Hardware Trojan Exploting the FiF-FoF Values of Machine States

***Abstract*—In this paper we use the metric FiF-FoF (fan-in-factor and fan-out-factor) to characterise an FSM state with respect to its vulnerability to host a Hardware Trojan (HT). The FiF-FoF defines the mobility among the FSM states by estimating how frequently a state can be reached from other states as well as the frequency of exit from the state. The uniform mobility among the states points to less stealthy place for HT and, therefore, less vulnerable. The explored HT expoits the state codes that are hard-to-reach or hard-to-exit. The HTs can be hidden behind such states to evade the detection schemes. The analysis/experimentation establishes that such HTs are rarely activated and hard to detect.**

***Index Terms*—Hardware Trojan, FiF-FoF, Finite State Machine, Trigger Signal**

I. Introduction

Integrated Chips (ICs) have been used in social information infrastructure and systems that contain confidential information. It is, therefore, necessary to ensure high secrecy for such ICs. However, the designer cannot completely control the whole process of designing and manufacturing of an IC. Globalization in the IC industry decreases control of a designer on the fabricated chips and, therefore subjected to malicious attack such as hardware trojan (HT).

Tampering through malicious modification of a design, can be introduced at different steps of the IC manufacturing flow -that is, malicious insertion in an IP, modification of netlist by a CAD tool during design, or tampering a GDSII file during fabrication. This can have serious consequences during in-field operation [1]. HTs disable the normal function of ICs, leak secret information, or destroy the ICs [2] [3].

In general, a trojan circuit consists of trigger unit and payload. The trigger unit activates trigger signal once the activation condition (trigger condition) set by the malicious attacker is satisfied. The payload unit executes the attacker desired function.

Several methods have been reported for detection trojans [3] [4]. These are divided into two categories. One verifies the behaviors of circuits (verification-based approach). When a trojan increasingly affects the behavior of the circuit, the detection of such trojan becomes easier. The other type of detection schemes analyze side-channel parameters (side-channel analysis-based approaches) such as leakage current, dynamic power, path-delay characteristics, or noise to sense presence of HT in the logic circuit/IC. Due to presence of the additional logic like HT, the amount of side-channel information such as the leakage current increases.

Many research works have been reported so far based on combinational HT. Bhunia.et.al describe sequential HT in [5] [6] and demonstrate possibility of different type of sequential HTs. One such HT is the FSM based trojan. It is also mentioned that the counter-based trojans [] can be generalized to FSM-based trojans that has a sequential and a combinational part. The edge of the FSM-based trojan is that they can be designed to be arbitrarily complicated with the same amount of resource and can reuse both the combinational logic and flip-flops (FFs) of the original circuit for FSM-hosting. A counter is uni-directional, but an FSM-based trojan can have state transitions leading back to the initial state. It dictates that the final trojan state to be reached only if the entire state sequence is satisfied in consecutive clock cycles [5].

Based on this scenario, this work proposes an HT model which is a sequential hardware trojan where the trojan affected parts are the FSM. The metric FiF-FoF (fan-in-factor and fan-out-factor) is used to characterise an FSM state with respect to its vulnerability to host the HT. The FiF-FoF defines the mobility among the FSM states. It estimates how frequently a state can be reached from other states as well as the frequency of exit from the state. The states that are hard-to-reach or hard-to-exit are exploited for HT. The HTs can be hidden behind such states to evade the detection schemes. The analysis as well as experimentation establish the fact that such HTs are rarely activated. The probability of triggering condition is also computed and it can be observed that the explored trojan is extremely rare and also random and, therefore, hard-to-detect with the present conventional detection schemes.

II. Hardware Trojan

The hardware trojans (HTs) are malicious changes or alterations done by some adversaries. The purpose of implanting HT in a logic circuit is to harm a secure system by leaking its secret data or destroying the system. To evade detection method, the HT is triggered rarely and the triggering condition is random in nature. The probability of rare and random properties of HT makes it harder to detect. The HTs can be of two types: combinational and Sequential HT [4].

A. Combinational and sequential hardware trojan

The activation of a combinational HT depends on the occurrence of a particular condition at certain internal nodes of the circuit. On the other hand, activation of sequential HT depends on the occurrence of a specific sequence of rare logic values at internal nodes [5].

The traditional verification and testing scheme detects an HT by triggering it. Therefore, malicious attackers consider rare HT trigger condition to ensure that the HT is silent during the verification.

The *trigger condition* is an event, manifested in the form of a particular boolean value of certain internal/external wires of the login circuit. It activates the HT trigger logic that

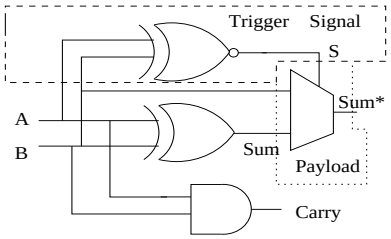

Fig. 1: Hardware Trojan in half-adder circuit [7]

leads to activation of *trigger signal*. The trigger signal invokes the payload circuitry to execute the function desired by the malicious attacker.

A trojan infused half adder circuit is shown in Fig. 1 [7]. Here, A = B is the *trigger condition*. The select line S of the multiplexer is the *trigger signal*. Once the trigger signal S is activated, the *payload* is invoked -that is, Sum = B, desired by the attacker, is executed instead of the original function Sum = A $\oplus$ B.

B. Trojan detection with HaTCh

There are many reported trojan detection approaches. Recently in [7], an efficient scheme namely HaTCh algorithm is proposed to detect HT. The four properties *trigger signal dimension* $d(T)$, *payload propagation delay* $t(T)$, *implicit behavior factor* $\alpha(T)$ and *trigger signal locality* $l(T)$ are defined in [7] to measure the stealthiness of an HT [8].

The $d(T)$ represents the number of wires used by the HT trigger circuitry. A large d complicates activation of trigger signal -that is, the HT is harder to detect.

The $t(T)$ is the number of cycles required to propagate a malicious behavior to the output port of a chip after an HT is triggered. A large t means it takes a long time, after triggering, to activate the malicious behavior. That is, it is less likely the HT can be detected during testing.

The $\alpha(T)$ represents the probability that an HT gets triggered. That is, the HT will not (explicitly) manifest malicious behaviorr. Such an HT shows implicit behavior. Higher probability of implicit malicious behavior means higher stealthiness during testing phase.

The $l(T)$ denotes the spread of trigger signal wires of the HT across the IP core. Small l implies that the wires are in the close vicinity to each other. For large l, it is harder to figure out exactly which wires form the trigger signals and, therefore, the HT is harder to detect.

The main purpose of introducing HaTCh is to map the trojan model with the parameters d, t, α, l and to prove that the trojan is hard-to-detect. The proposed HT model is represented by multiple sets of trigger states T, each having their own d, t, α and l, to slect the achievable region of the HT.

III. FSM

The general structure of a synchronous sequentte state machine (FSM) is shown in Fig. 2. Its combinational network is the CC. The system register (SR) consists of k memory elements. The outputs y_1, y_2, $\cdots$ y_k of the memory elements define the present state (PS) of the machine.

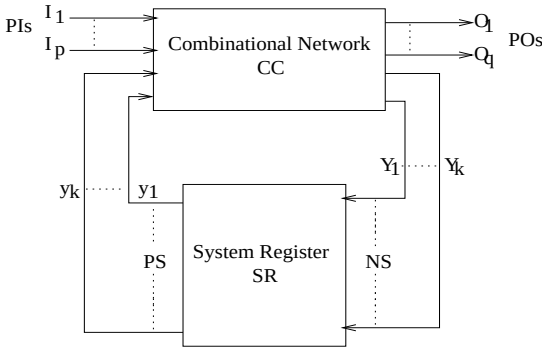

Fig. 2: FSM

In the state transition graph (STG) of an FSM if there are n states, then at least k memory elements/flip-flops are required to encode its states, where $2^{k-1}<n\leq 2^k$. Out of 2^k state codes, only n are assigned to the states of the FSM. The remaining N = 2^k-n codes are unreachable states of the FSM. Such unreachable states never appear on the PS lines during normal functioning of the FSM.

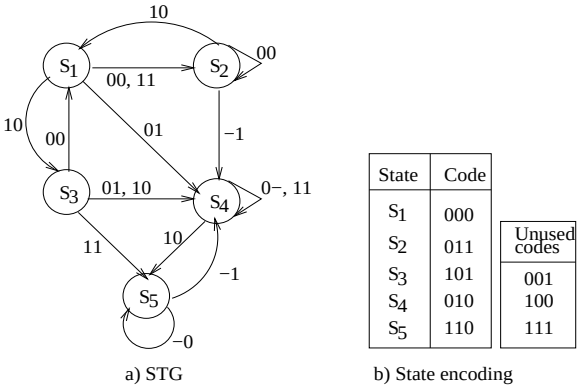

Fig. 3: State transition diagram of an FSM and state encoding

Example 1: Fig. 3b shows a random state assignment for the sequential machine shown in Fig. 3a. The codes 001, 100 and 111 are not assigned to any state. Therefore, these three are the unreachable states of the FSM.

A. Hard-to-reach states

An FSM can have a large number of states with a very few transitions falling on them. These are referred to as the hard-to-reach states. During normal functioning of the FSM a hard-to-reach state (S_3 of Fig. 3a) can be scarcely arrived. Therefore, the code (101) of such a state (S_3) rarely appears on the PS lines.

B. Hard-to-emit states

There are states of an FSM with a large number of self loops. These are referred to as hard-to-emit states. Such a state (S_4 of Fig. 3a) acts like a sink. That is, once the FSM reaches the state (S_4), it tends to remain there for a large number of input combinations.

C. Easy-to-reach states

In an FSM, there can have states with a large number of transitions falling on them. These are referred to as the easy-to-reach states. During normal functioning of the FSM an easy-to-reach state (S_4 of Fig. 3a) can be arrived from a number of states for a number of primary input combinations. That is, the code (010) of such a state (S_4) appears frequently on the PS lines.

D. Easy-to-emit states

There are states in an FSM with a very few number of self loops. These are the easy-to-emit states. Such a state (S_3 of Fig. 3a) acts like a trasient node. That is, once the FSM reaches the state (S_3), it immediately exits from that state for an input combination. Therefore, the code (101), of such a state (S_3), on the PS lines is transient.

The hard-to-reach and easy-to-emit states are of our primary concern in the cotext of trojan circuit insertion in a logic design. The next section explores the vulnerable points in a sequential machine/design that can be target of malicious attack.

IV. Overview of the HT

The target of an attacker can be to identify the vulnerable locations for the HT in a sequential machine so that the HT is hard to detect. This can be achieve as the following are true for an FSM.

a. There are hard-to-reach states.

b. There are easy-to-emit states.

This section focuses on the hard-to-reach and easy-to-exit states. Both these properties block the state codes (101) of the states (S_3 of Fig. 3a) to appear on the PS lines. To decide on the location/place for HT, the following terms defined in [9] are considered as the metric.

A. FiF-FoF

The concept of FiF-FoF is briefed from [9]. The idea behind the FiF-FoF analysis of an FSM is to quantify the merit of its states from the point of view of defining location for HT. Particularly, the FiF value of an FSM state denotes the relative ease with which it can be entered into, while the FoF value expresses the ease of leaving that state.

Definition 1: For each state S of an FSM, reachabilit(S) is the number of edges incident on S from other states. The self loops are not counted in reachability(S) analysis.

Definition 2: For each state S of an FSM, emitability(S) is the number of edges exit from S. The self loops are not counted as they fall on themselves.

Definition 3: For each state S, fan-in-factor of S, denoted as FiF(S), is the ratio of reachability(S) and emitability. Therefore FiF(S) = reachability(S)/emitability(S).

Definition 4: For each state S, fan-out-factor of S, denoted as FoF(S), is thc ratio of emitability(S) and reachability(S). Therefore, FoF(S) = emitability(S)/reachability(S).

It may he noted that for each state S of an FSM FiF(S) $\times$ FoF(S) = 1.

TABLE I: Computation of FiF-FoF

State	Reachabiliy	Emitability	FiF	FoF
S1	2	4	0.5	2.0
S2	2	3	0.67	1.5
S3	1	4	0.25	4.0
S4	4	1	4.0	0.25
S5	2	2	1.0	1.0

Example 2: In Fig. 3a, the number of input edges falling on the states S_3 and S_4 from the other states are respectively -that is, the reachabilify(S_3) = 1, whereas the reachability(S_4) = 4. Out of 2^2=4 input combinations, the number of times the FSM exits/emits from state S_3 is 4, which defines the emitability(S_3) = 4. Similarly, the emitability(S_4) = 1. Therefore, the FiF(S_3) = reachability(S_3)/emitability(S_3) = $\frac{1}{4}$ = 0.25 and FiF(S_4) = 4. The FiF anf FoF values of all the five states of the FSM shown in Fig. 3 are given in Table I.

The FiF-FoF analysis of an FSM requires computation of these values for cach state of the FSM. The computation of FiF-FoF is reported in [9]. Since our aim is to identify states for HT insertion, there is no need to be too precise about the FiF-FoF values. Therefore, the approximate computation of FiF-FoF values of an FSM states as reported in [9] is sufficient for the current purpose.

Instead of scanning the entire state transition table (STT), we can apply a sequence of patterns/test patterns and trace the behavior as it jumps from one state to another. At each step, the reachability and emitability values of the relevant states are updated. A number of such test pattern sequences are applied in succession and the average reachability and emitability values are computed. Finally, the FiF-FoF values are computed from the average reachability and emitability. The algorithmic steps of the heuristic are as reported in [9].

V. FiF-FoF for Hardware Trojan Implantation

This section describes how the HT implantation scheme exploits FiF-FoF values of state codes. A subset of the signal lines in PS is considered as the trigger condition.

Let the state S_{lowfif} (S_3 in Table I) is with lowest FiF value. Then the state code $b_{k-1}b_{k-2}\cdots b_1b_0$ (101 in Table I) can be used as to set the trigger condition. A specific k-bit pattern generated from a random pattern generator of P-bit, where $k \leq$P, is compared with the $b_{k-1}b_{k-2}\cdots b_1b_0$ to set the trigger signal.

Now, the value of k is to be optimized to m:

Compare the state code $b_{k-1}b_{k-2}\cdots b_1b_0$ of S_{lowfif} with the state code $c_{k-1}c_{k-2}\cdots c_1c_0$ (010/110/011/000 in Table I) of each of the states $S_{highfif}$ (S_4/S_5/S_2/S_1) in Table I) with a high FiF value to find the PS lines -that is, $k-i$, $k-j$, ... that can contribute to trigger condition. From Table I, it can be observed that any one the bit pairs of 10/01/11 of the code for S_3 representing the PS_2PS_1/PS_1PS_0/ PS_2PS_0 can be considered as the rare pattern. That is, the m = 2. However, if we consider only the S_4 and S_5 as the state with high FiF values, then the code 101 of S_3 is to be compared with 010 (S_4) and 110 (S_5). It can be then observed that the m = 1 (either PS_1 or PS_0 can be considered as the rare pattern).

NEED TO SPECIFY ALGO

The solution to avoid the vulnerability:

Assign state codes in such a way that the state S_{lowfif} with low FiF value is having state code $b_{k-1}b_{k-2}\cdots b_1b_0$ and the state $S_{highfif}$ with high FiF value has the code $c_{k-1}c_{k-2}\cdots c_1c_0$, where the hamming distance between these two is minimum. In case of the FSM of Fig. 3, code for S_3 with low FiF value is 101. The code for S_4 with high FiF value is 010. The hamming distance between these two codes is 3 -that is, k (the number of PS lines). The better choice for S_3 can be 001/100/111.

If it is 001, then PS_1PS_0 is the rare condition.

If it is 100, then PS_2PS_1 is the rare condition.

If it 111, then PS_2PS_0 is the rare condition.

That is, for m = 2, we have one choice for selecting the trigger condition.

If we consider only the S_4 and S_5 as the state with high FiF values -that is, m = 1, then the code 001/100/111 for S_3 can have the following impact.

If it is 001, then PS_1 or PS_0 is the rare condition.

If it is 100, then PS_1 is the rare condition.

If it 111, then PS_0 is the rare condition.

VI. Experimentation

The proposed scheme has been carried out in the framework of SIS/abc cite. Extensive experimentation was done on a large number of MCNC/ISCAS benchmark circuits and some randomly generated large FSMs. Table II represents the relevant parameters of the MCNC benchmarks and the randomly generated circuits as shown in Column I. The number of primary inputs, primary outputs, states in the FSM, flip-flops requiredi, ... are noted in columns 2, 3 , 4, 5, ..., respectively.

The ISCAS benchmark circuit descriptions are given in Table III. *Column 5* of *Table III* reports the number of flip-flops (M) with events (in its PS lines) less than the threshold value (10% of the maximum limit).

VII. Conclusion

References

[1] S. Bhunia, M. S. Hsiao, M. Banga and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," in Proceedings of the IEEE, vol. 102, no. 8, pp. 1229-1247, Aug. 2014. doi: 10.1109/JPROC.2014.2334493

[2] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using ic fingerprinting," in IEEE Symposium on Security and Privacy 2007 (SP '07) , May 2007, pp. 296-310.

[3] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," in IEEE Design & Test of Computers, vol. 27, no. 1, pp. 10-25, Jan.-Feb. 2010. doi: 10.1109/MDT.2010.

[4] Rajat Subhra Chakraborty, Seetharam Narasimhan and Swarup Bhunia, "Hardware Trojan: Threats and Emerging Solutions," in IEEE International High Level Design Validation and Test Workshop, 2009.

[5] X. Wang, S. Narasimhan, A. Krishna, T. Mal-Sarkar and S. Bhunia, "Sequential hardware Trojan: Side-channel aware design and placement," 2011 IEEE 29th International Conference on Computer Design (ICCD), Amherst, MA, 2011, pp. 297-300. doi: 10.1109/ICCD.2011.6081413

[6] X. Wang, M. Tehranipoor and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions,"in Hardware-Oriented Security and Trust, 2008. Host 2008. IEEE International Workshop on . IEEE, 2008, pp. 15-19.

[7] S. K. Haider, C. Jin, M. Ahmad, D. M. Shila, O. Khan and M. van Dijk, "Advancing the State-of-the-Art in Hardware Trojans Detection," in IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 1, pp. 18-32, 1 Jan.-Feb. 2019. doi: 10.1109/TDSC.2017.2654352

[8] S. K. Haider, C. Jin, and M. van Dijk, Advancing the state-of-the-art in hardware trojans design, arXiv:1605.08413, 2016. [Online]. Available: http://arxiv.org/abs/1605.08413.

[9] S. Roy, U. Maulik, S. Bandyopadhyay, S. Basu and B.K. Sikdar, "Fan-in- and fan-out-factor oriented BlST design for sequential machines", IEE Proceedings, vol. 150, no. 3, pp 183-188, May 2003.

TABLE II: Circuit description MCNC

Circuit	#PIs	#POs	# States	#FFs	#UR states	#HR states	#EE states
ex1	9	19	20	5	12		
s27	4	1	06	3	2		
s208	11	2	18	5	14		
s386	7	7	13	4	3		
s420	19	2	18	5	14		
s820	18	19	25	5	7		
s832	18	19	25	5	7		
s1488	8	19	48	6	16		
s1494	8	19	48	6	16		
bbsse	7	7	16	4	0		
styr	9	10	30	5	2		
keyb	7	2	19	5	13		
opus	5	16	10	4	6		
sse	7	7	16	4	0		
kirkman	12	6	16	4	0		
ex6	5	8	8	3	0		
scf	27	56	121	7	7		
tbk	6	3	32	5	0		
$RAND_1$	56	58	624	10	400		
$RAND_2$	64	77	819	10	205		
$RAND_3$	82	73	1016	10	8		

TABLE III: ISCAS circuit descriptions

Circuit	#PIs	#POs	# States	#FFs	#UR states	#HR states	#EE states
s344	9	11		15			
s349	9	11		15			
s1196	14	14		18			
s1238	14	14		18			
s1423	17	5		74			
s400	3	6		21			
s382	3	6		21			
s298	3	6		14			
s526	3	6		21			
s386	7	7		6			
s1488	8	19		6			
s1494	8	19		6			
s832	18	19		5			
s820	18	19		5			
s420.1	18	1		16			
s953	16	23		29			
s641	35	24		19			
s713	35	23		19			
s5378	35	49		179			
s for sequential							