This article has been accepted for publication in IEEE Design and Test of Computers but has not yet been fully edited.
Some content may change prior to final publication.

Hardware IP Protection during Evaluation Using Embedded Sequential Trojan

Seetharam Narasimhan, Case Western Reserve University
Rajat Subhra Chakraborty, Indian Institute of Technology, Kharagpur
Swarup Bhunia, Case Western Reserve University

Abstract—
Evaluation of hardware Intellectual Property (IP) cores is an important step in an IP-based system-on-chip (SoC) design flow. From the perspective of both IP vendors and Integrated Circuit (IC) designers, it is desirable that hardware IPs can be freely evaluated before purchase, similar to their software counterparts. However, protection of these IPs against piracy during evaluation is a major concern for the IP vendors. Existing solutions typically use encryption and vendor-specific toolsets, which may be unacceptable due to lack of flexibility to use in-house or third-party design tools. We propose a novel low-cost solution for hardware IP protection during evaluation, by embedding a hardware Trojan inside an IP in the form of a finite state machine (FSM) with special structure. The Trojan disrupts the normal functional behavior of the IP on occurrence of a sequence of rare events, thereby effectively putting an "expiry date" on the usage of the IP. The Trojan is structurally and functionally obfuscated, thus protecting against potential reverse engineering efforts that target isolation of the Trojan circuit.

Keywords – Hardware IP Protection, IP Evaluation, Hardware Trojan, IP Piracy

I. INTRODUCTION

Reuse-based System-on-Chip (SoC) design using hardware Intellectual Property (IP) cores has become a pervasive practice in the industry to realize bug-free complex SoCs under aggressive time-to-market target [1]. These IP cores usually come in the form of synthesizable Register Transfer Level (RTL) descriptions (*Soft IP*), or gate-level designs directly implementable in hardware (*Firm IP*), or GDS-II design database (*Hard IP*). During the life-cycle of an integrated circuit (IC), these IPs are vulnerable to various security issues as shown in Fig. 1(a). The cost of IP infringement in the United States was estimated to be crossing $1 billion per day in 1998 [2] with a large contribution coming from hardware IPs. These IPs are highly vulnerable to piracy issues at different stages. Other security threats include reverse-engineering efforts to facilitate cloning, counterfeiting, or re-marking of ICs as well as malicious alterations by untrusted third-party vendors. Existing solutions to protect IPs from piracy and reverse-engineering include passive defenses like watermarking [1] as well as active defenses like encryption (coupled with requirement to use vendor-specific tools) [9], hardware metering [7], and obfuscation [12].

Often, the IP vendors allow *evaluation versions* of their IPs to be downloaded and evaluated by the IC designers [3]. This is an important part of their business, because it enables wide publicity of their product leading to increased market share. It also helps them to get feedback about their product from potential customers. From the designers' perspective, IP evaluation is an important step as well, since it helps them explore alternative IP cores with regard to correctness, quality (power, performance, die-area), configurability, testability, and compatibility with other modules in an SoC design. Although the practice of IP evaluation before licensing is simultaneously encouraging to both IP vendors and SoC designers, unfortunately it makes the IP highly vulnerable to piracy. It creates the possibility of a design house illegally using an IP in an IC design or selling it to external parties without paying the license fee to the IP vendor.

To prevent IP piracy, the IP vendors traditionally enforce a binding licensing agreement with the design house. Alternatively, they provide an IP in encrypted form. The decryption process is accomplished with a vendor-specific design platform for simulation and synthesis [4], as illustrated in Fig. 1(b). The latter approach is prevalent in Field Programmable Gate Array (FPGA) based design framework [9]. On the other hand, some IP vendors allow simulation of the downloaded IP, but do not allow it to be synthesized to gate-level designs or bit-streams (for FPGA platforms) [3]. Such practices, however, force an SoC designer to evaluate only the IP's functional behavior, but not the important quality parameters.

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

To overcome the shortcomings of existing IP evaluation practices, a recent industry initiative has resulted in a design platform, which allows designers to download and use encrypted IPs from vendor websites. The synthesis and simulation tools in this design platform are capable of working in a user-transparent manner based on a technology undergoing IEEE standardization [5]. However, it mandates the use of a particular design platform throughout the design flow, which may not be acceptable for modern SoC designers, who typically use different software tools from diverse vendors as well as in-house design tools.

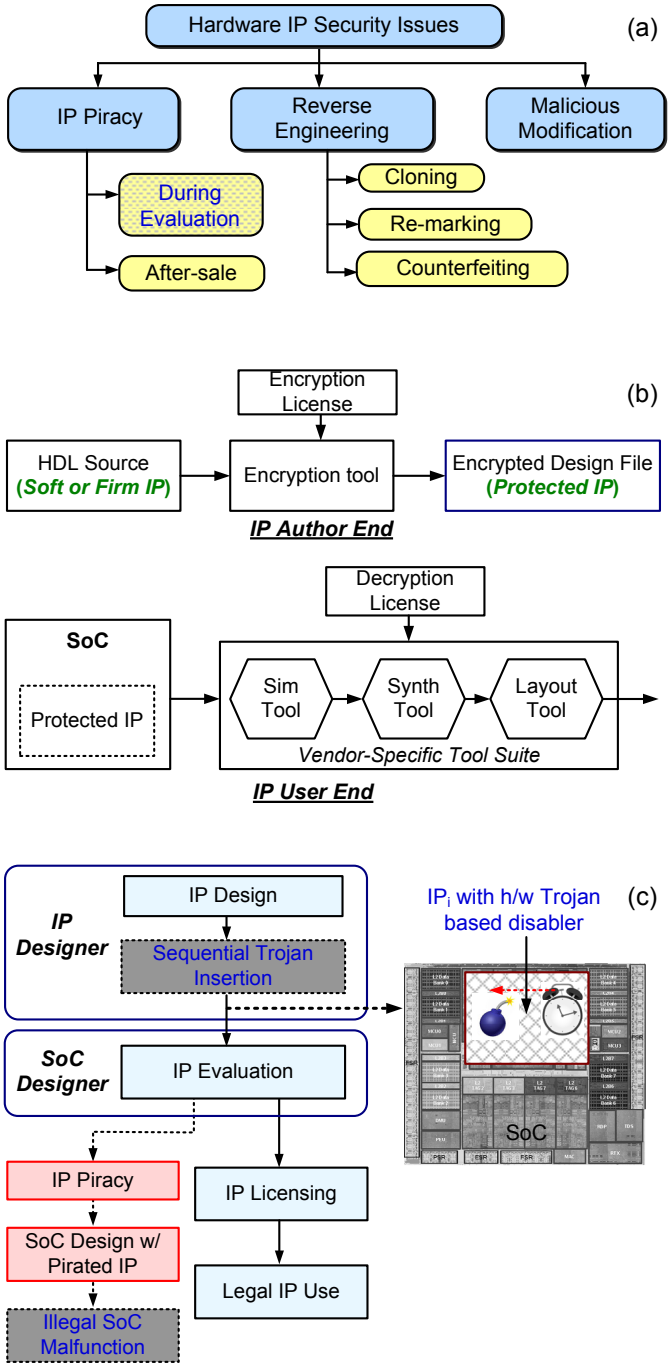

Fig. 1: (a) Taxonomy of hardware IP security issues; (b) traditional IP protection approach using encryption and vendor-specific toolset; (c) the proposed design flow for protecting hardware IP during evaluation.

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

In this paper, we propose a novel low-cost IP protection technique during IP evaluation. It effectively enables a vendor to impose an *expiry date* on the evaluation copy of a hardware IP. It mimics the protection approach for evaluation versions of commercial software, which can be downloaded from vendor websites. The proposed technique is based on embedding a specially-crafted Finite State Machine (FSM), which follows the structure of a sequential hardware Trojan (SHT) in the evaluation copy of a hardware IP. An SHT represents a malicious design alteration, realized by an FSM that triggers abnormal functionality on a sequence of rare events inside a design [11]. Fig. 1(c) illustrates the proposed design flow. An illegal SoC containing a pirated evaluation copy of an IP would cease to follow the specified functionality after the evaluation period due to the presence of a sequential Trojan, which acts like a "hardware time-bomb". To prevent potential reverse engineering of the modified IP aiming to isolate the Trojan, we implement low-overhead *design obfuscation* techniques. We also propose two options for an IP vendor to distinguish the legally sold version of an IP from its evaluation version containing Trojan: (a) a mechanism to de-activate the Trojan at power-on using a disabling key, or (b) providing a Trojan-free version. The proposed approach is language and platform-independent and, hence, can be used in all forms of IP.

II. BACKGROUND

A. *Related Work on Hardware IP Protection*

With increasing vulnerability of hardware IPs to piracy, investigation of IP protection techniques has become an emerging area of research. Recent investigations have targeted hardware IP protection benefiting IP vendors [1], [6], IC designers [7], or both [12]. A well-researched *passive* approach is *digital watermarking* which incorporates a hard-to-remove *digital signature* in an IP. It helps to establish the ownership in case of litigation [1]. Some *soft IP* protection methods perform string modifications of the RTL plain-text to obfuscate it by affecting its human comprehensibility [6]. However, they do not affect the *black-box functionality* of the IP. In [7], an *IC protection* technique ensures that every instance of an IC manufactured in the foundry requires an instance-specific enabling pattern from the IC designer to be operational. It prevents the manufacturing of illegal *clones* of an IC in a fabrication house. In [12], a gate-level IP obfuscation technique has been proposed that allows the IP to be used only after a pre-defined *initialization key* (sequence of input vectors) is applied.

Note that these IP protection techniques are not directly applicable for protecting evaluation versions of an IP. The technique proposed in [6] does not prevent an SoC designer from stealing the IP, cloning it, or performing illegal fabrication, while the digital watermark [1] does not affect the functionality and usability of a stolen IP. The techniques in [7] and [12] are only useful for protection of an IP post-evaluation, because the *initialization key i*s provided to a trusted design house after legal purchase. In this work, we propose a low-cost protection technique for evaluation version of hardware IP. It provides a designer adequate flexibility to evaluate it, while protecting the interest of the IP vendors.

B. *Hardware Trojan*

The issue of *hardware Trojan* has emerged recently due to the widely prevalent industrial practice of fabrication of ICs in potentially untrusted foundries [11]. *Hardware Trojans* are malicious modifications of a circuit that can cause it to fail during deployment, with potentially disastrous consequences. These Trojans would typically evade detection during conventional post-manufacturing test because they trigger malfunction only under extremely rare conditions unlikely to be exercised during test. Moreover, the Trojans can be tiny relative to the original circuit which makes them difficult to detect by *side-channel measurements* [11]. Some Trojan circuits, referred to as *sequential Trojans*, represent FSM structures, which go through a sequence of state transitions before getting activated. On activation, they trigger malfunction, typically by altering logic values at some payload nodes or by leaking secret information.

III. METHODOLOGY

In this work, we leverage sequential hardware Trojans for the beneficial purpose of IP protection to prevent illegal usage of the evaluation version of a hardware IP in chip fabrication. The proposed method consists of three major steps: **(a) *Trojan Design*; (b) *Trojan Insertion*** and **(c) *Trojan Obfuscation***. The design flow is shown in Fig. 2. In the first step, a pool of Trojans is designed for insertion into different IP instances. In the second step, the IP designer judiciously integrates the Trojan with the IP netlist, such

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

that the inserted Trojan allows a sufficient evaluation period before it is activated. Finally, the inserted Trojan is well-hidden by circuit obfuscation to make it difficult for an adversary to reverse engineer. Next we describe these three steps in detail.

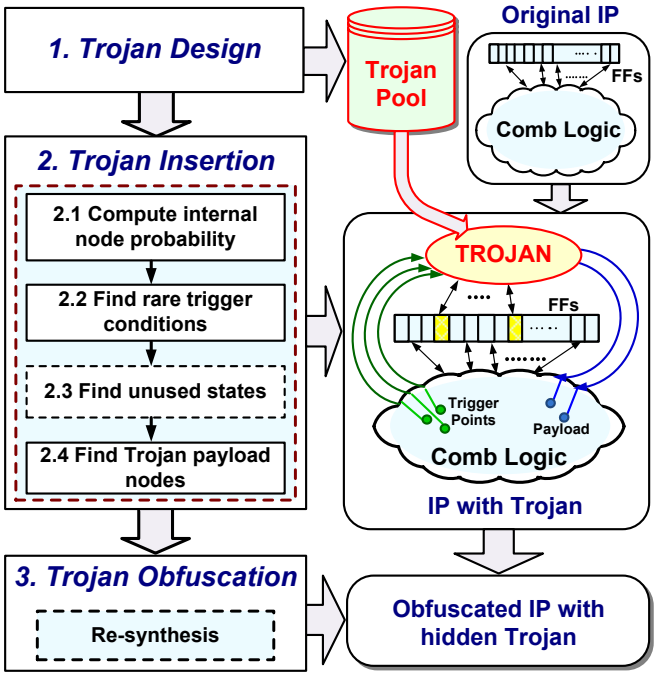

Fig. 2: Major steps in the proposed design flow for the protection of evaluation version IPs.

A. *Trojan Design*

First, the IP designer needs to design one or more Trojans which will let the circuit function properly during the evaluation phase and trigger only after a large number of clock cycles have elapsed. Note that, unlike evaluation copies of software, there is no way to assign a deterministic "expiry date" in the absence of an always-on global timer. A counter circuit [9] which triggers malicious modifications after a fixed number of clock cycles needs to contain a large number of flip-flops, leading to unacceptable design overhead. Besides, these counters can be easily located and isolated by an attacker.

Fig. 3(a) shows the FSM of an example Trojan. At power-on, the state machine starts at the initial state S_0. It moves from state S_{i-1} to state S_i if the condition C_i is satisfied. Since C_is are chosen to be rare conditions, the number of cycles required to reach the "Trojan activation state" (S_T) can be significant, even for small state machines. After traversing some intermediate states during which normal operation of the circuit is ensured, the circuit enters a state (S_T) where the Trojan is activated and its output can be used to modify several *payload* nodes of the circuit. To avoid self-loops in the state diagram where the Trojan flip-flops can be identified due to lack of activity, one can expand each state into a group of states as shown in Fig. 3(b). By increasing number of Trojan activation states and Trojan output nodes which are alternately asserted after activation, such Trojans can cause maximum impact to the circuit while avoiding detection. If the *evaluation* copy of the IP is the same as the *sale* version, the IP vendor needs to keep a provision for disabling the Trojan state machine by providing a *disabling sequence* $\{P_i\}$ at the primary inputs, denoted as the "key". This key will be provided to an IP customer who legally acquires the IP post-evaluation. Note that a possible attack model to bypass the Trojan would be to detect the Trojan effect and reset the FSM. However, such repeated error detection followed by system reset during deployment of an SoC containing the stolen IP would adversely affect design overhead and user experience.

The activation time of the inserted Trojan depends on the actual Boolean logic used as Trojan state transition condition and the input vectors. Let the *Trojan activation time* (t_{active}) be a random variable following a probability distribution $p(t_{active})$ with maxima μ, as shown in Fig. 3(c). The *Expected Time of Trojan Activation* (T_{mean}) is defined as the mean number of clock cycles after which an embedded Trojan

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

is activated, during simulation or during post-fabrication deployment. The *minimum evaluation period* T_{eval} should satisfy the condition:

$P_1 = P(t_{active} \leq T_{eval}) = \int_0^{T_{eval}} p(t_{active})\, dt_{active} < \varepsilon_1$ (1)

Similarly, the upper limit T_{max} needs to satisfy the condition:

$P_2 = P(t_{active} \geq T_{max}) = \int_{T_{max}}^{\infty} p(t_{active})\, dt_{active} < \varepsilon_2$ (2)

The goal is to design a Trojan such that ε_1 and ε_2 are minimized for a given T_{eval} and T_{max}. This can be ensured by a proper choice of triggering nodes. Suppose the probability of the Trojan transitioning from S_{i-1} to S_i is given by p_i, $1 \leq i \leq N+1$. This is essentially a *Markov Process*. Hence, the probability of Trojan activation, is simply the product of all state transition probabilities. Once in state S_{i-1}, the probability of the Trojan staying there is $1 - p_i$. Hence, on average, the Trojan spends $(1 - p_i) \cdot 2^M$ cycles in state S_{i-1}, where M is the total number of primary inputs and state elements in the original circuit. Hence,

$T_{mean} = \sum_{i=1}^{N+1} (i - p_i).2^M$ (3)

For instance, with a modest number of states $N = 20$, $p_i = 10^{-5}\ \forall_i$ and $M = 50$, $T_{mean} \approx 2.25 \times 10^{16}$ cycles.

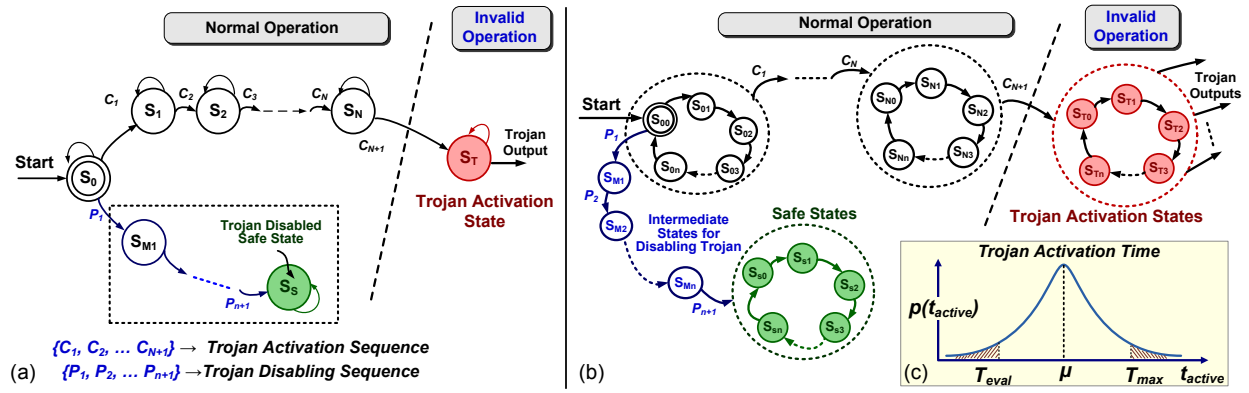

Fig. 3: (a) Trojan state diagram: The sequence of rare conditions $C_1, C_2, \ldots, C_{N+1}$ causes the state machine to reach the Trojan state S_T. Application of the disabling sequence $P_1, P_2, \ldots, P_{n+1}$ on power-on causes the circuit to reach a safe state S_S. (b) To avoid identification of stuck-at states in the Trojan state machine, each state with a self-loop can be expanded into a group of states with non-rare transitions within the group. (c) Probability distribution of Trojan Activation Time t_{active} can be represented by a bell-shaped curve where μ may not be equal to T_{mean}.

B. *Trojan Insertion*

Once the Trojan design is complete, the Trojan is inserted by judicious choice of trigger and payload nodes for the Trojan from among the internal circuit nodes.

Trojan trigger: To determine the Trojan trigger nodes, a set of random vectors is generated; the circuit is simulated using this set of vectors, and the signal probability of the internal nodes is estimated. The nodes with signal probability below a given *threshold* are defined as *rare nodes*. From this set of *rare nodes*, the required number of trigger nodes for the Trojan are chosen with corresponding rare logic values, to construct the Trojan state transition conditions.

Trojan payload: Next, the fanout (*FO*) and fanin (*FI*) cones of each internal node are enumerated and a weighted normalized metric (P_n) [12] based on their sizes is used for ranking them.

$$P_n = 0.5 \times \left(\frac{|FI_n|}{\max(|FI_n|)} + \frac{|FO_n|}{\max(|FO_n|)} \right) \quad (4)$$

Nodes with a higher value of P_n are chosen for modification, to ensure that the effect will be propagated to large parts of the IP. The payload nodes can be selectively inverted upon Trojan activation using XOR gates. On the other hand, they can be set to reveal a pre-determined signature indicating that the IP has crossed its "expiry date". One way to do this would be to force the state elements to an originally unreachable" state [8] upon Trojan activation. To increase the malicious effect of the inserted Trojan, multiple groups of state elements can be forced to unreachable states by the different Trojan outputs. This also helps in obfuscating the Trojan and makes it difficult to bypass the Trojan using error detection and reset circuit.

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

C. *Trojan Obfuscation*

After the Trojan is inserted, it is important to hide it effectively inside the IP to prevent an adversary from reverse-engineering the IP and isolating the Trojan. This is done by resynthesizing the modified design to generate a single flattened netlist. The logic optimization and resource sharing steps during re-synthesis help in reducing the area and performance overhead. A potential adversary can resort to the following possible approaches of unveiling the inserted Trojan:

1) *Functional simulation based identification of Trojan circuitry*: By using high-speed simulation or hardware emulation, the malfunction due to the Trojan can be observed after a time interval $T \ll T_{mean}$ for a typical clock frequency. Then the adversary has to "unroll" the entire state machine in the IP back in time, in order to identify the Trojan circuitry. However, such "FSM unrolling" usually involves Boolean justification of enormous combinational circuits, which is computationally an *NP-complete* problem. Because of the inter-mingled Trojan flip-flops in the flattened netlist, it becomes even more challenging to detect the correct state of the unmodified IP.

Functional simulation can also be used to identify rare nodes as potential Trojan triggers. However, the adversary needs to identify the particular rare event that triggers the Trojan, which is derived by combining many such nodes. The number of such rare events is an exponential function of circuit nodes. For example, in case of an 8-bit ALU circuit (*c880*) with 451 internal nodes and about half of them having signal probability of 0.2 or less, the total number of rare conditions comprising of 4 nodes or less is ~10^9. Moreover, a rare event can be derived by combining both rare and non-rare events, which significantly increases the number of possible rare Trojan trigger conditions.

2) *Side-channel analysis or logic testing based Trojan detection*: The logic testing based approaches [11] aim to cause rare events at internal nodes to trigger arbitrary Trojans while the side-channel based approaches [11] aim at observing Trojan effect on a physical parameter, such as the power trace or the critical path delay. It should be noted that, here, the attacker is trying to identify the Trojan and disable it so that the IP can be used illegally. Hence, it is not enough to simply detect that the IP contains a Trojan. Moreover, in the absence of a golden design, it is difficult to understand which gates of the circuit are "original" and which belong to the Trojan.

3) *Structural analysis through formal verification*: This is the best possible attack scenario for an adversary to identify the Trojan-related design modifications, assuming he/she has a functionally equivalent design (e.g. an earlier generation sale version of the IP). First, the adversary derives the F failing nodes using formal verification. For each failing node the node modification scheme is given by $f_{mod} = f \cdot \overline{en} + g \cdot en$, where *en* is the Trojan output which modifies the node *f* to the function *g* (= $\overline{f}$, for XOR). To detect the effect of a particular *en* signal, the adversary should be able to represent the *Reduced Ordered Binary Decision Diagram* (ROBDD) of the modified node with the *en* signal as the root node. Finding the correct ROBDD representation for a node with fanin cone size f_i has a computational complexity of $O(2^{f_i})$. Next, establishing equivalence for one of the sub-graphs of the root node en with the graph for f through graph isomorphism has a computational complexity $2^{O(\sqrt{f_i \log f_i})}$. The combined problem of complexity $O(2^{f_i} \cdot 2^{O(\sqrt{f_i \log f_i})})$. must be solved for each of the *F* nodes failing formal verification. Due to semantic obfuscation whereby the node names are changed, the adversary has to perform compare point matching to identify each of the failing verification nodes in the original circuit, which has an added complexity of $(S_N)!$ combinations, where S_N is the number of failing state elements.

Hence, the computational complexity of this approach can be quantified as an Obfuscation Metric:

$$M_D = F \cdot 2^{\overline{f_i}} \cdot 2^{(\sqrt{\overline{f_i} \log \overline{f_i}})} + (S_N)! \quad (5)$$

where $\overline{f_i}$ is the average fanin-cone size of each node failing formal verification. Any increase in the value of this metric implies increase in the computational effort required by an attacker to isolate the Trojan, which in turn is indicative of the obfuscation achieved by the scheme. Based on this metric, the following guidelines can be used by the IP designer to design a well-obfuscated Trojan:

- Those nodes which have larger fanin cones should be preferably modified.
- Forcing unreachable states on the state elements of the original circuit upon Trojan activation increases the practical computational complexity of its detection, since the inputs of the modified state elements would contribute in increasing f_i.
- Modification of a larger number of nodes increases *F*, which in turn increases the level of obfuscation.
- A Trojan with larger number of flip-flops increases its obfuscation level slightly because S_N increases.

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

From these observations, it is evident that more complex Trojan design and insertion to attain high levels of obfuscation incur greater design overhead. Hence, the IP designer has to make a trade-off between design overhead and the level of security achievable through obfuscation.

In the case where the *sale* copy of the IP still contains the Trojan and a disabling key is provided to the IP customer who legally acquires it, the attacker will try to get hold of the disabling key to illegally use the IP or distribute the key. The first step to prevent this is to use a different random key for different evaluation instances of an IP. Another way to thwart such malicious attempts is to use Trojans with varying structure, randomly chosen from the pool of Trojans, in different evaluation copies.

IV. RESULTS

To demonstrate the feasibility of the proposed approach, we applied it to several sequential benchmark circuits (ISCAS-89) as well as two open-source Verilog IP cores - the *Advanced Encryption Standard* (AES) and the 2-D *Discrete Cosine Transform* (DCT) from [10]. Sequential Trojans of three different sizes (number of states - 4, 16 and 64, respectively) were designed and inserted in the synthesized gate-level circuits. These Trojan designs are referred to as Trojans Type I, II and III. Table I shows the area and power overhead for these Trojans at iso-delay for some of the benchmark circuits. The circuit delay was kept unaffected by avoiding Trojan insertion in critical timing paths and by specifying the same delay constraint for both original and modified versions of the circuits during re-synthesis. We can note from Table I that the area overhead increases with the size of the Trojan circuit, while the area overhead decreases with an increase in the size of the original IP. Hence, the size of the original circuit should guide the choice of the Trojan to be inserted.

TABLE I: Area and power overhead results for five ISCAS-89 benchmark circuits and two open-source IP cores with different Trojan sizes, and Obfuscation metric (*log* M_D) for two different payload schemes: Case I selects the nodes randomly, Case II selects nodes with higher P_n.

IP	Trojan Type (No. of states)	Overhead (%)		Obfus. Metric	
		Area	Power	Case 1	Case 2
s5378	Trojan I (4)	2.05	-12.29	166.3	247.9
	Trojan II (16)	6.24	-6.75		
	Trojan III (64)	31.04	6.61		
S9234	Trojan I (4)	3.31	1.08	101.3	248.9
	Trojan II (16)	13.31	5.14		
	Trojan III (64)	35.96	18.57		
S15850	Trojan I (4)	0.91	0.97	498.7	819.5
	Trojan II (16)	1.84	0.52		
	Trojan III (64)	5.69	0.77		
S35932	Trojan I (4)	0.34	4.53	1202.2	2050.4
	Trojan II (16)	0.76	4.66		
	Trojan III (64)	2.25	2.42		
S38584	Trojan I (4)	0.20	0.88	877.3	3624.5
	Trojan II (16)	0.62	-0.28		
	Trojan III (64)	2.13	1.13		
DCT	Trojan I (4)	0.70	2.78	12.3	57.4
	Trojan II (16)	0.94	2.68		
	Trojan III (64)	2.07	2.60		
AES	Trojan I (4)	0.34	0.05	10.8	69.1
	Trojan II (16)	0.44	0.02		
	Trojan III (64)	0.66	0.01		

To estimate the level of obfuscation achieved by the proposed method, we performed formal verification using Synopsys *Formality* tool between a benchmark and its modified version. We observed large number of verification failures for different Trojan payload schemes, such as XOR-ing internal nodes and forcing state elements to unused states. This shows that the resultant design after Trojan insertion

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

and obfuscation has significant structural difference from the original. We also evaluated the values of the obfuscation metric M_D (in logarithmic scale), as presented in Table I for F = 10. The log M_D values show that a judicious choice of payload nodes with high fanins (Case II) provides higher level of obfuscation. We observed similar increases in log M_D for increasing F, while it was almost independent of the number of Trojan flip-flops.

To get an estimate of the number of cycles required for activation, the DCT IP core with different inserted Trojans was simulated with 100,000 random input vectors. The state transition condition was formed using nine rare node values. The number of cycles it took for the Trojan to activate (i.e. the "*activation cycles*") in each case is plotted in Fig. 4(a) in logarithmic scale vs. the number of bits in the Trojan state machine. This plot clearly demonstrates the exponential dependence of *activation cycles* on the number of bits in the Trojan state machine.

Since the number of *activation cycles* is also a function of rareness of the state transition condition, it can be increased by changing the number of Trojan trigger nodes. We calculated the probability of the rarest condition that could be achieved using different number of rare nodes in a single IP. Fig. 4(b) shows that activation probability decreases with increase in the number of trigger nodes, which increases the number of cycles for Trojan III activation for the ISCAS-89 circuit *s35932*. Similar trends were observed for other benchmark circuits and IPs. From these results, we can infer that the proposed approach is capable of providing high levels of security at nominal area and power overhead. The level of security as well as overhead can be adjusted through choice of Trojan design parameters.

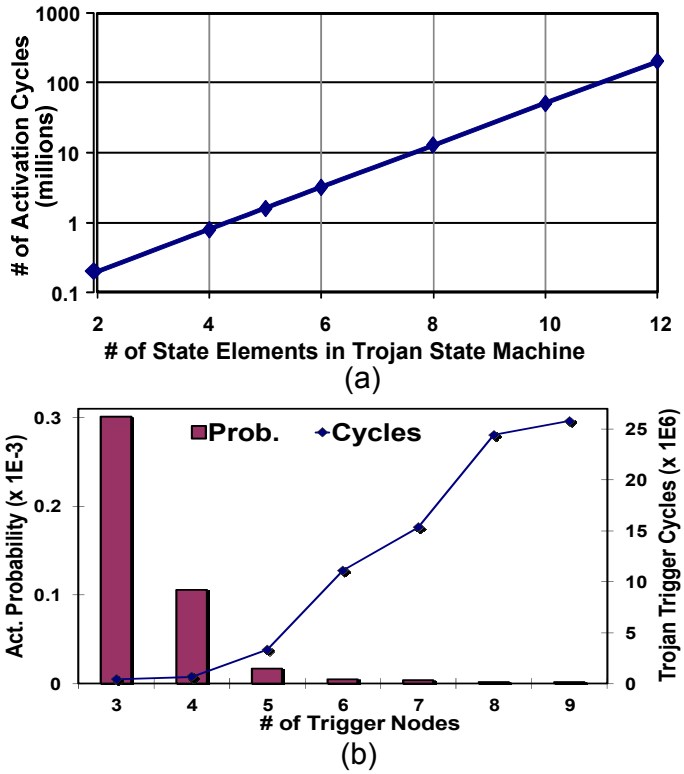

Fig. 4: (a) Dependence of *activation cycles* on size of inserted Trojan. (b) Dependence of activation probability of trigger condition on number of trigger nodes and corresponding number of *activation cycles* for Trojan III.

V. CONCLUSION

We have presented a low-cost design approach for protecting the evaluation version of hardware IPs from potential piracy. It exploits the properties of sequential hardware Trojan circuits, which are activated through a sequence of rare events. The proposed approach is suitable for all forms of hardware IP and for both SoC and FPGA design platforms. It allows a designer to evaluate an unencrypted IP using a

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE

preferred design flow and toolsets. Judicious choice of triggering condition can help to control the Trojan activation time, thus ensuring sufficient time to evaluate an IP. The Trojan remains well-obfuscated in the IP which makes it computationally challenging for an adversary to isolate it using functional or structural analysis.

The Trojans can be designed to produce recognizable signature at the primary outputs which can help an evaluator identify the expected wrong behavior. Potential use of an evaluation copy in a fabricated IC can be detected during the prolonged system integration and testing step. It helps us avert possible harm to innocent end-users due to illegal IP usage. The proposed approach also provides additional benefit of embedding a *digital watermark* or authentication feature at minimal design overhead to increase the level of post-sale security. Finally, the approach can be combined with other IP protection techniques such as key-based IP locking to achieve comprehensive IP security.

REFERENCES

[1] E. Castillo, U. Meyer-Baese, A. Garcia, L. Parrilla and A. Lloris, “IPP@HDL: Efficient Intellectual Property protection scheme for IP cores”, *IEEE Trans. on VLSI*, pp. 578-590, 2007.
[2] VSI AllianceTM White Paper, “Intellectual Poperty Protection: Schemes, Alternatives and Discussion”. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.9058&rep=rep1&type=pdf
[3] R. Goering, “Synplicity initiative eases IP evaluation for FPGAs”. [Online]. Available: http://www.scdsource.com/article.php?id=170
[4] T. Batra, “Methodology for protection and licensing of HDL IP”. [Online]. Available: http://www.us.design-reuse.com/news/?id=12745\&print=yes
[5] “Recommended practice for encryption and [Use Rights] management of electronic design Intellectual Property (IP)”. [Online]. Available: http://www.eda.org/twiki/bin/view.cgi/P1735/WebHome
[6] “ThicketTM family of source code obfuscators”. [Online]. Available: http://www.semdesigns.com
[7] Y. Alkabani, F. Koushanfar and M. Potkonjak, “Remote activation of ICs for piracy prevention and digital right management”, *Intl. Conf. on CAD*, 2007.
[8] H.-C. Liang, C.-L. Lee and J.-E. Chen, “Invalid state identification for sequential circuit test generation”, *Asian Test Symposium (ATS)*, 1996.
[9] Altera Corporation, “US Patent 7234159 - Method and apparatus for controlling evaluation of protected intellectual property in hardware”. [Online]. Available: http://www.patentstorm.us/patents/7234159.html
[10] [Online] http://www.opencores.org
[11] M. Tehranipoor and F. Koushanfar, “A survey of hardware Trojan taxonomy and detection”, *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10-25, 2010.
[12] R.S. Chakraborty and S. Bhunia, “HARPOON: An obfuscation based SoC design methodology for hardware protection”, *IEEE Trans. on CAD of Integrated Circuits and Systems*, Oct. 2009.

Seetharam Narasimhan is pursuing a PhD in computer engineering at Case Western Reserve University, Cleveland, Ohio. His research interests include low power and robust design, implantable electronics and hardware security. He has a BE in electronics and telecommunication engineering from Jadavpur University, India.

Rajat Subhra Chakraborty is an assistant professor of computer science and engineering at Indian Institute of Technology, Kharagpur, India. His research interests include hardware security and low power and robust design. He has a PhD in computer engineering from Case Western Reserve University with a focus on hardware IP protection and hardware Trojan detection.

Swarup Bhunia is an assistant professor of electrical engineering and computer science at Case Western Reserve University, Cleveland, Ohio. His research interests include low power and robust design, hardware security and implantable electronics. He has a PhD in electrical engineering from Purdue University with a focus on yield-aware, low-power and testable design approaches.

S. Narasimhan is with the Department of Electrical Engineering and Computer Science, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, Ohio, USA - 44106. E-mail: sxn124@case.edu.
R.S. Chakraborty is with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal, India. E-mail: rschakraborty@cse.iitkgp.ernet.in.
S. Bhunia is with the Department of Electrical Engineering and Computer Science, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, Ohio, USA - 44106. E-mail: skb21@case.edu.
Phone: +1-216-3685550 Fax: +1-216-3686039

Digital Object Indentifier 10.1109/MDT.2011.70 0740-7475/$26.00 2011 IEEE