Hardware Trojan Detection at Behavioral Level Using Inline Assertions and Verification Using UVM

Aki Vamsi Krishna and E. Prabhu

Abstract Recently, hardware Trojan (HT) is posing a significant challenge to the integrated circuit (IC) industry and has inspired various improvements in the Trojan identification plans. This research study presents the inline assertions for the detection of hardware Trojan at the behavioral level of a system on chip (SoC). In the proposed RTL design, a modified circuit design flow is suggested to incorporate inline assertions into a SoC. Flexible inline assertions are developed in the RTL block within the design module. The router IP design and inline assertions are synthesized and implemented in Xilinx Vivado and Aldec Rivera Pro using Verilog HDL. The universal verification methodology (UVM) is also used to verify the proposed design with the different test case scenarios. The functional coverage and code coverage are analyzed in Aldec Rivera Pro. Parameters such as power and area are analyzed in the Synopsys design compiler (DC).

Keywords Hardware Trojan · Behavioral level · Inline assertions · Verilog HDL · UVM · Xilinx Vivado · Aldec Rivera pro · Synopsys design compiler

1 Introduction

With the recent increase in the IC production and the cost of profound sub-micrometer innovation, many IC design houses are currently importing some modules and outsourcing production to the third party (3P), which is considered as a typical practice in the chip improvement cycle. The 3P IP cores and design automation tools are extensively used to improve the circuits. Therefore, the integrity of manufactured product could be undermined. The likelihood that an IC will be susceptible to attack by HT has been increased. A chip, or otherwise a circuit, can be hacked and attacked, resulting in certain modifications if an attacker accesses specific stages of the IC design flow, as shown in Fig. 1 [1].

A. V. Krishna · E. Prabhu (✉)
Department of Electronics and Communication Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: e_prabhu@cb.amrita.edu

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022

S. Smys et al. (eds.), *Inventive Computation and Information Technologies*, Lecture Notes in Networks and Systems 336, https://doi.org/10.1007/978-981-16-6723-7_2

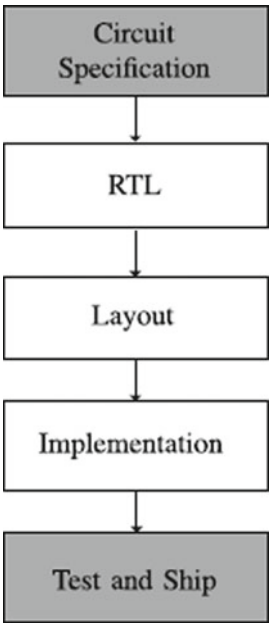

Fig. 1 IC design life cycle

A HT can be described as a malicious change or consideration to IC that changes its functions or cause it to perform an extra malicious function [3]. These malicious incorporations or modifications are mostly customized to activate under a specific set of conditions formulated by an attacker and are extremely difficult to identify or detect at the dormant state. Many different industries, for example, military, telecommunications, and business applications are continuously focused on expanding the challenges faced by malicious circuits that are included for the design [13]. The method proposed to overcome this vulnerability is to utilize HT detection methods at different levels of chip IC design process [4].

The proposed research work is concentrated on one of the level of chip IC design flow that is the RTL block. RTL is a hardware description language (HDL), which is basically the detailed logical implementation of the entire IC and detailed design or circuit specifications that are converted into Verilog or VHDL language. An RTL modeling style is a synthesizable coding style, which could be a data flow model or a behavioral model. The data flow model is a signal that is assigned through the data manipulating equations; all such assignments are concurrent in nature. The behavioral level is the highest level of design description that contains functions, procedural statements, and design modeling. RTL style of coding is widely used in synchronous designs, which involve both combinational and sequential designs. RTL basically represents the data flow between combinational clouds and registers.

This paper focuses on the designing and implementation of router IP protocol using Verilog HDL language. Also, the proposed research work introduces the application of inline assertions in order to detect HT at the behavioral level of the design

or system. The primary goal is to provide a dedicated RTL block that can be a module design with inline assertions in order to detect the HTs during runtime. The primary objectives of this research work are as follows. (1) The inline assertions are proposed within a design module dedicated to finding HT detection. (2) The proposed inline assertions technique is applied on industrial protocols like router IP which contains (FIFO, FSM, synchronizer, register). (3) The proposed router IP design and inline assertions have very little power and area. (4) With different test cases, the proposed design is verified with the latest methodology called UVM.

The paper shows some related works on this concept in Sects. 2 and 3 shows a brief introduction on router IP design and proposed detection method in Sect. 4, correspondingly; Sect. 5 illustrates the UVM test bench. Section 6 shows the simulation outputs, power, area values, and coverage report. Section 7 provides conclusion.

2 Background and Related Works

The authors stated that a better understanding of what HT may resemble and what influence they could have on an IC are necessary. HT is a malicious module, which is introduced inside the IC during the fabrication or design process. Further, they present eight particular attack procedures by utilizing RTL HTs to bargain the safety of an alpha encryption module [2].

HT is made out of a few gates and tries to change the functionality of the chip. These types of HTs are difficult to detect with offline HT detection methods, for example, digital systems tests and side-channel analysis techniques, and authors proposed methodology focuses on an online method for quickly HTs at the runtime [3]. The authors survey made on a different type of HTs present in IC, different Trojans insertions at various stages of IC, and different techniques for detection of the HTs [4].

The author proposed a secured netlist generation using obfuscation techniques without modifying the functionality of circuits with reduced area and power [5]. The authors provide a technique that involves inserting observation sites into the circuit to capture the most difficult-to-observe faults, which works in conjunction with off-chip and on-chip structural testing to provide greater coverage [6].

The authors have implemented the hardware router IP design with different protocol versions like IPv4 and IPv6 results in higher switching speeds of per packet routing for two protocols by applying VLSI architecture techniques [7]. The time of arrival is calculated by using two different time analysis STAs and statistical STAs. The implementation was carried out in ISCAS-89 benchmark circuits with results of the time improvement [8]. Proposed an efficient activity estimator which is fast and accurate and a survey paper on switching activity estimations techniques, power estimation was done in Synopsys DC tool gave a reduction in power for the circuits [9].

The authors proposed a technique for the automated checker generation of the PSL properties for the verification [10]. In this paper, assertions checkers are used

for security of the processors designs during memory instructions, and also survey made on PSL2HDL tool and code coverage techniques to detect malicious [11]. Ngo et al. proposed a built-in assertion checkers that integrate into the design of general useful designs to identify Trojan during runtime in which ACs selection happens pre-synthesis [12]. The author proposed a reconfigurable assertion checker to detection of the HTs at the SoC and demonstrated the mapping of ACs into RAC [13].

Demonstrating the UVM methodology for design verification, explain the UVM test bench hierarchy, registration of factory, components, TLM, mailbox, and call-backs. Different approaches are demonstrated for developing a test strategy and test cases for design verification [14]. This paper describes the implementation of various types of verification mechanisms that can be used with the UVM-based verification environment to improve the ability to protocol verify, hidden bugs, functional checking of design under verification (DUV) [15].

3 Router IP Design

A router IP protocol that forward data packets between computer networks. Packet header contains address based on that it drives the incoming packet to an output channel. At the same time, three parallel connections will support in the router. Router top-level block as shown in Fig. 2, which shows inputs and outputs signals from source network to three client networks.

Router interface of input and out signals defined the functionality of each signal is shown in Table 1. Router design features contain packet routing, parity checking, reset, header, payload, and parity. Packet routing is driven from the input port and

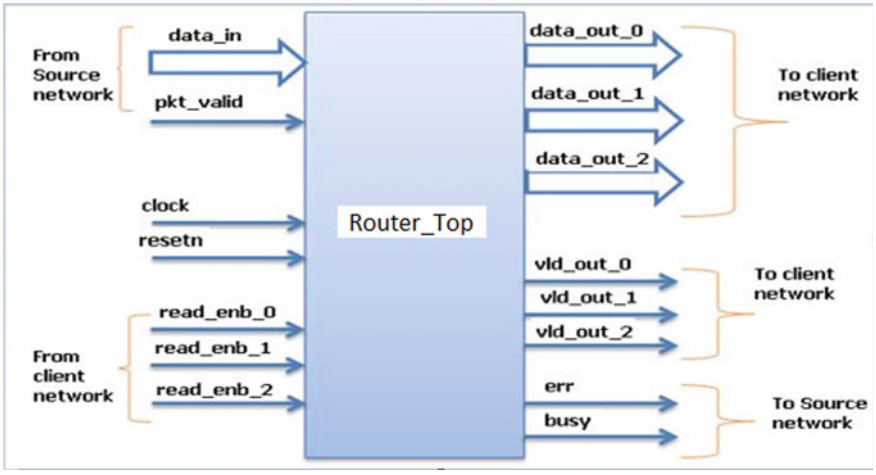

Fig. 2 Block diagram of router top level

Table 1 Router interface

Input/output	Functionality
Clock	Active high clocking event
pkt_valid	Pkt_valid is an active high input signal that detects an arrival of a new packet from a source network
Resetn	Active low synchronous reset
data_in	Eight-bit input data bus that transmits the packet from source network to router
read_enb_0	Active high input signal for reading the packet through output data bus data_out_0
read_enb_1	Active high input signal for reading the packet through output data bus data_out_1
read_enb_2	Active high input signal for reading the packet through output data bus data_out_2
data_out_0	Eight-bit output data bus that transmits the packet from the router to destination client network 1
data_out_1	Eight-bit output data bus that transmits the packet from the router to destination client network 2
data_out_2	Eight-bit output data bus that transmits the packet from the router to destination client network 3
vld_out_0	Active high signal that detects that a valid byte is available for destination client network 1
vld_out_0	Active high signal that detects that a valid byte is available for destination client network 2
vld_out_0	Active high signal that detects that a valid byte is available for destination client network 3
Busy	Active high signal that detects a busy state for the router that stops accepting any new byte
Err	Active high signal that detects the mismatch between packet parity and internal parity

is routed to any output port, based on the address of the destination network. Parity checking is an error detection being transmitted between server and client. This technique guarantees that the data transmitted by the server network is received by the client network without getting corrupted. Reset is an active low-synchronous input that resets the router, and three FIFO are made empty, and the valid out signals go low indicating that no valid packet on the output data bus. Packet format consists three parts parity, header, and payload; each packet has eight bits width and 1 byte to 63 bytes of the pay load length as shown in Fig. 3. Header destination address has 2 bits of packet, and length has 6 bits of the data. Payload was the data; it is in format of bytes. Parity is used as security to verify of the packet.

The top-level block above as shown in Fig. 2 consists of 6 sub-blocks as shown in Fig. 4, as followed three FIFO, synchronizer, register, and finite state machine.

- FIFO: In router design, three FIFO are used; each one consists of 16 bytes depth and 8 bits width, depending on control signals given by FSM, and it stores the data coming from the input port. The FIFO can be reset by a soft_reset signal; that is, an internal signal is an active high signal of that block coming from the

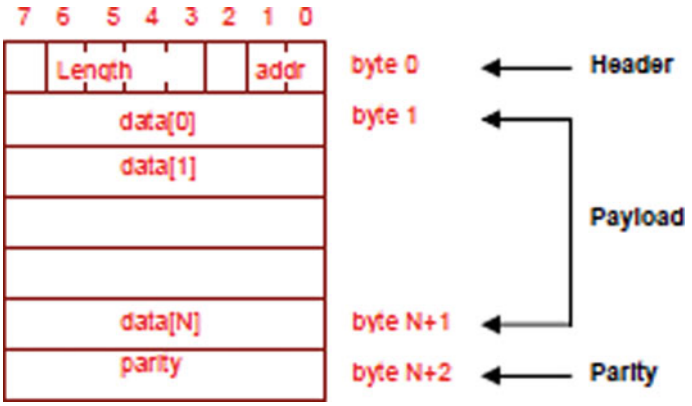

Fig. 3 Packet format

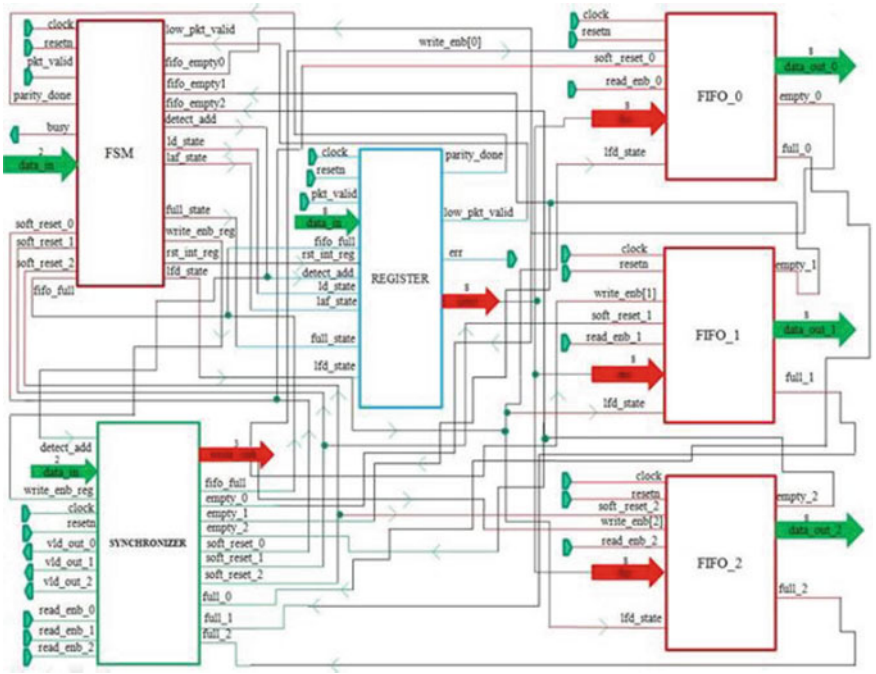

Fig. 4 Block diagram of top-level architecture

synchronizer module during a time out. Reset becomes low, then empty = 1, full = 0, and data_out = 0. Write operation and read operation occur when write_enb, read_end were high, the data_in, data_out sampled at the positive edge of clock, and FIFO is not become full, empty state. Write and read operations can be done at the same time. Full signal demonstrates that all the areas inside FIFO have been

written. Empty signal demonstrates that all the areas of FIFO are empty and have been read.

- Synchronizer: This block provides synchronization between FIFO nad FSM modules. It also provides correct information between one input port and three output ports.
- Register: This block implements four internal registers to hold, that is, packet parity byte, internal parity, FIFO full state, header byte; all these are register latched on the positive edge of the clock.
- FSM: This block was the controlled design of the router IP. When router gets the new packet, this block generates all controlled signals; these signals are used to transfer the packet to output by other design components.

4 Proposed Detection Method

4.1 Trojan-Free Implementation

In this router IP design, we implemented a HT in a FIFO block as shown in Fig. 5, in order to demonstrate the uses of inline assertions to detect HTs. The HT effect in a design is modification of functionality and leakage of critical information. The FIFO can be reset by a soft_reset in that signal Trojan was added; that is, an internal signal is an active high signal of that block coming from the synchronizer module during a time out. Synchronizer block has three out signal, that is, vld_out_x, and this signal is generate depending at the empty status of FIFO like conditions (vld_out_0 = ~ empty_0, vld_out_1 = ~ empty_1, and vld_out_2 = ~ empty_2). The signals soft_reset_0, soft_reset_1, and soft_reset_2 are three internal signals for each FIFOs, and these signals go high if the read_enb_x like (read_enb_0, read_enb_1, read_enb_2) is not assert within the 30 clock cycles of the vld_out_x. As explained functionality, now, after adding Trojan, the effect on those three internal reset signals of this block, reset signal goes high after one clock cycle without read_enb_x signal not assert within the 30 clock cycles of vld_out_x.

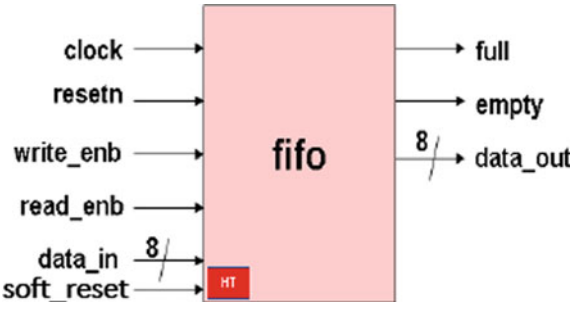

Fig. 5 FIFO block with hardware Trojan

4.2 Inline Assertions

Inline assertions are primarily used to validate the behavior of the design and capture the knowledge about how a design should operate. Assertions are the properties, which must be true. Assertion increases the controllability and observability of a design. Controllability is the measurement of the ability to activate, stimulate, or sensitize a specific point within the design. Observability is the measurement of the ability to observe the effects of a specific, internal, stimulate point within the design. Assertions monitor the expected behavior, forbidden behavior, and signal protocols and also depend on the quality of the stimulus.

The objective of the proposed methodology of IC design flow is shown in Fig. 6 to detect the HTs at the behavioral level of RTL block using inline assertions and verified by our IP design with UVM methodology. Inline assertions are embedded check-in RTL code executable specifications, during the simulation phase assertions monitor specific conditions that occur or a specific sequence of events occurs, expected behavior, forbidden behavior, and signals protocols.

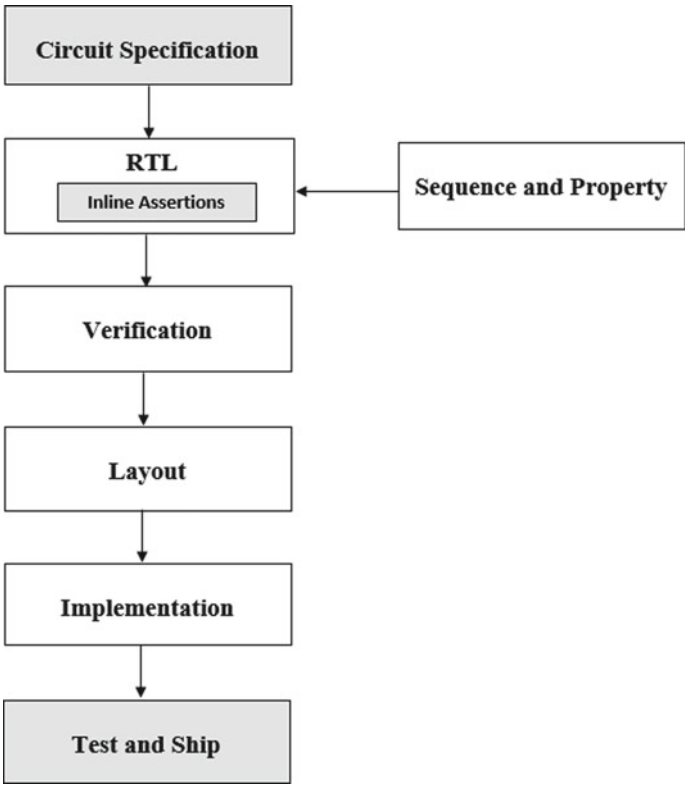

Fig. 6 Proposed IC design flow

Mapping the property and sequence into the proposed design module of inline assertions for Trojan detection. It also produces warnings and errors when a specified condition fails and the sequence does not complete properly. Inline assertions dependent on the clock cycles and test expression are evaluated at clock edges dependent on the variables involved for the sampled values. A variable sampling, evaluations are done at the preponed and observed region of the scheduler. Our inline assertions are placed in a module, interface, procedural block. It can be used with both dynamic and static tools.

5 UVM Verification

UVM methodology is a standard framework to build the verification environment; it has its base class library like uvm_component, uvm_sequence_item, uvm_object. In UVM language, TLM is used as a standard communication mechanism to achieve interoperability configuration of the test bench from the top level. It generates scenarios independent of the test bench environment. UVM achieves reusability in plug and play manner. Typical UVM test bench hierarchy is shown in Fig. 7.

Agent: UVM agent is also called universal verification component (UVC). An agent can be encapsulated, ready to use, reusable, and configurable components. It contains a driver, monitor, and sequencer. Test bench infrastructure can have more than one agent. It can configure as an active and passive agents. Driver: It gets data repeatedly from a sequencer; it drives the DUT based on the protocol using the virtual interface. Derive a driver from the uvm_driver base class. Monitor: The monitor extracts information from the bus signal and translates it into transactions. It is connected to other components, via standard TLM ports. Drive a monitor from

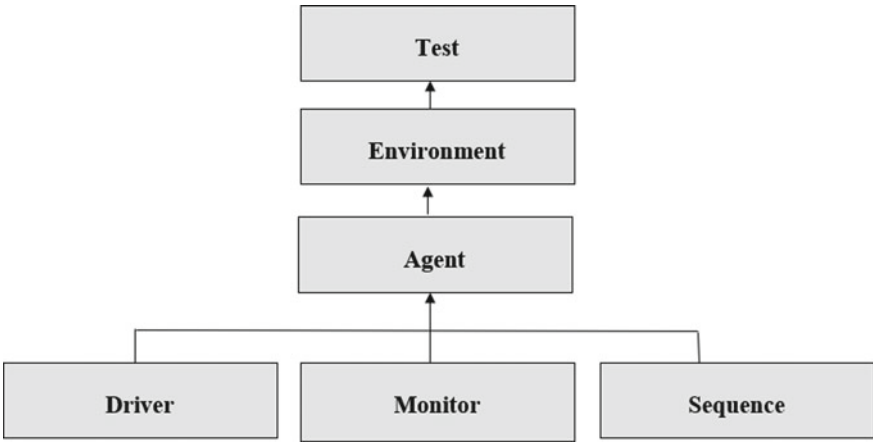

Fig. 7 UVM test bench hierarchy

the uvm_monitor base class. Sequencer: It creates stimuli depending on restrictions and can do so on the fly or at zero time. A factory can be used to override sequences. It is derived from uvm_sequencer. Environment: It is at top of the UVM test bench architecture and contains one or more agents depending on the design.

6 Result and Discussion

The simulation output result of the router top IP design with Trojan implementation is as shown in Fig. 8, and the result of the three data outputs is zero because of the Trojan present in the FIFO block. The data packet could not able to find data coming from source because read enable signal is not becoming high within the 30 clock cycles of valid signal.

The simulation output result of the router top IP design without Trojan implementation as shown in Fig. 9.

The output results of router IP design as shown in Figs. 8 and 9 are synthesized and implemented in Xilinx Vivado tool.

From, Table. 2 shows the output results of inline assertions, and it gives assertions coverage results for each signal at particular sequence and property. As Trojan was added in soft_reset signal, the inline assertions are failing at the soft_reset signal. So, it shows that the Trojan has detected. Inline assertions are implemented in Aldec Rivera Pro tool.

The simulation output result of the design under verification by using UVM is as shown in Fig. 10.

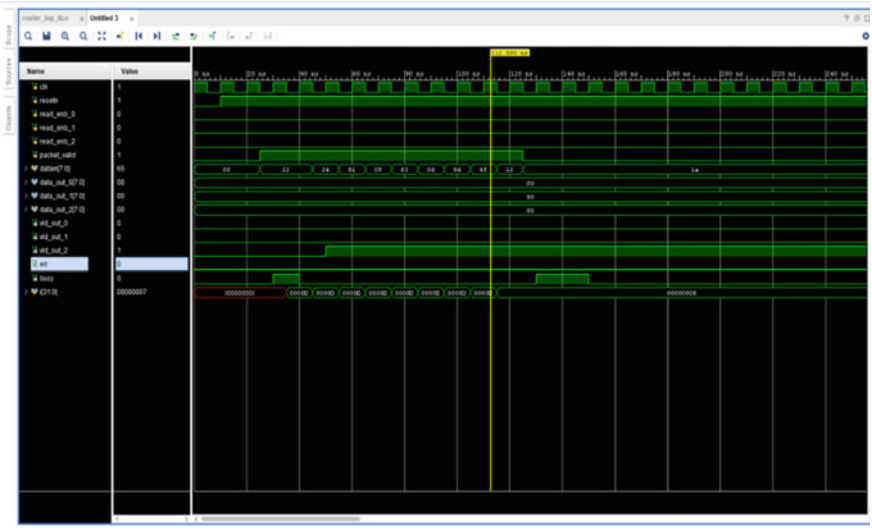

Fig. 8 Router top simulation result with Trojan

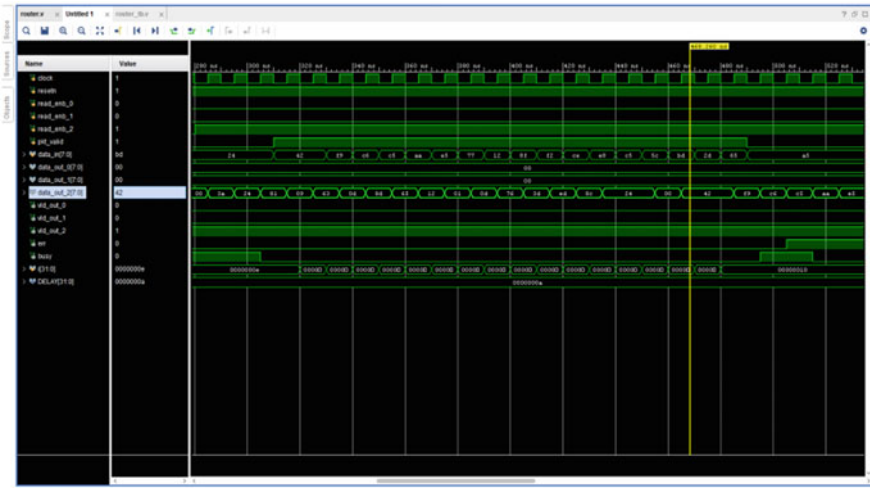

Fig. 9 Router top simulation result without Trojan

Table 2 Inline assertions result

Signal	Assertions		Assertions coverage (5)	Result
	Sequence	Property		
Reset	reset_seq	reset_prty	100	Passed
Busy	busy_seq	busy_prty	100	Passed
read_enb	read_seq	read_prty	100	Passed
ld_state	pvld_seq	pvld_prty	100	Passed
pkt_vld	pvld_seq	pvld_prty	100	Passed
vld_out	vldo_seq	vldo_prty	100	Passed
lfd_state	deassert_seq	deassart_prty	100	Passed
Empty	vldemp_seq	vldemp_prty	100	Passed
Full	fifo_seq	fifo_prty	100	Passed
soft_reset	vld_soft_seq	vld_soft_prty	0	Failed
parity_done	psns_seq	psns_prty	100	Passed
low_pkt_vld	parity_seq	parity_prty	100	Passed

Table 3 shows the result of power and area of router IP design with inline assertions. Our design is synthesized in 90 nm technology with Synopsys DC. From Synopsys DC tool, power and area results are obtained.

The code coverage results are as shown in Fig. 11; total cumulative is statement coverage (SC) with 87%, branch coverage (BC) with 80%, expression coverage with 45%, condition coverage with 69%, and Toggle coverage with 48%.

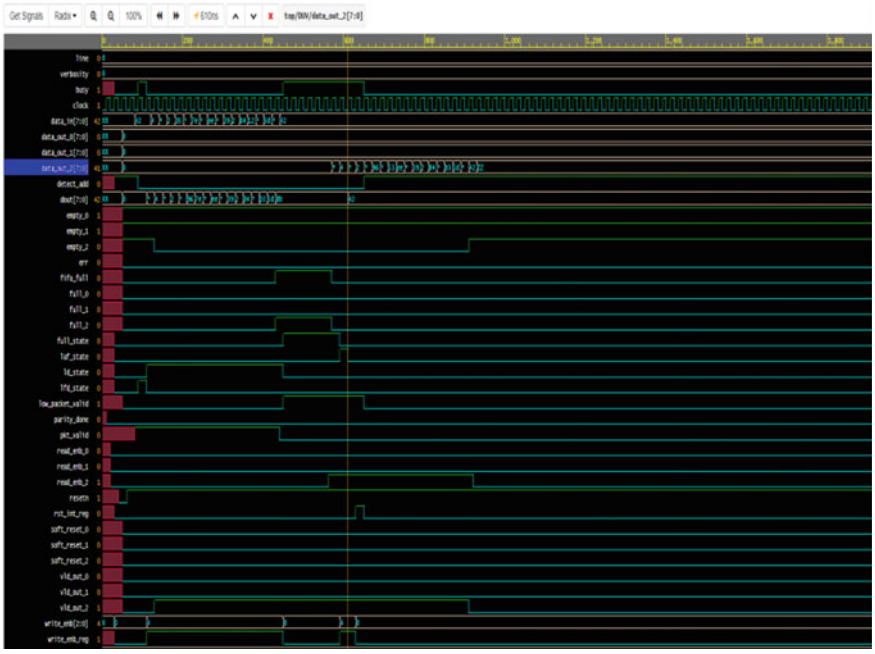

Fig. 10 DUV output result

Table 3 Power and area result

	Circuit
Parameter	Router IP design
Total power (uW)	8.290
Total area (μm^2)	2221.595629

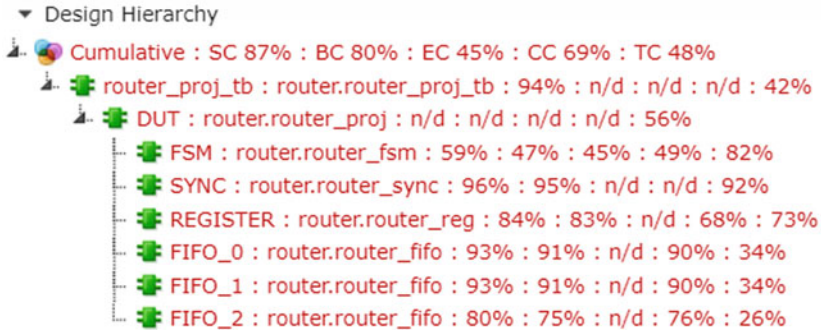

Fig. 11 Code coverage report

Table 4 Functional coverage report

Coverage type	Hits% (%)	Goal/at least (%)	Status
Coverpoint coverage	100	100	Covered
Covergroup coverage	80.555	100	Uncovered

From, Table 4 shows the result of the functional coverage, all test cases or test scenarios have been passed, and all bins are covered.

7 Conclusion

This research work has successfully designed and implemented the router IP protocol using Verilog HDL language. Also, this research work has proposed the application of inline assertions in order to detect HTs at the behavioral level of the design or system. Router IP design with inline assertions occupied very little power and area. And also, the proposed design is verified with different test cases using UVM methodology. Assertions must be defined carefully; incurrent assertions can give misleading results. Debugging assertions will be difficult.

References

1. M. Tehranipoor, C. Wang, *Introduction to Hardware Security and Trust* (Springer, New York, NY, USA, 2011)
2. Y. Jin, N. Kupp, Y. Makris, Experiences in hardware Trojan design and implementation, in *Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST '09* (IEEE Computer Society, Washington, DC, USA, 2009), pp. 50–57
3. T.F. Wu, K. Ganesan, Y.A. Hu, H.-P. Wong, S. Wong, S. Mitra, TPAD: hardware Trojan prevention and detection for trusted integrated circuits. IEEE Trans. Computer-Aided Des. Integr. Circ. Syst. **35**(4), 521–534 (2016). https://doi.org/10.1109/TCAD.2015.2474373
4. M. Tehranipoor, F. Koushanfar, A survey of hardware Trojan taxonomy and detection. IEEE Des. Test Comput. 10–25 (2010)
5. M. Hemachand, E. Prabhu, Secured netlist generation using obfuscation technique. J. Critical Rev. **7**(4), 878–881 (2020)
6. V. Veena, E. Prabhu, N. Mohan, Improved test coverage by observation point insertion for fault coverage analysis. Proc. Int. Conf. Trends Electron. Inform. ICOEI 2019 **8862789**, 174–178 (2019)
7. C. Mattihalli, S. Ron, N. Kolla, VLSI based robust router architecture. Third Int. Conf. Intell. Syst. Modell. Simul. **2012**, 43–48 (2012). https://doi.org/10.1109/ISMS.2012.32
8. S.R. Ramesh, R. Jayaparvathy, Artificial neural network model for arrival time computation in gate level circuits. Automatika **60**(3), 360–367 (2019)
9. S.R. Ramesh, R. Jayaparvathy, Probabilistic activity estimator and timing analysis for LUT based circuits. Int. J. Appl. Eng. Res. **10**(13), 33238–33242 (2015). ISSN 0973-4562

10. M. Boule, Z. Zilic, Efficient automata-based assertion-checker synthesis of PSL properties. IEEE Int. High Level Des. Valid. Test Workshop **2006**, 69–76 (2006). https://doi.org/10.1109/HLDVT.2006.319966
11. M. Bilzor, T. Huffmire, C. Irvine, T. Levin, Evaluating security requirements in a general-purpose processor by combining assertion checkers with code coverage. IEEE Int. Symp. Hardware-Oriented Secur. Trust **2012**, 49–54 (2012). https://doi.org/10.1109/HST.2012.6224318
12. X.T. Ngo, J.-L. Danger, S. Guilley, Z. Najm, O. Emery, Hardware property checker for run-time hardware Trojan detection. Proc. IEEE ECCTD, 1–4 (2015)
13. U. Alsaiari, F. Gebali, Hardware Trojan detection using reconfigurable assertion checkers. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **27**(7), 1575–1586 (2019). https://doi.org/10.1109/TVLSI.2019.2908964
14. IEEE Standard for Universal Verification Methodology Language Reference Manual, in IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017), pp. 1–458, 14 Sept. 2020. https://doi.org/10.1109/IEEESTD.2020.9195920
15. R. Madan, N. Kumar, S. Deb, Pragmatic approaches to implement self-checking mechanism in UVM based TestBench. Int. Conf. Adv. Comput. Eng. Appl. **2015**, 632–636 (2015). https://doi.org/10.1109/ICACEA.2015.7164768