 2-to-1 MUX is implemented by using transmission gate logic, the area overhead can be reduced. At the same time, our proposed selection algorithm can also be utilized to optimize the number of the insertion points. Therefore, in order to fairly compare the area overhead, only the ratio of the number of insertion points to the total number of the nets with the transition probability smaller than TP_{th} is shown Table 8. In [1], the ratio of the number of insertion dSFFs to the total number of the nets with the transition probability smaller than TP_{th} is in the

Table 7 The experimental results after applying our proposed method

Circuit		TP_{th}	#Net $TP<TP_{th}$	#MUXs	#MUXs/ #Net $TP<TP_{th}$	Runtime (s)	Increment of Overhead(%)			
Name	#Gate						Power	Delay	Area1 (MUXs)	Area2 (MUXs+FFs)
S5378	2836	1.0E-1	725	207	0.29	28.09	26.08	8.37	15.59	44.81
		1.0E-2	158	38	0.24	3.78	6.17	0.00	2.86	8.23
		1.0E-3	48	16	0.33	1.62	1.61	0.00	1.20	3.46
		1.0E-4	24	7	0.29	0.68	1.45	0.00	0.53	1.52
S9234.1	5597	1.0E-1	1216	258	0.21	209.49	31.94	14.22	9.63	27.68
		1.0E-2	551	35	0.06	17.73	13.61	0.00	1.31	3.76
		1.0E-3	282	20	0.07	10.11	10.50	0.00	0.75	2.15
		1.0E-4	88	7	0.08	3.40	4.14	0.00	0.26	0.75
S13207.1	7979	1.0E-1	1436	409	0.28	469.48	30.97	5.59	10.91	31.37
		1.0E-2	959	129	0.13	94.18	10.86	2.05	3.44	9.89
		1.0E-3	765	59	0.08	39.25	4.93	0.00	1.57	4.53
		1.0E-4	404	42	0.10	26.35	3.90	0.00	1.12	3.22
S15850.1	9775	1.0E-1	1750	420	0.24	1444.06	23.26	13.48	9.15	26.30
		1.0E-2	764	125	0.16	234.81	11.68	6.28	2.72	7.83
		1.0E-3	377	98	0.26	176.57	9.62	5.32	2.13	6.14
		1.0E-4	74	14	0.19	22.46	0.36	0.00	0.30	0.88
S38417	22257	1.0E-1	3179	944	0.30	8007.30	24.54	3.40	9.21	26.47
		1.0E-2	1286	188	0.15	1184.03	5.89	2.11	1.83	5.27
		1.0E-3	785	73	0.09	441.50	3.05	0.00	0.71	2.05
		1.0E-4	240	40	0.17	232.74	1.68	0.00	0.39	1.12
S38584.1	19405	1.0E-1	3669	1342	0.37	15354.27	27.54	9.81	12.57	36.14
		1.0E-2	1710	190	0.11	962.33	8.72	2.13	1.78	5.12
		1.0E-3	1080	40	0.04	194.60	4.35	0.00	0.37	1.08
		1.0E-4	665	24	0.04	115.49	3.56	0.00	0.22	0.65

range of 36%-100% when TP_{th} is in the range of 1.0E-1~1.0E-4 for s38417. Comparing with that of [1], our proposed method requires small ratio of the number of insertion points of the total number of the nets with the transition probability smaller than TP_{th}, and there is a 6%-83% gap. Even consider the possible 1.5-2 times larger area of MUX than dSFF point, our proposed method may still result in averagely smaller hardware and delay penalty.

Table 8 Comparison of the ratio of #insertion points to #the nets with transition probability smaller than TP_{th}

TP_{th}	1.0E-1	1.0E-2	1.0E-3	1.0E-4
Method [1]	36	46	57	100
Proposed	30	15	9	17

7. CONCLUSION

In this paper, we proposed a low hardware overhead acceleration method for the detection of HTs based on the insertion of 2-to-1 MUXs. In the proposed selection algorithm, the fact that one logical gate has significant influence on the transition probability of the logical gates in its logical fan-out cone is utilized to reduce the number of the insertion of 2-to-1 MUXs. Simultaneously, as for one candidate net, its input net with the minimal signal probability is selected as the inserted net to improve itself transition probability. Because the proposed selection algorithm gets the optimal insertion points by analyzing the logical connection relationship, it is compatible with any other methods based on the insertion of specific circuits to improve the controllability and/or observability. In the proposed test architecture, a uniform insertion circuit like 2-to-1 MUX is employed to simplify the design overhead. Through our proposed method, the transition probability of the entire circuit can be improved with small delay and area overhead.

8. ACKNOWLEDGMENTS

This work was supported by A*Star - SERC Public Sector Funding (PSF) of Singapore under Grant No. 1121202015. The authors would like to acknowledge the National Natural Science Foundation of China for the support received from the team working under its Grant No. 61100031

9. REFERENCES

[1] H. Salmani, M. Tehranipoor, J. Plusquellic. A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2012, 20(1):112 - 125

[2] M. Tehranipoor, F. Koushanfar. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design & Test of Computers*. 2010, 27(1): 10 - 25

[3] R. S. Chakraborty, S. Bhunia. Security against hardware Trojan through a novel application of design obfuscation. In *Proceedings of International Conference on Computer-Aided Design(ICCAD)*. 2009:113–116

[4] B. Cha, S. K. Gupta. Trojan detection via delay measurements: A new approach to select paths and vectors to maximize effectiveness and minimize cost. In *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2013:1265 - 1270

[5] Y. Jin, Y. Makris. Hardware Trojan Detection Using Path Delay Fingerprint. In *Proceedings of IEEE International Workshop on Hardware-Oriented Security and Trust(HOST)*. 2008:51-57

[6] K. Xiao, X. Zhang, M. Tehranipoor. A Clock Sweeping Technique for Detecting Hardware Trojans Impacting Circuits Delay. *IEEE Design & Test*. 2013, 30(2): 26 - 34

[7] H. Salmani, M. Tehranipoor. Layout-Aware Switching Activity Localization to Enhance Hardware Trojan Detection. *IEEE Transactions on Information Forensics and Security*. 2012, 7(1): 76-87

[8] R. Rad, J. Plusquellic, M. Tehranipoor. Sensitivity analysis to hardware Trojans using power supply transient signals. In *Proceedings of IEEE International Workshop on Hardware-Oriented Security and Trust(HOST)*. 2008:3-7

[9] M. Banga, M. S. Hsiao. A Novel Sustained Vector Technique for the Detection of Hardware Trojans. In *Proceedings of 22nd International Conference on VLSI Design*. 2009:327-332

[10] F. Wolff, C. Papachristou, S. Bhunia, et al. Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme. In *Proceedings of Design, Automation and Test in Europe(DATE)*. 2008:1362-1365

[11] R. S. Chakraborty, F. Wolff, S. Paul, et al. MERO : A Statistical Approach for Hardware Trojan Detection. In *Proceedings of 11th International Workshop Cryptographic Hardware and Embedded Systems*. 2009:396-410

[12] R. S. Chakraborty, S. Paul, S. Bhunia. On-demand transparency for improving hardware Trojan detectability. In *Proceedings of IEEE International Workshop on Hardware-Oriented Security and Trust(HOST)*. 2008:48-50

[13] M. Potkonjak, A. Nahapetian, M. Nelson, et al. Hardware Trojan horse detection using gate-level characterization. In *Proceedings of ACM/IEEE Design Automation Conference(DAC)*. 2009:688 - 693

[14] S. Wei, M. Potkonjak. Self-Consistency and Consistency-Based Detection and Diagnosis of Malicious Circuitry *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2013, PP(99): 1-9

[15] Y. Alkabani, F. Koushanfar. Consistency-based Characterization for IC Trojan detection. In *Proceedings of International Conference on Computer-Aided Design(ICCAD)*. 2009:123-127

[16] D. D. Wackerly, I. W. Mendenhall, R. L. Scheaffer. Mathematical Statistics With Application. 7th Edition, Thomson Learning, 2008