Chapter 2
Hardware Trojan Detection: Untrusted Third-Party IP Cores

In general, third-party IP (3PIP) cores fall into one of three categories: soft, firm, and hard, depending on their format when they are supplied. Soft IP cores are described using VHDL or Verilog and are the most flexible and popular cores in practice. Firm cores are described and synthesized for specific libraries, while hard IP cores are described at the physical level and are supplied as layout or GDSII file. Since soft IP cores are the most widely used, detecting hardware Trojans in 3PIP, defined as IP trust, has gained significant attention in the recent years.

Hardware Trojans can be inserted into 3PIPs by IP vendors during IP design to steal security information/data from other IPs in the system. Detection of such Trojans is extremely difficult since there is no known golden model for 3PIPs as IP vendors usually provide specification and source code, both of which may contain Trojans. The conventional side-channel techniques for IC trust are not applicable to IP trust. When a Trojan exists in an IP core, all the fabricated ICs will contain Trojans. The only trusted component would be the specification from the SOC designer which defines the function, primary input and output, and other information about the 3PIP that they intend to use in their systems. A Trojan can be very well hidden during the normal functional operation of the 3PIP supplied as register transfer level (RTL) code. A large industrial-strength IP core can include thousands of lines of code. Identifying the few lines of RTL code in a soft IP core that represent a Trojan is an extremely challenging task.

A case study that tries to address the IP trust problem is presented in [1]. This chapter will present this case study in details. Several concepts such as formal verification, code coverage analysis, and ATPG methods are employed in this case study to achieve high confidence, whether the circuit is Trojan-free or Trojan-inserted. It is important to note that a 3PIP source code is largely Trojan free; only a few parts may be suspicious. The challenge is to identify the suspicious parts that are most likely to be part of a Trojan. Suspicious signals are identified first by coverage analysis. Removing redundant circuit and equivalence theorems reduces

M. Tehranipoor et al., *Integrated Circuit Authentication: Hardware Trojans and Counterfeit Detection*, DOI 10.1007/978-3-319-00816-5_2,

the number of suspicious signals. Sequential ATPG is used to generate patterns to activate these suspicious signals. This method considers both the characteristics of dormant Trojans and the redundant circuit.

2.1 A Case Study for Hardware Trojan Detection in Third-Party Digital IP Cores

2.1.1 Formal Verification and Coverage Analysis

One of the important concepts in this case study is formal verification, an algorithmic-based approach to logic verification that exhaustively proves the functional properties of a design [2]. 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. The specific corner cases in the test suite as they monitor particular objects in a 3PIP could also be represented by properties, such as worry cases, inter-block interfaces, and complex RTL structures. They can be represented as properties wherever the protocols may be misused, assumptions violated, or design intent incorrectly implemented. Formal verification uses property checking to check whether the IP satisfies those properties. With property checking, every corner of the design can be explored. For example, in benchmark RS232, there are two main functionalities in the specification: (1) transmitter, and (2) receiver. Figure 2.1 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 the SystemVerilog property shown in Fig. 2.2, 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).

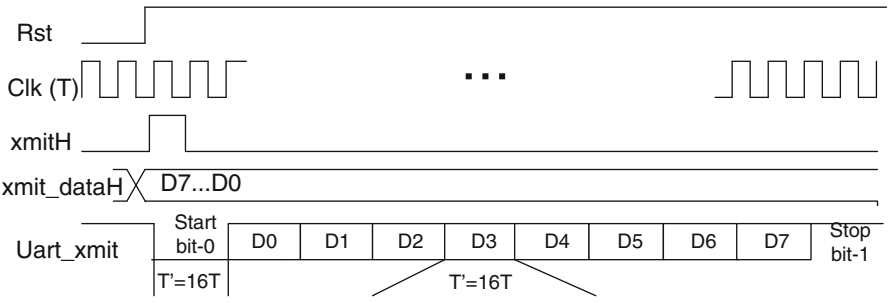

Fig. 2.1 Transmitter property in the specification stage

```
01:   property e1;
02:     @(posedge uart_clk) disable iff (Rst)
03:     $rose(xmitH) |− > ##1 (uart_XMIT_dataH==0);
04:     endproperty
05:
06:     a1: assert property( e1 );
```

Fig. 2.2 One of the properties and assertions definitions for RS232

01: Line No	Coverage	Block Type
02: 69	1	ALWAYS
03: 70	1	CASEITEM
04: 71	1	CASEITEM
05: 72	0	CASEITEM
06: 73	1	CASEITEM
07: 74	0	CASEITEM
08: 82	1	ALWAYS
09: 82.1	1	IF
... ...	...	...

Fig. 2.3 Part of the line coverage report

Coverage metrics include code coverage and functional coverage. Code coverage analysis is a metric that evaluates the effectiveness of a test bench in exercising the design [3, 4]. There are many different types of code coverage analysis, 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 the 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 2.3 shows parts of line coverage report during the simulation with RS232. This report shows that lines 72 and 74 are not executed, which helps 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 the test bench. Otherwise, random patterns will be added to verify the 3PIP.

Functional coverage is the determination of how much functionality of the design has been exercised by the verification environment. The functional requirements are imposed on both the design inputs and outputs and on their interrelationships by the design specifications from SOC designer (i.e. IP buyers). All the functional requirements can be translated as different types of assertion, as in Fig. 2.2. Functional coverage checks those assertions to see whether they are successful or not. Table 2.1 shows part of the assertions coverage report (Assertion a1 is defined in Fig. 2.2). The number of Attempts in the table means there are 500, 003 positive edge clocks during the simulation time when the tool tries to check the assertion.

Table 2.1 Part of the 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
...	...	...	...	...

The Real Success represents the assertion success rate while Failure/Incomplete denote the frequency of assertion failure/incomplete. When there are zero Failures, 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 it can be assumed with high confidence that the 3PIP is Trojan-free. Otherwise, the uncovered lines, statements, states in FSM, and signals are considered suspicious. All the suspicious parts constitute the suspicious list.

2.1.2 Techniques for Suspicious Signals Reduction

Based on the formal verification and coverage metric, a flow is proposed to verify the trustworthiness of 3PIP in [1]. The basic idea of the proposed solution is that without redundant circuit and Trojans in a 3PIP, all the signals/ components are expected to change their states during verification and 3PIP should function perfectly. Thus, the 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 *TriggercEnablex*. Figure 2.4 shows the flow to identify and minimize the suspicious parts, including test pattern generation, suspicious signal identification, and suspicious signal analysis. Each step in the figure will be discussed in detail in the following.

2.1.2.1 Phase 1: Test Bench Generation and Suspicious Signal Identification

In order to verify the trustworthiness of 3PIPs, 100 % coverage of the test bench is best. However, it is very difficult to achieve 100 % coverage for every 3PIP, especially those with tens of thousands of lines of code. In the flow, the first step is to improve the test bench to obtain a higher code coverage with an acceptable simulation run time. With each property in the specification and basic functional test vectors, formal verification reports line, statement, and FSM coverage for the RTL code. If one of the assertions fails even once during verification, the 3PIP is considered untrustworthy, containing Trojans or bugs. If all the assertions are successful and the code coverage is 100 %, the 3PIP can be trusted. If at least one assertion fails or the code coverage is less than 100 %, more test vectors need to

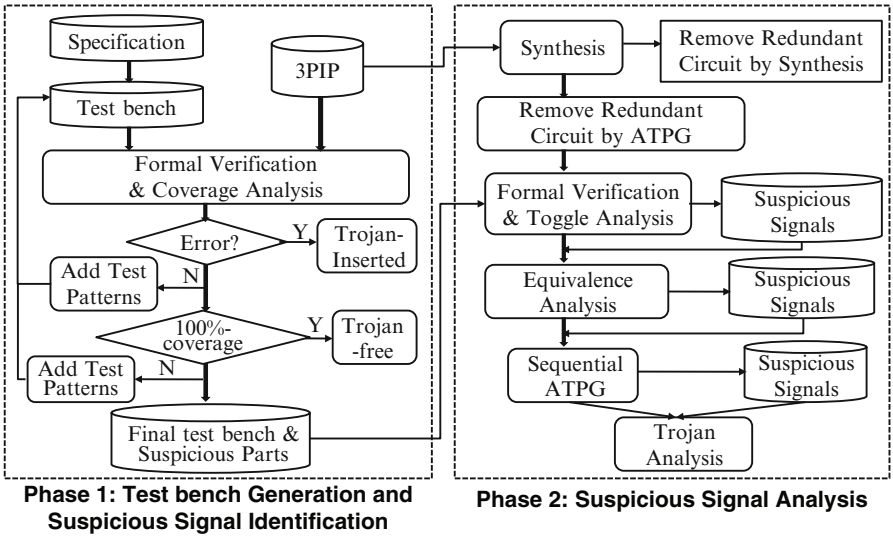

Fig. 2.4 The proposed flow for identifying and minimizing suspicious signals

be added to the test bench. The basic purpose of adding new vectors is to activate the uncovered parts as much as possible. But the verification time will increase as the number of test vectors increases. With the acceptable verification time and certain coverage percentage, both defined by the IP buyer, the final test bench will be generated and the RTL source code will be synthesized for further analysis.

2.1.2.2 Phase 2: Suspicious Signals Analysis

Redundant Circuit Removal (RCR): Redundant circuits must be removed from the suspicious list since they also tend to stay at the same logic value during verification, and input patterns cannot activate them. Removing a redundant circuit involves sequential reasoning, SAT-sweeping, conflict analysis, and data mining. The SAT method integrated in the Synopsys Design Compiler (DC) is used in this flow.

Another method to remove redundant circuits is developed in [1]. Scan chains are inserted into the gate-level netlist after synthesis for design testability, and ATPG generates patterns for all the stuck-at faults. The untestable stuck-at faults during ATPG are 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 2.5a shows the circuit before redundant circuit removal.

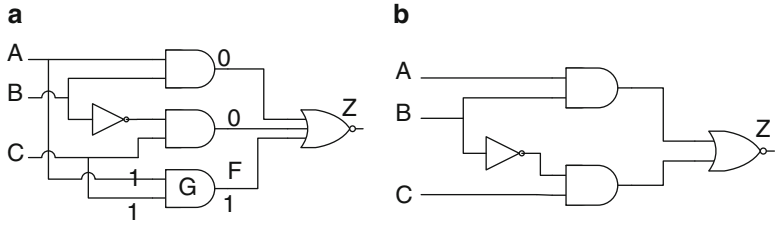

Fig. 2.5 (**a**) Before removing the redundant circuit with untestable F stuck-at-0 fault and (**b**) After removing the redundant circuit

The 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 Fig. 2.5b.

After redundant circuit removal, toggle coverage analysis for the 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 considered suspicious and added to the suspicious list. By monitoring these suspicious signals during verification, the authors obtain the logic value those signal are stuck at.

Equivalence Analysis: Fault equivalence theorems are known to reduce the number of faults during ATPG [5]. Similarly, the authors develop suspicious signal equivalence theorems to reduce the number of suspicious signals in [1].

Theorem 1. *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 the pattern that can activate A is found, 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.

Theorem 2. *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.*

Theorem 3. *One of the input of AND gate A stuck-at-0 will cause the output B to 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.*

Sequential ATPG: After reducing the number of suspicious signals by applying the above equivalence theorems, the authors use sequential ATPG to generate special patterns to change the value of certain signals during simulation in [1]. 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 are considered part of the original circuit. Otherwise, there must be malicious inclusion in the 3PIP.

2.1.3 Simulation Results

The flow is applied to the RS232 circuit. 9 Trojans from the original design and 10 Trojans from [6] are inserted into the 3PIP. In total, there are 19 RS232 benchmarks with one Trojan in each IP. The following presents the simulation setup and test bench analysis for the 19 Trojan-inserted benchmarks. Next, the results of redundant circuit removal and the reduction of suspicious signals will be presented. Finally, Trojan coverage analysis will be discussed.

2.1.3.1 Benchmark Setup

Currently, there are over 80 benchmarks with different Trojans in the Trust-Hub [6], from which 51 benchmarks are at the RT Level. Readers can visit [6] for more details about the specification, structure, and functionality of these Trojans in the 10 RTL benchmarks. However, the other 9 Trojans are briefly described in the following:

*Trojan*1: The trigger of Trojan 1 is a special input sequence $8'ha6 - 8'h75 - 8'hc0 - 8'hff$. The payload changes the FSM in the transmitter of RS232 from state $Start$ to $Stop$, which means that once the Trojan is triggered, RS232 will stop transmitting data ($outputdata = 8'h0$). Since the trigger of the Trojan is a sequence of four special inputs, the probability of detecting the Trojan during verification is $1/2^{32}$. If the baud rate is 2,400 and RS232 transmits 240 words in one second, it will take 207.2 days to activate the Trojan and detect the error. In other words, it would be practically impossible to detect it by conventional verification. When this Trojan is inserted into RS232, an FSM is used to describe the Trojan input sequence. A three-bit variable state represents the FSM.

*Trojan*2: This Trojan only adds four lines to the original RTL code. If the transmitting word is odd and the receiving word is $8'haa$, RS232 will stop receiving words. This Trojan is less complex compared to Trojan 1, however, it provides opportunities to demonstrate the effectiveness of each step of the proposed flow.

*Trojan*3: The trigger of Trojan 3 is the same as that of Trojan 1, but the payload is different. Trojan 1 changes the state machine while Trojan 3 changes the shift process. The eighth bit of the transmitting word will be replaced by a Trojan bit during transmission. The Trojan bit could be authentication information, the special key to enable the system, or other important information.

*Trojan*4: Trojan 4 is designed to act like a time bomb. A counter is inserted into RS232 to count the number of words that have been sent out. After sending $10'h3ff$ words, the Trojan will be activated. The sixth bit of the transmitting word will be replaced by a Trojan bit.

Table 2.2 Analyzing the impact of the test bench on coverage metrics (a 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 min	6 min	11 min	56 min	10 h
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

*Trojan*5: After $24'hffffff$ positive edge clock, this Trojan's enable signal will become high. The sixth bit of the transmitting word will be replaced by a Trojan bit.

*Trojan*6: If RS232 receives "0" when the system is reset, the Trojan will be activated. The eighth bit of the transmitting word will be replaced by a Trojan bit.

*Trojan*7: When the transmitter sends a word $8'h01$ and the receiver receives a word $8'hef$ at the same time, the Trojan will be activated. A Trojan bit will replace the first bit of the transmitting word.

*Trojan*8 & 9: These Trojans do not tamper the original function of RS232 but add extra one stage (Trojan 8) and three stage (Trojan 9) ring oscillator to the RTL, which will increase the temperature of the chip quickly if they get activated.

2.1.3.2 Impact of Test Bench on Coverage Analysis

All the items in the specification are translated into properties and defined as assertions in the test bench. Assertion checkers will verify the correctness of assertions by SystemVerilog. Another important feature of a test bench is the input patterns. Some test corners need special input patterns. The more input patterns in the test bench, the more, for example, lines will be covered during verification. Table 2.2 shows five test benches with different test patterns and verification times 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 indicates that the Trojan is dormant during the entire verification. The special test patterns added in Test Bench 5 increase the pattern count significantly and can activate the Trojans inserted in the benchmark. 100 % code coverage could be achieved with these additional test patterns. If one of the assertion experiences a failure, it signifies Trojan activation and the RS232 will give an erroneous output. One can conclude that the IP is Trojan-inserted. 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 the quality of the test bench. Given the time-coverage trade off, Test Bench 4 is selected for further analysis.

2.1.3.3 Reducing the Suspicious Signals

All the 19 benchmarks with different Trojans are synthesized to generate the gate-level netlist. The removal of redundant circuits is done during the synthesis process with special constrains using the Design Compiler. The simulation results are shown in Table 2.3. The second column in the table shows the area overhead of each Trojan after generating the final layout. As the table shows, Trojans are composed of different sizes, gates, and Structures, as well as different triggers and payloads as previously mentioned. The smallest Trojan has only 1.15 % area overhead. The percentage of Trojan area covered by suspicious signals *SS-Overlap-Trojan* is obtained by $SS\text{-}Overlap\text{-}Trojan = \frac{N_{SS}}{N_{TS}}$ where N_{SS} is the number of suspicious signals and N_{TS} is the number of Trojan signals. The results in Table 2.3 show that *SS-Overlap-Trojan* is between 67.7 % and 100 %, as shown in seventh column. If all the suspicious signals are part of the 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 at identifying Trojan signals. In addition, if the Trojan is removed or detected by sequential ATPG, the *SS-Overlap-Trojan* would also be 100 %.

Test Bench 4 is used to verify using 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 third column of Table 2.3. Different benchmarks have a different number of suspicious signals based on the size of its Trojans. 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 the 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 fourth column of Table 2.3. As can be seen in the table, the suspicious nets of benchmarks with Trojan 8 and Trojan 9 are zero, which means that if the redundant circuits 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 it can be concluded 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 fifth and sixth columns in Table 2.3 show the number of suspicious signals after the first two steps. It can be concluded that equivalence analysis can reduce a large number of suspicious signals, and sequential ATPG can be effective as well. For benchmarks with Trojan 2

Table 2.3 Suspicious signal analysis

Benchmark (RS232)	Trojan area overhead (%)	Step 1: Number of SS after RCR with Synthesis	Step 2: Number of SS after RCR with ATPG	Step 3: Number of SS after equivalence analysis	Step 4: Number of SS after sequential ATPG	SS-Overlap-Trojan (%)
With Trojan 1	11.18	22	20	17	12	100
With Trojan 2	20.35	17	16	3	Trojan is identified	100
With Trojan 3	10.48	20	15	15	10	97.3
With Trojan 4	20.35	3	3	3	2	87.6
With Trojan 5	4.59	9	8	8	7	100
With Trojan 6	1.15	1	1	1	Trojan is identified	100
With Trojan 7	3.79	3	3	3	2	100
With Trojan 8	1.15	1	Trojan is removed	-	-	100
With Trojan 9	3.79	3	Trojan is removed	-	-	100
TR04C13PI0	1.6	8	3	3	3	100
TR06C13PI0	1.8	9	3	3	3	100
TR0AS10PI0	2.09	8	1	1	1	100
TR0CS02PI0	25.3	59	55	39	39	67.7
TR0ES12PI0	2.09	8	1	1	1	100
TR0FS02PI0	25.0	30	28	20	20	73.3
TR2AS0API0	11.9	19	18	11	11	100
TR2ES0API0	12.0	20	18	11	11	100
TR30S0API0	12.4	22	20	13	13	93.6
TR30S0API1	12.3	25	22	14	14	87.3

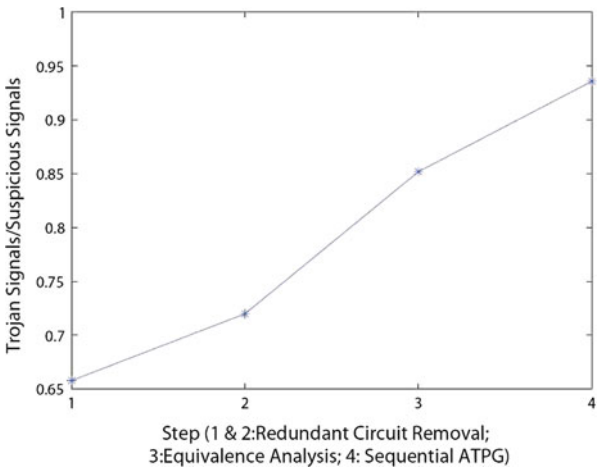

Fig. 2.6 Average Trojan signals/Suspicious signals in 19 benchmarks

and Trojan 6, the sequential ATPG can generate sequential patterns 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, the benchmarks with Trojan 2 and Trojan 6 are identified as Trojan-inserted.

The flow is implemented on 10 trust benchmarks from the Trust-Hub [6] and the results reported in rows 11–20 in Table 2.3 show that the presented flow can effectively reduce the total number of suspicious signals. In addition, as shown in seventh column, there is a good overlap between the number of suspicious signals and the actual Trojan signals inserted into each benchmark. However, some benchmarks experience low *SS-Overlap-Trojan*, such as RS232-TR0CS02PI0, since only part of this Trojan was activated during simulation.

2.1.3.4 Trojan Coverage Analysis

In the suspicious list, not all of signals are a result of Trojans. However, the $TriggerEnable$ signal must be in the suspicious list if the IP contains a Trojan. Once one net is identified as part a Trojan, it can be concluded that the 3PIP is Trojan-inserted. All the gates driving this net are considered to be Trojan gates. Figure 2.6 shows that the percentage of Trojan signals in the suspicious list increases significantly with the flow. As the authors apply different steps (step 1–4) to the benchmarks, 72 %, on average, of the suspicious signals are of the result of Trojans after redundant circuit removal with synthesis and ATPG in the 19 benchmarks. However, the percentage increases to 85.2 % when equivalence analysis is done and 93.6 % of signals in the suspicious signal list come from Trojans after sequential ATPG is applied to these benchmarks.

2.2 Summary

In this chapter, a case study is presented to verify the trustworthiness of 3PIPs, involving formal verification, coverage analysis, redundant circuit removal, sequential ATPG, and equivalence theorems. The code coverage generates the suspicious signals list. Redundant circuit are removed to reduce the number of suspicious signals. Equivalence theorems are developed for the same purpose. Sequential ATPG is used to activate these suspicious signals and some Trojans will be detected. However, more work is needed to get 100 % hardware Trojan detection rates in 3PIPs.

References

1. X. Zhang and M. Tehranipoor, "Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores," in Int. IEEE Hardware-Oriented Security and Trust (HOST), 2011.
2. A. J. Hu, "Formal Hardware Verification with BDDs: An Introduction," *IEEE*, 1997.
3. Synopsys, "The Synopsys Verification Avenue Technical Bulletin", Vo1.4,issue 4, December 2004.
4. I. Ugarte and P. Sanchez, "Formal Meaning of Coverage Metrics in Simulation-Based Hardware Design Verification," IEEE, 2005.
5. M. Bushnell and A. Vishwani, "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits," 2000.
6. http://trust-hub.org/resources/benchmarks.

http://www.springer.com/978-3-319-00815-8

Integrated Circuit Authentication
Hardware Trojans and Counterfeit Detection
Tehranipoor, M.; Salmani, H.; Zhang, X.
2014, XVI, 222 p. 120 illus., 65 illus. in color., Hardcover
ISBN: 978-3-319-00815-8