Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores

Xuehui Zhang and Mohammad Tehranipoor
Dept. of Electrical & Computer Engineering, University of Connecticut
{xhzhang},{tehrani}@engr.uconn.edu

Abstract

The intellectual property (IP) blocks are designed by hundreds of IP vendors distributed across the world. Such IPs cannot be assumed trusted as hardware Trojans can be maliciously inserted into them and could be used in military, financial and other critical applications. It is extremely difficult to detect Trojans in third-party IPs (3PIPs) simply with conventional verification methods as well as methods developed for detecting Trojans in fabricated ICs. This paper first discusses the difficulties to detect Trojans in 3PIPs. Then a complementary flow is presented to verify the presence of Trojans in 3PIPs by identifying suspicious signals (SS) with formal verification, coverage analysis, removing redundant circuit, sequential automatic test pattern generation (ATPG), and equivalence theorems. Experimental results, shown in the paper for detecting many Trojans inserted into RS232 circuit, demonstrate the efficiency of the flow.

I. Introduction

Due to globalization of the semiconductor design and fabrication process, integrated circuits (ICs) are becoming increasingly vulnerable to malicious activities and alterations. Today's system-on-chips (SOCs) usually contain tens of IP cores (digital and analog) performing various functions. In practice, very seldom IPs are developed by the SOC integrator; in fact most of them are currently being designed offshore by 3PIP vendors. This raises a major concern toward the trustworthiness of 3PIPs. These concerns have been documented in various reports and technical events [1] [2]. IP trust problem is defined as detecting Trojans inserted into 3PIPs. This issue has gained significant attention as a small Trojan in a 3PIP can negatively impact the functionality of the whole chip. Trojans in a 3PIP can be designed to destroy the fabricated ICs, cause erroneous behavior in the field, or provide adversary with an access to secret keys in a secure hardware.

Although IP trust has gained attention in the past few years, currently there is little active research in this area. An approach for pre-synthesis embedding of hardware Trojans horse is introduced in [3]. It demonstrates difficulties for detecting/removing hardware Trojans before synthesis. A four-step method to filter and locate malicious insertions in 3PIPs is proposed in [4] in pre-silicon designs. This method assumes specification circuit presents same time frames and state information as in suspicious circuit to check whether these two circuits produce the same behavior. However, such assumption may be invalid for large industry design.

Given the complexity of the problem, there is no silver bullet available due to the issues mentioned earlier to detect hardware Trojans in 3PIPs. Several concepts such as formal verification, code coverage analysis, and ATPG methods will be employed in this study to achieve high confidence in whether the circuit is Trojan-free or Trojan-inserted. In this paper, a case study is presented to detect Trojans in 3PIPs, based on identification of suspicious signals. Suspicious signals are identified first by coverage analysis with improved test bench. Removing redundant circuit and equivalence theorems will be applied to reduce the number of suspicious signals. Sequential ATPG is used to generate patterns to activate them. This method considers both the characteristics of dormant Trojans and the redundant circuit. Experimental results show the effectiveness of our method.

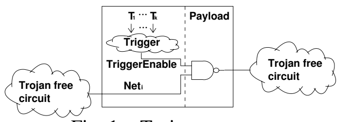


Fig. 1. Trojan structure.

II. Trojan Structure and Formal Verification

A. Trojan Structure

The basic structure of a Trojan in a 3PIP can include two main parts, Trigger and Payload. When trigger is activated, it can change the output of payload which in turn can change the circuit value. Figure 1 shows the basic structure of the Trojan at gate level which is also used in addressing IC trust problem. The trigger inputs (T_1, T_2, ..., T_k) come from various nets in the circuit. The payload taps the original signal (Net_i) from the original (Trojan-free) circuit and the output of the Trigger. Since the trigger is expected to be activated under rare condition, the payload output stays at the same value as Net_i most of the time. However, when trigger is active (i.e. $TriggerEnable$ is '0') the payload output will be different from Net_i; this could result in injecting an erroneous value into the circuit and causing error at the output. Note that Trojans at RTL could be similar to the one shown in Figure 1.

B. Formal Verification and Coverage Analysis

Formal verification is an algorithmic-based approach to logic verification that exhaustively proves functional properties about a design. It contains three types of verification methods that are not commonly used in the traditional verification namely model checking, equivalence checking, and property checking. All functions in the specification are defined as properties. Formal verification uses property checking to check whether the IP satisfies those properties. With property checking, we can explore every corner of the design. In the benchmark RS232, there are two main functionalities in the specification: (1) transmitter; (2) receiver. Figure 2 shows the waveform of the transmitter. Take the $start\ bit$ as an example; with $Rst == 1'b1$, clk positive edge, and $xmitH == 1'b1$, the output signal $Uart\_xmit$ will start to transmit $start\ bit$ "0". This functionality is described using SystemVerilog property shown in Figure 3 and the corresponding assertion is defined simultaneously. The remaining items in the specification are also translated to properties during formal verification. Once all the functionalities in the specification are translated to properties, coverage metrics can help identify suspicious parts in the 3PIP under authentication. Those suspicious parts may be Trojans (or part of Trojans).

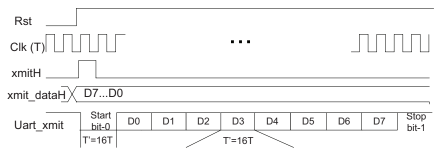


Fig. 2. Transmitter property in the specification.

```
property e1;
   @(posedge uart_clk) disable iff (Rst)
   $rose(xmitH) |− > ##1 (uart_XMIT_dataH==0);
   endproperty

   a1: assert property( e1 );
```

Fig. 3. One of the properties and assertions definition for RS232.

Coverage metrics include Code Coverage and Functional Coverage. It has many different types but only a few of them are helpful for IP trust, namely line, statement, toggle, and finite state machine (FSM) coverage. Toggle coverage reports whether signals switch in gate-level netlist while the other three coverage metrics show which line(s) and statement(s) are executed, and whether states in FSM are reached in RTL code during verification. Figure 4 shows parts of line coverage report during our simulation with RS232. This report shows lines 72 and 74 are not executed, which help us improve the test bench by checking the source code. If the RTL code is easily readable, special patterns that can activate those lines will be added to test bench. Otherwise, random patterns will be added to verify the 3PIP. All the functional requirements can be translated as different types of assertions like Figure 3. Functional coverage checks those assertions to see whether they are successful or not. Table I shows part of assertions coverage report (assertion a1 is defined in Figure 3). The Real Success represents assertion success while Failure/Incomplete denote assertion failure/incomplete. With "0" failure, this property is always satisfied. If all the assertions generated from the specification of the 3PIP are successful and all the coverage metrics such as line, statement, and FSM are 100%, then with a high confidence we can assume that the 3PIP is Trojan-free. Otherwise, the uncovered lines, statements, states in FSM, and signals are considered as suspicious. All the suspicious parts constitute our suspicious list.

III. Techniques to Reduce the Suspicious Signals

It is important to note that a 3PIP source code is largely Trojan free; only few parts may be suspicious. The challenge is to identify suspicious parts that will most likely be part of a Trojan. Figure 5 shows our flow to identify and minimize the suspicious parts. In order to obtain higher coverage for RTL source code, test bench needs to be improved by adding more test patterns, as in Phase 1. Then we synthesize the RTL code to generate the gate-level netlist and remove the redundant circuit. Without redundant circuit and Trojans in the 3PIP, all the signals/components are expected to change their states during the formal verification and 3PIP should function perfectly. Thus, the

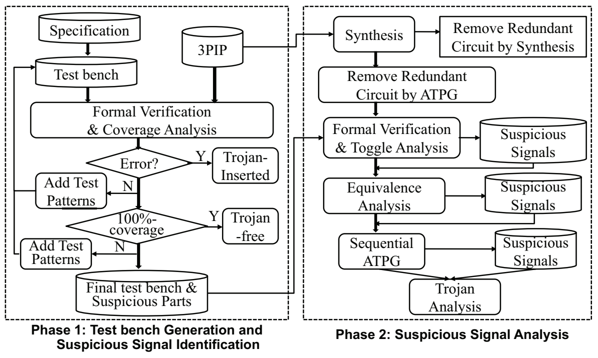
TABLE I

Part of assertion report with RS232.

Assertion	Attempts	Real Success	Failure	Incomplete
test.uart1.uart_checker.a1	500,003	1,953	0	0
test.uart1.uart_checker.a2	1,953	1,952	0	1
...	...	...	...	...

```
Line No        Coverage        Block Type
   69              1              ALWAYS
   71              1            CASEITEM
   72              0            CASEITEM
   73              1            CASEITEM
   74              0                  IF
...                ...                ...
```

Fig. 4. Part of line coverage report.



Specification
3PIP
Test bench
Formal Verification & Coverage Analysis
Error?
Y
N
Trojan-Inserted
Add Test Patterns
100%-coverage
Y
N
Trojan -free
Add Test Patterns
Final test bench & Suspicious Parts
Phase 1: Test bench Generation and Suspicious Signal Identification
Synthesis
Remove Redundant Circuit by Synthesis
Remove Redundant Circuit by ATPG
Formal Verification & Toggle Analysis
Suspicious Signals
Equivalence Analysis
Suspicious Signals
Sequential ATPG
Suspicious Signals
Trojan Analysis
Phase 2: Suspicious Signal Analysis

Fig. 5. The proposed flow for identifying and minimizing suspicious signals.

signals/components that stay stable during toggle coverage analysis are considered suspicious as Trojan circuits do not change their states frequently. Each suspicious signal is then considered as the $TriggerEnable$ of a Trojan.

A. Phase 1: Test Bench Generation and Suspicious Signal Identification

In order to verify the trustworthiness of 3PIPs, we hope the coverage of test bench could be 100%. However, it is very difficult to achieve 100% coverage for every 3PIP, especially those with tens of thousand lines of code. In our flow, the first step is to improve the test bench to obtain a higher code coverage with acceptable simulation run time. With each property in the specification and basic functional patterns, formal verification reports line, statement, and FSM coverage for the RTL code. If one of the assertions is failed even just once during verification, the 3PIP is considered untrusted, containing Trojans or bugs. If all the assertions are successful and the code coverage is 100%, the 3PIP is considered trusted. Otherwise, more test patterns need to be added in the test bench. The basic rule of adding new patterns is to activate the uncovered parts as much as possible. But the verification time will increase as the number of test patterns increases. With the acceptable verification time and certain coverage percentage, both defined by IP buyer, the final test bench will be generated and the RTL source code will be synthesized for further analysis.

B. Phase 2: Suspicious Signals Analysis

1) Remove Redundant Circuit: The redundant circuit must be removed from our suspicious list since they also tend to stay at the same logic value during the verification and input patterns cannot activate them. Removing the redundant circuit involves sequential reasoning, SAT-sweeping, conflict analysis, and data mining. The SAT method integrated in Synopsys Design Compiler (DC) is used in our flow [7]. We also develop another method to remove redundant circuit. Scan chain will be inserted into the gate-level netlist after synthesis for design for testability and ATPG generates patterns for all the stuck-at faults. The untestable stuck-at faults during ATPG is likely to be redundant logic. The reason is that if the stuck-at fault is untestable, the output responses of the faulty circuit will be identical to the output of the fault-free circuit for all possible input patterns. Thus, when ATPG identifies a stuck-at- 1/0 fault as untestable, the faulty net can be replaced by logic 1/0 in the gate-level netlist without scan-chain. All the circuits driving the faulty net will be removed, as well. Figure 6(a) shows the circuit before redundant circuit removal. Stuck-at-0 fault of net F is untestable when generating patterns. Net F will be replaced by 0 and the gate G driving it will be removed from the original circuit as shown in Figure 6(b).

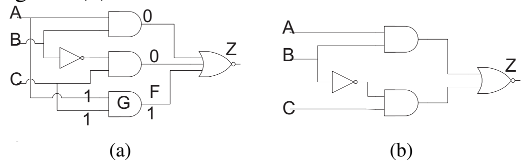


Fig. 6. (a) Before removing redundant circuit with untestable F stuck-at-0 fault and (b) After removing redundant circuit.

After redundant circuit removal, toggle coverage analysis for gate-level netlist without scan chain will identify which signals do not toggle (also called quiet signal) during verification with the test bench generated in Phase 1. These signals will be suspicious and added to our suspicious list. By monitoring these suspicious signals during verification, we can obtain the logic value those signal are stuck at. We try to further reduce the number of suspicious signals in the following.

2) Equivalence Analysis: Fault equivalence theorems are known to reduce the number of faults during ATPG. Similarly, we develop suspicious signal equivalence theorems to reduce the number of suspicious signals.

$Theorem1$: If signal A is the D pin of a flip-flop (FF) while signal B is the Q pin of the same FF, the quiet signal A makes signal B quiet. Thus signal A is considered equal to B, which means if we can find the pattern that can activate A, it will activate B as well. Then signal B will be removed from the suspicious signal list. As the QN port of a FF is the inversion of the Q port, they will stay quiet or switch at the same time. Thus the suspicious signal B would be considered equal to A and should be removed from the suspicious list.

$Theorem2$: If signal A is the output pin of an inverter while signal B is its input, they will stay quiet or switch at the same time. Thus the suspicious signal B would be considered equal to A and should be removed from the suspicious list.

$Theorem3$: One of the input of AND gate A stuck-at-0 will cause the output B stay quiet and one of the input of OR gate C stuck-at-1 will make the output D high all along. Thus, for AND gate, B stuck-at-0 is identical to A stuck-at-0 while for OR gate, D is identical to C stuck-at-1.

3) Sequential ATPG: We use sequential ATPG to generate special patterns to change the value of certain signals during simulation. Stuck-at faults are targeted by the sequential ATPG to generate a sequential pattern to activate the suspicious signals when applied to the 3PIP. If the 3PIP functions perfectly with this pattern, the activated suspicious signals is considered to be part of the original circuit. Otherwise, there must be malicious inclusion in the 3PIP. We acknowledge that sequential ATPG provides limited capability in targeting faults in the circuit and an improved ATPG can be used to collect better results in the future.

TABLE II
ANALYZING IMPACT OF TEST BENCH ON COVERAGE METRICS (BENCHMARK WITH TROJAN 1 IS USED).

Test Bench #	Test Bench 1	Test Bench 2	Test Bench 3	Test Bench 4	Test Bench 5
Test Patterns #	2,000	10,000	20,000	100,000	1,000,000
Verification Time	1 minutes	6 minutes	11 minutes	56 minutes	10 hours
Line Coverage	89.5%	95.2%	98.0%	98.7%	100%
FSM State Coverage	87.5%	87.5%	93.75%	93.75 %	100%
FSM Transition Coverage	86.2%	89.65%	93.1%	96.5%	100%
Path Coverage	77.94 %	80.8%	87.93%	97.34%	100%
Assertion	Successful	Successful	Successful	Successful	Failure

IV. Simulation Results

We apply the flow to RS232 circuit. 9 Trojans from our original design and 10 Trojans from [8] are inserted into the 3PIP. Readers can visit [8] for more details about the specification, structure, and functionality of these Trojans in the 10 RTL benchmarks. Limited by the pages, we won't dissucss these Trojans in details. Totally, there are 19 RS232 benchmarks with one Trojan in each IP. In the following, we first present test bench analysis for the 19 Trojan-inserted benchmarks. Next, the results of redundant circuit removal and reducing the suspicious signals will be presented. Finally, Trojan coverage analysis will be discussed.

A. Impact of Test Bench on Coverage Analysis

All the items in the specification are translated into properties and defined as assertions in test bench. Assertion checkers will verify the correctness of assertions by SystemVerilog. Another most important feature of a test bench is the input patterns. Some test corners need special input patterns. The more input patterns in test bench, the more, for example, lines will be covered during verification. Table II shows five test benches with different test patterns and verification time for various coverage metric reports for the RS232 benchmark with Trojan 1. Generally, the verification time will increase with more test patterns and the code coverage will be higher, as well. For Test Bench 1 to Test Bench 4, all the coverage reports are less than 100% and all the assertions are successful, which states that the Trojan is dormant during the entire verification. But if we add special test patterns in Test Bench 5 and increase the pattern count significantly, which can activate the Trojans inserted in the benchmark, the code coverage could achieve 100% and one of the assertion experiences a failure, which means the Trojan is triggered and RS232 gives an erroneous output. A conclusion that the IP is Trojan-inserted would be made. However, it is not easy to generate a test bench with 100% code coverage for large IPs and the verification time will be extremely long. This phase of the flow can help improve quality of the test bench. Given the time-coverage trade off, a test bench is selected for further analysis. Thus, here, we will select Test Bench 4 to verify this and the remaining benchmarks.

B. Reducing the Suspicious Signals

All the 19 benchmarks with different Trojans are synthesized to generate the gate-level netlist. Removing redundant circuit is

TABLE III

SIMULATION RESULTS WITH 10 BENCHMARKS.

Benchmark	Total Area Unit	Trojan Area Unit	Area Overhead	Step 1: Number of SS after Redundant Circuit Removal with Synthesis	Step 2:Number of SS after Redundant Circuit Removal with ATPG	Step 3: Number of SS after Equivalence Analysis	Step 4: Number of SS after Sequential ATPG	*SS-Overlap-Trojan*
Benchmark with Trojan 1	15995.2	1789.1	11.18%	22	20	17	12	100%
Benchmark with Trojan 2	16824.7	1254.6	20.35%	17	16	3	Trojan is identified	100%
Benchmark with Trojan 3	15005.2	1573.1	10.48%	20	15	15	10	97.3%
Benchmark with Trojan 6	14364.3	166.2	1.15 %	1	1	1	Trojan is identified	100 %
Benchmark with Trojan 8	14224.3	20.2	1.15 %	1	Trojan is removed	-	-	100%
RS232-TR04C13PI0	21394.9	342.5	1.6%	8	3	3	3	100%
RS232-TR0CS02PI0	34379.3	8700.0	25.3%	59	55	39	39	67.7%
RS232-TR0ES12PI0	21117.9	422.3	2.09%	8	1	1	1	100%
RS232-TR0FS02PI0	34217.0	8552.6	25.0%	30	28	20	20	73.3%
RS232-TR30S0API0	23778.7	2945.8	12.4%	22	20	13	13	93.6%

done during the synthesis process with special constrains using Design Compiler. The simulation results are shown in Table III (limited by the pages, only ten of them are shown in the table). The second and third columns in the table show the total area of each benchmark and the area overhead of each Trojan after generating the final layout. From this table, we can see that Trojans are composed of different sizes, gates, and structures as well as different triggers and payloads as mentioned earlier. The smallest Trojan has only 1.15% area overhead. The percentage of Trojan area covered by suspicious signals *SS-Overlap-Trojan* is obtained by *SS-Overlap-Trojan*=$\frac{N_{SS}}{N_{TS}}$ where N_{SS} is the number of suspicious signals and N_{TS} is the number of Trojan signals. The results in Table III show that *SS-Overlap-Trojan* is between 67.7% and 100% as shown in column 9. If all the suspicious signals are part of Trojan, the *SS-Overlap-Trojan* would be 100%. This indicates that the number of signals in the final suspicious list fully overlapped with those from Trojan. This is an indicator of how successful the flow is in identifying Trojan signals. In addition, if the Trojan is removed or detected by sequential ATPG, the *SS-Overlap-Trojan* would be 100%, as well.

Test Bench 4 is used to verify the gate-level netlist and toggle coverage analysis reports which signals in each Trojan-inserted circuit are not covered by the simulation with all the successful assertions. Those quiet signals are identified as suspicious. The number of suspicious signals of each benchmark is shown in the fifth column in Table III. The larger the Trojan is, the more suspicious signals it has. On the other hand, the suspicious signals stuck-at values are monitored by verification. All stuck-at-faults are simulated by ATPG tool with scan chain in the netlist. If the fault is untestable, the suspicious circuit is a redundant circuit and will be removed from the original gate level netlist, in addition to the gates that drive the net. The number of suspicious nets after redundant circuit removal is shown in the sixth column in Table III. We can see that the suspicious nets of benchmarks with Trojan 8 (also Trojan 9, not shown in the table) are zero, which means if the redundant circuit are removed in the two benchmarks, the benchmarks will be Trojan-free. The reason that redundant circuit removal can distinguish Trojans is that some Trojans are designed without payload and have no impact on circuit functionality. Thus, we can conclude that such Trojans can be removed by redundant circuit removal.

The remaining suspicious nets of each benchmark, are needed to be processed by equivalence analysis and sequential ATPG. The seventh and eighth columns in Table III show the number of suspicious signals after the two steps. We can see that equivalence analysis can reduce a large number of suspicious signals and sequential ATPG can be effective as well. For benchmarks with Trojan 2 (also, Trojan 6 not shown in the table), the sequential ATPG can generate sequential pattens for the stuck-at faults in the suspicious signal. The sequential test patterns improve the test bench and increase its coverage percentage. Even though the coverage percentage is not 100%, some assertions experience failure during simulation. Thus, we conclude that the benchmark with Trojan 2 and Trojan 6 is Trojan-inserted.

We have implemented our flow on 10 of trust benchmarks from Trust-Hub [8] and 5 of them are reported in rows 7 to 11 in Table III show that the presented flow in this paper can effectively reduce the total number of suspicious signals. In addition, as shown in column 9, there is a good overlap between the number of suspicious signals and actual Trojan signals inserted into each benchmark. However, we experience low *SS-Overlap-Trojan* with a couple of benchmarks, such as RS232-TR0CS02PI0, since part of this Trojan was activated during simulation.

V. ACKNOWLEDGEMENT

This work was supported in part by National Science Foundation (NSF) grants CNS-0844995 and CNS-1059390.

VI. CONCLUSION AND FUTURE WORK

We have presented a study to verify trustworthiness of 3PIPs, involving formal verification, coverage analysis, redundant circuit removal, sequential ATPG, and equivalence theorems. In the future work, we plan to improve the proposed methods and verify it through a large number of Trojans inserted into RS232 circuit especially Trojans that hide their activity well.

REFERENCES

[1] "Report of the Defense Science Board Task Force on High Performance Microchip Supply," Defense Science Board, US DoD, http://www.acq.osd.mil/dsb/reports/2005-02-HPMSi_Report_Final.pdf, Feb, 2005.

[2] "IEEE Int. Symposium on Hardware-Oriented Security and Trust (HOST)," http://hostsymposium.org.

[3] Y. Alkabani and F. Koushanfar, "Extended Abstract: Designer's Hardware Trojan Horse," in Proc. *IEEE Int Workshop Hardware-Oriented Security and Trust (HOST 08)*, pp. 82-83, 2008.

[4] M, Banga and M, Hsiao, "Trusted RTL: Trojan detection methodology in pre-silicion designs," in Proc. *IEEE Int Symposium on Hardware-Oriented Security and Trust (HOST 10)*, pp. 56-59 2010.

[5] I. Ugarte and P. Sanchez, "Formal Meaning of Coverage Metrics in Simulation-Based Hardware Design Verification," IEEE, 2005.

[6] P. Andrew, Functional Verification Coverage Measurement and Analysis, 2004.

[7] Synopsys Inc. Design Compiler Reference Version 2006.06, 2006.

[8] http://trust-hub.org/resources/benchmarks, 2010.