**HARDWARE IP PROTECTION DURING EVALUATION USING EMBEDDED SEQUENTIAL TROJAN**

**ABSTRACT:**

Evaluation of Hardware IP Cores is a crucial step in an IP- based system-on-chip(SoC) design flow. We emphasize on protection of IPs against piracy during evaluation is a major concern for IP vendors. Existing solutions are vendor specific third-party design tools. This deals with Hardware Trojan inside in the form of a finite state machine with special structure. The trojan disrupts the normal functional behaviour of the IP on occurrence of a sequence of rare events, thereby introducing an expiry date or an end date for the IP. The Trojan is structurally and functionally obfuscated, protecting against reverse engineering attacks.

**INTRODUCTION:**

Reuse-based System-on-chip design using hardware Intellectual property(IP) cores has become an important and crucial practice in the industry to realize bug free complex SoCs under aggressive time-to-market target. These IP cores usually come in the form of synthesisable RTL descriptions or gate level designs directly implementable in hardware (Firm IP) or GDS-II design database (Hard IP). During the lifecycle of an integrated circuit, these IP’s are vulnerable to integrated circuit. Industries usually invest in protecting IP’s from piracy and reverse engineering attacks by using watermarking with encryption, hardware metering and obfuscation.

To prevent IP piracy, IP vendors traditionally enforce a binding licensing agreement with the design house. The decryption process is accomplished with a vendor specific design platform for simulation and synthesis. This is done in FPGA based framework while some IP vendors donot allow it to be synthesized to gate level designs or bitstreams. Such practices, force an SoC designer to access the IP’s functional behaviour. To overcome the shortcomings of IP evaluation practices, we can use a IP design platform throughout the design flow by IEEE standardisation.

We propose a novel low-cost technique for IP protection during IP evaluation. WE introduce an “expiry date” on the evaluation copy of the IP h/w. This is done by embedding a specially crafted FSM model which follows a sequential H/w Trojan in the evaluation copy of h/w IP. It triggers a sequence of rare events inside the design. An illegal version would cease to follow the design and will stop working after the evaluation period making the IP act like a “h/w time bomb”. To prevent reverse engineering attacks, we implement obfuscation techniques like a mechanism to de-activate the time bomb at power on using a disabling key or providing a Trojan free version.

**METHODOLOGY:**

The three important steps in SHT is Trojan Design, Trojan Insertion and Trojan Obfuscation. In the first step, a bunch of Trojans is inserted into different IP instances. In second step, IP designer integrates the Trojan with the IP netlist such that it allows a evaluation period before it is activated. Finally, it is well hidden by circuit obfuscation to make it tedious for reverse engineers.

A picture containing screenshot

Description generated with high confidence

**TROJAN DESIGN:**

IP designers should let the trojan work properly until a large clock cycles have elapsed. There is no way to assign an “expiry date” in the absence of an always on global timer. A counter circuit which triggers malicious modification after a fixed number of clock cycles needs to contain many flip flops, leading to unacceptable design overhead. These counters can be isolated by the attackers. At power-on the state machine starts at the initial state S0. It moves from state *Si−1* to state *Si* if the condition *Ci* is satisfied. Since *Ci*s are chosen to be rare conditions, the number of cycles required to reach the “Trojan activation state” (*ST*) 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 (*ST*) 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. 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* {*Pi*} at the primary inputs. The activation time of the inserted Trojan depends on the actual Boolean logic used as Trojan state transition condition. Let the *Trojan activation time* (*tactive*) be a random variable following a probability distribution *p*(*tactive*) with maxima *μ*. The *Expected Time of*

*Trojan Activation* (*Tmean*) is defined as the mean number of clock cycles after which an embedded Trojan is activated, during simulation or during post-fabrication deployment. The *minimum evaluation period Teval* should satisfy the condition:

The goal is to design a Trojan such that ε1 and ε2 are minimized for a given *Teval* and *Tmax*. This can be ensured by a proper choice of triggering nodes. Suppose the probability of the Trojan transitioning from *Si−1* to *Si* is given by *pi*, 1<i<N + 1. This is essentially a *Markov Process*.

**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 (*Pn*) [12] based on their sizes is used for ranking them:

Nodes with a higher value of *Pn* 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.

**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 h/w emulation, the malfunction of a trojan can be observed after a time interval T<< Tmean for a typical clock frequency. The adversary must unroll the entire state machine in the IP back in time, to identify the Trojan circuitry. It involves Boolean justification of enormous combinational circuits furthermore it is even more challenging to detect the correct state of the unmodified IP.

2. Side Channel analysis or logic testing based trojan detection: It is done by observing the power trace or by the critical path delay. Here attacker tries to detect and disable the Trojan. Moreover, in the absence of the design, it is difficult to detect which gates of the circuit are “original” and which belong to the trojan.

3. Structural analysis through formal verification: This is the best attack for Trojan design modifications, having in mind the functionally equivalent designs. First, the adversary derives the F

failing nodes using formal verification. For each failing node the node modification scheme is given by *fmod = f ·+ g · en*, where *en* is the Trojan output which modifies the node *f* to the function *g* (= 􀝂􀇡 for XOR).To detect the effect of a ensignal, 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 *fi has* a computationalcomplexity. Next, establishing equivalence for one of the sub-graphs of the root node en with thegraph for f through graph isomorphism has a computational complexity. The combined

problem of complexity O. must be solved for each of the *F* nodes failing formal verification. Due to semantic obfuscation whereby, the node names are changed, the adversary must perform compare point matching to identify each of the failing verification nodes in the original circuit, which has an added complexity of (*SN*)! combinations, where *SN* is the number of failing state elements.

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

**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

varied sizes (number of states - 4, 16 and 64, respectively) were designed and inserted in the

synthesized gate-level circuits.