Sequential Hardware Trojan: Side-channel Aware Design and Placement

Xinmu Wang[1], Seetharam Narasimhan[1], Aswin Krishna[1], Tatini Mal-Sarkar[2], and Swarup Bhunia[1]
[1]Electrical Engineering and Computer Science Department, Case Western Reserve University
[2]Hathaway Brown High School
Cleveland, OH, USA
{xxw58, sxn124, ark70, skb21}@case.edu[1], tmal-sarkar13@hb.edu[2]

***Abstract*—Various design-for-security (DFS) approaches have been proposed earlier for detection of hardware Trojans, which are malicious insertions in Integrated Circuits (ICs). In this paper, we highlight our major findings in terms of innovative Trojan design that can easily evade existing Trojan detection approaches based on functional testing or side-channel analysis. In particular, we illustrate design and placement of sequential hardware Trojans, which are rarely activated/observed and incur ultralow delay/power overhead. We provide models, examples, theoretical analysis of effectiveness, and simulation as well as measurement results of impact of these Trojans in a hardened design. It is shown that efficient design and placement of sequential Trojan would incur extremely low side-channel (power, delay) signature and hence, can easily evade both post-silicon validation and DFS (e.g. ring oscillator based) approaches.**

Keywords- Hardware Trojan Attacks, Sequential Trojan, DFS

I. INTRODUCTION

Globalization in the IC industry decreases control of a designer on the fabricated chips. Incorporation of 3rd party Intellectual Properties (IPs) or design tools, or outsourcing fabrication to off-shore facilities help to lower cost and meet aggressive time-to-market targets. However, such a manufacturing flow typically involves multiple untrusted parties, who can potentially compromise an IC's functional or parametric behavior in a way that can evade conventional post-silicon validation [1]. Such tampering through malicious modification of a design, referred as hardware Trojan, can be introduced at different steps of the IC manufacturing flow, e.g. 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, especially in security-critical applications such as military, communication and national infrastructure [1-3].

In order to motivate researchers to investigate novel Trojan models and protection approaches, the Polytechnic Institute of New York University has been holding the Embedded Systems Challenge (ESC) [2] competition yearly since 2008, as part of the Cyber Security Awareness Week (CSAW). The competition usually targets hardware Trojan design and insertion techniques [3], hardening mechanisms and Trojan detection approaches. Our participation in ESC 2010 focused on designing novel Trojan attacks against a hardened design, which can bypass protection of existing design-for-security (DFS) based hardening mechanisms [4-5] or testing steps. We designed various ultralow-overhead hard-to-detect hardware Trojans in the test vehicles, and studied models and examples of sequential Trojans as well as efficient placement techniques to bypass the proposed defense mechanisms.

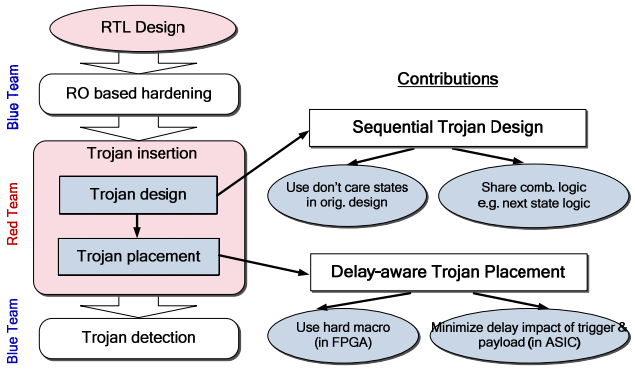

Fig. 1. ESC 2010 competition procedure and our contributions in Trojan insertion as a red team.

Fig. 1 shows the procedure of ESC 2010 competition. The competition was organized as a Red Team – Blue Team approach for hardware Trojan design and detection, using a Digilent BASYS Spartan-3e FPGA platform. The designs provided by the Blue Team were hardened using two defense mechanisms based on delay calibration of different paths of a circuit. The detection mechanism was a side-channel approach whereby any change to the path delays due to inserted Trojans would be reflected in the oscillation frequencies of ring oscillators. Our objective as a Red Team was to insert innovative Trojans, which could evade the detection mechanisms, e.g. conventional functional/structural testing, existing side-channel analysis techniques and the hardening mechanisms, while causing visible malfunction upon Trojan activation. This prompted us to use sequential Trojans with rare trigger conditions which would make the Trojans difficult to activate or observe under normal testing. Other important criteria for Trojan design are minimum impact on delay, power and area of the original circuit, which motivated us to consider novel placement techniques such as delay-aware Trojan insertion and creation of hard macro for the original design (to prevent delay variations in FPGA due to routing changes). Our proposed Trojan insertion mechanism consisting of Trojan design and placement procedures address all the above requirements.

In Section II, we present the concept of hard-to-detect sequential Trojans and discuss their design criteria. In Section III, we provide placement strategies for evading detection mechanisms along with supporting simulation and experimental results. Finally we conclude in Section IV.

II. SEQUENTIAL HARDWARE TROJAN

To prevent hardware Trojans from being detected during conventional post-silicon validation procedures, intelligent attackers are expected to design Trojans which are stealthy in nature. Typically, attackers would insert Trojans that can trigger upon some rare conditions and compromise the security or functionality of the design. Hardware Trojan circuits can either be combinational or sequential [1]. Combinational Trojans are triggered on the occurrence of rare logic values of one or more internal nodes, while a sequential Trojan exhibits its malicious effect after a sequence of rare events during long period of field operation, acting as a time-bomb. Generally, sequential Trojans can be designed to be exponentially harder-to-detect than combinational Trojans by increasing the length of trigger sequence. In fact, these sequential Trojans can be extremely small in size and hard-to-detect during normal post-Si testing. They can also bypass exhaustive testing of a design in full-scan mode.

A. Functional Sequential Trojan Models

A sequential Trojan can be represented as a finite state machine (FSM), where the Trojan trigger sequence is mapped to one of the rarely-satisfied paths in its state transition diagram. The general FSM based model of a sequential Trojan is illustrated in Fig. 2. The next state logic of the Trojan FSM depends on the occurrence of certain rare events, i.e. combinations of rare logic values, of the original circuit's internal nodes. The Trojan circuit undergoes state transition under certain pre-defined rare events in the original circuit; otherwise the Trojan will remain in the current state or go back to the initial state if the expected rare event does not happen. The Trojan output is activated only upon reaching the final Trojan state (S_T), when it affects the payload node compromising the original circuit's normal operation. Next, we provide examples of various types of sequential Trojans.

1) Free-runnning/Enabled Synchronous Counter Trojan: Fig. 3(a) shows a k-bit synchronous counter Trojan with or without an enable signal. A synchronous free-running counter works like a time-bomb where there is no event-dependent trigger condition. The Trojan will get triggered, independent of the operation of the original circuit, and the only design parameter is the time duration for activation of the Trojan (referred as time-to-trigger). It has a deterministic time-to-trigger 2^k-1 clock cycles, where k is the number of state elements in the counter. The drawback of this type of Trojan is the large area/power overhead required in order to guarantee a certain trigger time. By using rare nodes of the original circuit to generate an enable signal for the counter, we can lower the trigger probability and thus greatly increase the time-to-trigger for the same area overhead.

2) Asynchronous Counter Trojan: The asynchronous counter-based Trojan uses an internal signal as the clock for counting the occurences of a rare event. As the example shown in Fig. 3(b), p and q are two rare internal nodes in the original circuit, both of which have the rare logic value of 1. Therefore ANDing p and q creates a signal that seldom switches from 0 to 1, and thus can be used as the clock signal for the counter. By proper choice of the rare events, one can ensure an extremely large time-to-trigger.

3) Hybrid Counter Trojan: To further lower the Trojan trigger probability, a hybrid counter Trojan model is developed as demonstrated in Fig. 3(c). It contains multiple cascaded counters, where the counters can be synchronous or asynchronous, with the clock of the second counter depending on both the first counter state and rare internal events. In the particular example shown in Fig. 3(c), whenever the first counter achieves its maximum value of 2^{k1}-1, if both signals p and q happen to be at their rare value of logic 1, the second counter will be updated.

4) FSM based Trojan: The counter-based Trojans can be generalized to FSM-based Trojans, which contain a sequential and combinational part, with the inputs being derived from rare circuit conditions. The advantage of the FSM-based Trojan is that they can be designed to be arbitratily complicated with same amount of resource and can re-use both combinational logic and flip-flops (FFs) of the original circuit for FSM-hosting. Moreover, unlike counters which are uni-directional, the FSM-based Trojan can have state transitions leading back to the initial state, thus causing the final Trojan state to be reached only if the entire state sequence is satisfied in consecutive clock cycles.

B. Expected Time-to-Trigger

Although the Trojan trigger condition can be deterministic (e.g. counter based Trojan) or probabilistic, the time taken by the inserted Trojan to get activated (time-to-trigger) is not deterministic (except for free-running counter) - instead, it is a pseudo-random value. It depends on the actual Boolean logic used as Trojan state transition function, which gets satisfied based on the actual sequence of input vectors applied to the circuit.

To estimate the expected time of Trojan activation T_{mean}, consider that the Trojan passes through a sequence of states S_1, S_2….S_N before getting activated, as shown in Fig. 2. Suppose the probability of the Trojan transitioning from state S_{i-1} to state S_i is given by p_i, $1 \le i \le (N+1)$, where $N=2^k$-2 and k is the number of state elements. This is essentially a Markov Process, with p_i

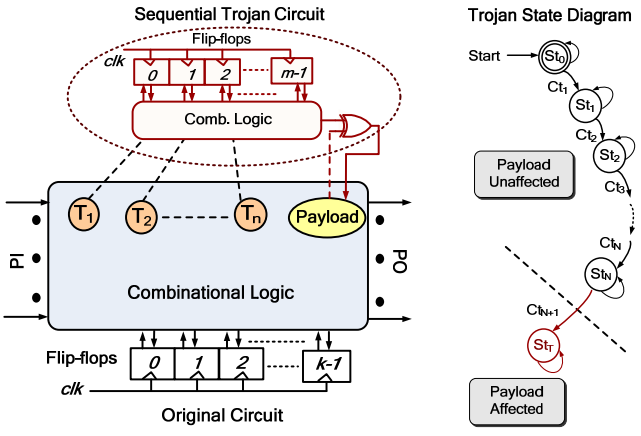

Fig. 2. Sequential Trojan model and Trojan state diagram.

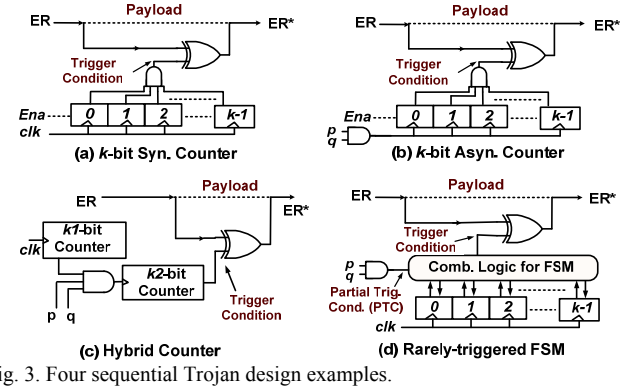

Fig. 3. Four sequential Trojan design examples.

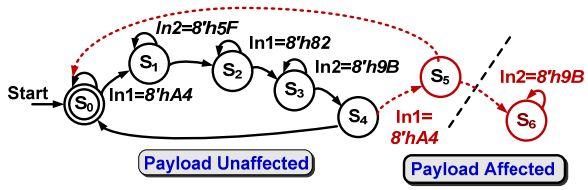

Fig. 4. State diagram of a sequential Trojan with sequential and combinational logic sharing with original circuit.

depending only on the present state S_{i-1} of the Trojan FSM. For simplicity, consider that the Trojan while inactive stays at its present state for all input state space conditions except the unique condition that causes a state transition. Once in state S_{i-1}, the probability of the Trojan staying in state S_{i-1} is $1 - p_i$. Hence, on average, the number of cycles the Trojan spends in state S_{i-1} is:

$$T(S_{i-1}) = p_i.1 + (1-p_i).p_i.2 + (1-p_i)^2.p_i.3 + \cdots \rightarrow \infty$$
$$= \lim_{n\to\infty} \sum_{j=1}^{n} (1-p_i)^{j-1}.p_i.j$$
$$= \lim_{n\to\infty} \frac{1-(1-p_i)^n}{p_i} - n.(1-p_i)^n = \frac{1}{p_i} \quad (1)$$

Hence, the expected time-to-trigger for the Trojan in terms of clock cycles (assume continuous operation): $T_{mean} = \sum_{i=1}^{N+1} \frac{1}{p_i}$ (2)

For an FSM-based Trojan which goes back to the initial state in absence of the rare state transition conditions, the trigger requires a continuous satisfaction of the rare trigger sequence, therefore the trigger probability is: $P(S = S_T) = \prod_{i=1}^{N+1} p_i$ (3)

The Trojan model can be simplified to a two-state FSM containing only the initial state (S_0) and the Trojan state (S_T) where the transition probability from S_0 to S_T is given by equation (3). Since equation (1) is applicable to this one-step model, the expected time-to-trigger is given by: $T_{mean} = \frac{1}{\prod_{i=1}^{N+1} p_i}$ (4)

C. Optimized Implementation

Although in the previously described sequential Trojan model, Trojan state elements are shown separately from those of the original circuit, it is not necessary for sequential Trojan insertions to introduce extra state elements. Instead, they could use existing unused states of the original circuit, if applicable. For example, Fig. 4 shows an FSM of five states, requiring 3 state elements with binary encoding. Here, the unused don't care states (S_5 and S_6) can be leveraged by the attacker to implement a sequential Trojan. Such sequential elements sharing benefits the attackers in both minimizing the area and power overhead, since only the next state logic is modified, as well as in protecting the Trojan from formal verification based approaches. To further reduce the area/power overhead, the Trojan can be carefully designed to reuse the combinational logic of the original circuit. For example, the Trojan state machine in Fig. 4 reuses the transition conditions of the original FSM, whose consecutive occurrence is an extremely rare event in state S_4. Table I demonstrates the area/power overhead due to a sequential Trojan with the same functionality yet different implementations. In particular, Trojan 1 is implemented with extra

TABLE I. AREA / POWER OVERHEAD OF SEQUENTIAL TROJANS OF SAME FUNCTIONALITY BUT VARYING IMPLEMENTATIONS

Overhead* / Design	Area			Power
	Seq.	*Comb.*	*Overall*	
Orig. w/ Troj.1	8.1%	4.3%	5.4%	3.5%
Orig. w/ Troj.2	0	3.4%	2.3%	1.2%
Orig. w/ Troj.3	0	0.8%	0.6%	0.4%

*All designs are synthesized at iso-delay as the original circuit

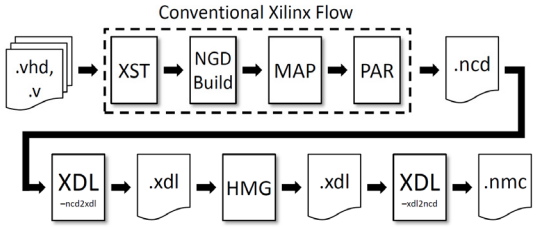

Fig. 5. Hard macro creation flow for the FPGA platform [7].

state elements; Trojan 2 reuses the existing don't care states without next state logic sharing; and Trojan 3 re-uses both state elements and next stage logic, by exploiting existing rare conditions in the combinational logic. For example, in a microprocessor, it is not difficult to find such rare conditions in the memory controller or ALU logic. The power overhead is mainly caused by the leakage power of the sequential Trojans, because dynamic power due to the Trojans is negligible due to their low switching activity.

III. TROJAN PLACEMENT

Recently, Ring Oscillator Network (RON) based design hardening approaches [4, 5] were proposed for hardware Trojan detection through ring oscillator (RO) frequency change. These RON based approaches mainly fall into two categories: One is securing the design by dynamically configuring circuit paths into ROs to monitor undesired design modification [5]; the other is additionally inserted RON to detect voltage drops due to extra Trojan circuitry [4]. The first approach was adopted in the ESC 2010 competition by the "blue team" to harden their design. As an attacker in the competition, we could successfully insert Trojans which evade the hardening/detection mechanism in the FPGA platform. However, we analyze the effectiveness of Trojan insertion with respect to both FPGA and ASIC scenarios. Our analysis shows that attackers can design and insert stealthy Trojans which successfully evade the hardening mechanisms.

In the ESC competition, we were given a 4-bit combinational adder (Beta design) hardened by configuring different paths into ROs during specially-instrumented test modes. In absence of sequential elements in the original design, we resorted to inserting a separate FSM as a rarely-triggered sequential Trojan. However, in an FPGA-based framework, the insertion of any extra circuitry caused the entire design to be re-synthesized and re-routed, resulting in wide fluctuations in the RO frequencies, which were dominated by interconnect routing delays. To mimic an ASIC scenario, where the Trojan is inserted post-layout, we tried to preserve the routing and placement of the original design by making it a Hard Macro. Hard macro generally refers to a pre-compiled module, which can be re-used in its optimized form. Fig. 5 illustrates the flow of creating a hard macro using Xilinx ISE [7]. Since a hard macro consists of previously synthesized, mapped, placed and routed circuitry, the placement and routing of the ROs will not change due to insertion of Trojan circuit. Table II contains

TABLE II. MEASURED RO FREQUENCY CHANGE FOR DIFFERENT TROJANS INSERTED IN THE ESC 2010 COMPETITION DESIGNS

Trojan Type	Beta Easy		Beta Medium	
	RO1	*RO2*	*RO1*	*RO2*
Syn. Count.	1.46%	1.44%	1.59%	2.83%
Syn. Count. w/ En	0.06%	0.49%	2.23%	1.89%
Async. Count.	0.05%	0.83%	0.77%	0.06%
Hybrid count.	0.55%	0.51%	0.85%	1.12%
FSM	3.45%	2.35%	0.80%	3.49%

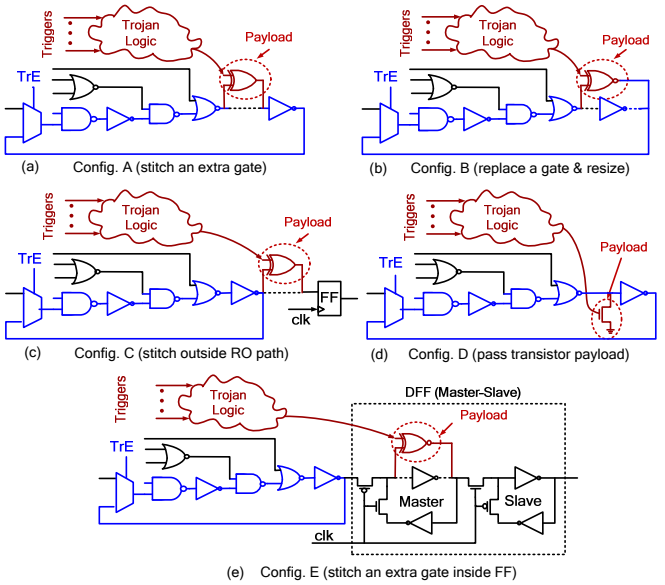

Fig. 6: Different payload insertion approaches: (a) stitching an extra gate (XOR) inside a delay path; (b) replacing an existing gate (e.g. NOT by XOR) and resizing; (c) stitching a gate outside built-in RO path; (d) inserting a NMOS pull-down transistor as payload; and (e) inserting the payload inside a master-slave FF.

the measurement results of RO frequencies from FPGA based framework for the Trojans inserted in the Beta design with hard macro. In the two versions of Beta design (Easy and Medium), different paths were selected to be configured into ROs. The same Trojans inserted in Beta Easy and Medium without hard macro caused 20.01% and 7.47% average RO frequency change respectively. A complete list of our implemented Trojans can be found in ESC website [2]. The only factors causing change in RO frequency are: (i) Load capacitance of certain nodes in the ROs may change due to different routing of non-hard-macro part of the design; (ii) Different placement of the entire hard macro may cause RO frequency change due to intra-die process variations of the FPGA chip. The measured RO frequency changes for different inserted Trojans were found to be within the tolerance of process-induced parameter variations.

Even in the scope of ASIC, dynamically configured ROs cannot guarantee the security of the design, since a clever attacker can always try to bypass them. Even if all paths (and all gates) of the circuit are covered by some ROs, the Trojan can still be inserted without affecting the delay too much. The trigger condition of the Trojan only adds load capacitance to some internal nodes of the circuit, which can be chosen from different ROs. In the Trojan models described in Section II, the payload of the Trojan adds an (XOR) gate delay to the original circuit path, as shown in Fig. 6(a). However, four ways of designing Trojan payload can avoid directly *inserting* gates in RO path: (i) Re-synthesizing the design and re-sizing the gates after insertion of Trojan payload to preserve the path delay. For example, in Fig. 6(b) the Trojan payload is implemented by modifying an inverter to an XNOR gate with the other input coming from the Trojan output, and sizing the gate to have the same delay impact. (ii) Inserting the Trojan payload outside RO paths at a primary output or flip-flop input, so as to add only an extra load capacitance, as shown in Fig. 6(c). This load can be minimized by re-sizing the payload gate capacitance to match the original load capacitance. (iii) The payload can be realized without adding an extra level of gate, e.g. one can simply add an NMOS transistor to the payload node controlled by the Trojan trigger signal to pull the node to 0 as shown in Fig. 6(d), equivalent to a stuck-at-0 fault activated only under rare conditions. This would have virtually no impact on a delay path, since it only adds a diffusion capacitance load, which is much lower than gate capacitance. (iv) Fig. 6(e) provides an example of merging the payload gate into the flip-flop, by replacing one inverter in the D flip-flop with an XNOR gate. In this case, change of the load capacitance cannot be seen by the RO directly thus causes negligible impact. HSPICE simulations of above configurations are performed on Beta design and several ISCAS'85 benchmark circuits using 70nm Predictable Technology Model (PTM) [6] with a supply voltage of 1V at 25ºC, and the results are given in Table III.

TABLE III. IMPACT OF DIFFERENT TROJAN CONFIGURATIONS (AS SHOWN IN FIG. 6) ON RO FREQUENCY, 70NM PTM @1V, 25ºC

Ckt	# of levels in RO path	RO Frequency change*			
		Config. A	*Config. B*	*Config. C*	*Config. D*
Beta	11	7.76%	2.21%	1.80%	0.28%
c880	13	6.40%	2.05%	1.77%	0.26%
c2670	15	5.92%	1.97%	1.51%	0.24%
c3540	15	5.25%	1.76%	1.12%	0.14%
c5315	17	4.38%	1.15%	0.85%	0.11%
c6288	17	3.95%	1.05%	0.74%	0.07%
c7550	25	2.89%	0.85%	0.56%	0.06%

*Config. E does not cause any change in RO frequency

IV. CONCLUSION

In the context of ESC 2010, we have presented innovative approaches of designing Trojans and placing them inside a circuit in a way that effectively evades existing protection mechanisms. In particular, we focused on sequential Trojans triggered by a sequence of rare events, and showed that clever design of Trojan trigger/payload circuits can incur ultralow delay/power overhead, thus bypassing side-channel analysis based detection. Through examples, analysis, and results, we have shown that Trojans inserted in foundries can have minimal impact on path delay, thus evading on-chip monitors (e.g. RO) based DFS approaches. Since they trigger under extremely rare conditions, it is also difficult to detect these Trojans in functional testing. The Trojan design and placement approaches presented are effective for both FPGA and ASIC platforms. Further investigation is underway to evaluate the effect of different hardening/DFS mechanisms on these Trojans.

ACKNOWLEDGEMENT

The authors would like to acknowledge financial support from NSF grant CNS 1054744, CNS 0958510 and Xilinx university program for providing the FPGA platforms and tools.

REFERENCES

[1] R.S. Chakraborty et al, "MERO: A statistical approach for hardware Trojan detection", CHES Workshop, 2009.
[2] ESC website: http://www.poly.edu/csaw-embedded.
[3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: identifying and classifying hardware Trojans", IEEE Computer, vol. 43 , no. 10, Oct 2010, pp. 39 – 46.
[4] X. Zhang and M. Tehranipoor, "RON: An on-chip ring oscillator network for hardware Trojan detection", DATE, 2011.
[5] J. Rajendran, V. Jyothi, O. Sinanoglu., and R. Karri, "Design and analysis of ring oscillator based Design-for-Trust technique", VTS, 2011.
[6] Predictable Technology Model. Available: [Online] http://ptm.asu.edu/.
[7] C. Lavin et al, "Using hard macros to reduce FPGA compilation time", FPL, 2010.