IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

Deployment of Formal Verification for RISC Processor RTL Signoff

Arpit Awasthi[1] , VNR Vignana Jyothi Institute of Engineering & Technology
Dr Kalapala Vidya Sagar[2], Prof EIE, VNR Vignana Jyothi Institute of Engineering & Technology
G.Venkata Hari Prasad[3], CMR College of Engineering & Technology,
drgvenkatahariprasad@cmrcet.ac.in
Aneesha Dacha[4], VNR Vignana Jyothi Institute of Engineering & Technology
D Ramya Sai[5], VNR Vignana Jyothi Institute of Engineering & Technology
Mail ID: aneesha.dacha@gmail.com

Abstract:
With increasing logic density being encapsulated in modern-day processing electronic chips, verification has been an increasingly taxing endeavour with enormous engineering and human resources to validate the integrity of design Intellectual Property modules. The primary simulation-driven testbench environment is not supporting exhaustive assessment to the stringent adherence to specifications, and these missed bugs percolate to the silicon stage, making defective chips. This paper demonstrates the successful deployment of a robust novel Formal Verification based methodology for verifying of RTL Design Code of the RISC Processor. The proposed methodology is also complemented with visualization of relevant quantitative signoff metrics for capturing insights about quality of Hardware Description Language (HDL) Code aiding in reducing time for RTL Freeze.
Keywords:
Formal Verification, RISC, SystemVerilog Assertions, Coverage, RTL Signoff

Introduction:

The verification RTL Signoff endeavour primarily deals with evaluating compliance with specifications. Conventional techniques have been focused on generating stimulus to activate the Design Under Test (DUT) and observe output behaviour. The major pitfalls in such an approach are poor quality of randomized stimulus fails to exhaustively evaluate the design leaving many unverified regions in RTL Code. 2020 Wilson Research Group Functional Verification Study [1] did an elaborative study and concluded that statistically, 68 per cent of projects are running behind expected Time to Market constraints and multiple respins of happening to owe to the non-detection of logical functional bugs. Due to this non-exhaustive stimulus in a simulation-based approach, multiple regressions and randomization with different seeds are required, which consumes exorbitant time in building different verification testbench components and debugging them. This calls for a serious introspection to overcome these inefficient verification methodologies, and this necessitated the development of novel analytical frameworks with a combination of semantic analysis and formal methods to develop RTL Signoff methodology [2]. This paper adopts a Formal Verification based Model Checking approach for a RISC based processor for RTL Signoff and adopts an empirically driven Signoff approach coupled with mutation analysis to develop a Signoff approach.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

Literature Survey:

Formal Verification (FV) is a rigorous mathematical algorithmic approach in which the different temporal activities in design are captured as properties and then fed to a Formal Verification tool to test those scenarios exhaustively by applying all the possible combinations [3]. The adoption of FV was extremely limited for RTL Signoff, but recent advancements in SMT and SAT-based solvers have enabled enhanced adoption and development of RTL Signoff Techniques [4] [5] [6]. Siegal [7] laid the novel foundational work of developing an empirically driven property development approach for exhaustive verification using formal techniques. There is currently a lack of systemic methodology and techniques for deploying for End-to-End RTL Signoff. Yalin [8] outlines an effective strategy for combating issues encountered in deploying FV, aiding in seamlessly mitigating issues encountered. Nicole et al. [9] [10] demonstrate a SystemVerilog Assertions (SVA) [11] based approach for detecting verification blindspots and Hardware Trojans vulnerabilities for robust assessment. Ronak et al. [12] exhibit successful integration of FV to shrink verification signoff at subsystem level. N. Bombieri et al. [13] [14] presents an Assertion Based Verification (ABV) environment solution to build assertion reusable libraries and plug the gap with respect to bug escapes. B. Alizadeh et al. [15][16] introduced a formal debugging approach coupled with mutation analysis to detect multiple functional specification mismatch in a shorter run time. P. Aggarwal et al. [17] illustrate a robust coverage driven formal methodology for determining the effectiveness of different abstraction models and enhance coverage metric. In the existing Simulation approach, a Constrained Random Verification (CRV) based methodology is adopted to generate a stimulus that triggers DUT, and output is analyzed with respect to state transition soundness and integrity. For a simulation-based approach, the input test vectors are randomized. Figure 1 represents the block diagram of the simulation-based Dynamic Verification Methodology. The primary conclusion drawn is Simulation-based testbench environment takes a large amount of time to craft different testbench components like Transactor, Generator, Driver with multiple interfaces. There are a plethora of Coverage holes and corner-case bugs escaping causing respins along with multiple regressions and delayed Time to Market (TMM), which depicts that it is not an efficient approach.

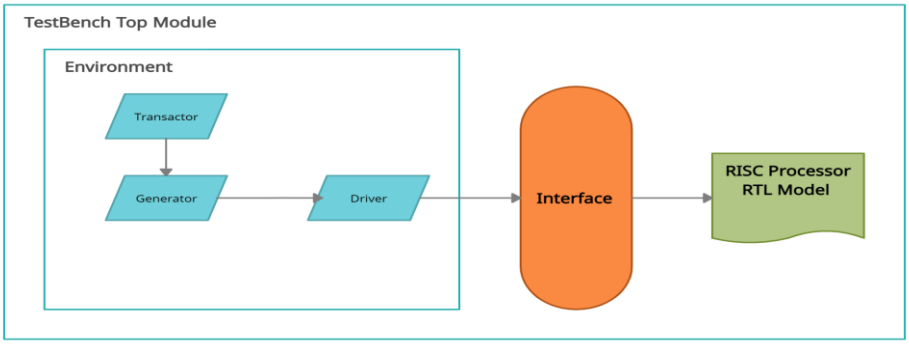

Figure 1 Dynamic Simulation Verification Environment for RISC Processor

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

Research Methodology

This paper presents a novel formal verification methodology approach which is coupled with the evaluation of different metric relevant to qualify the integrity of the Design RTL Model. In Formal Verification Methodology, the design intent from specifications and all the temporal activities are captured in the form of SystemVerilog Assertions (SVA). These SVA form the building blocks for crafting checkers, including Boolean expressions, sequences, and properties that are all finally integrated to build SVA's, as shown in Figure 2. Cover statements are scripted to assess the quality of stimulus generated and evaluate if all stimulus scenarios are generated to be driven to the Design module. Assume statements are used to set any configuration environment for setting any constraints. These three statements, i.e., Asserts, Cover and Assume, are then fed to the Formal Verification Tool to perform an exhaustive analysis on DUT. The tool being used for Formal Analysis is Synopsys VC Formal.

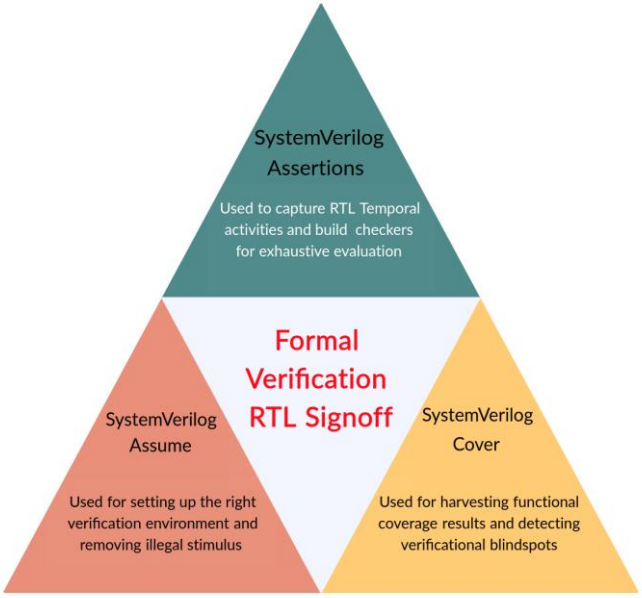

Figure 2 Components of Formal Verification

SystemVerilog Assertion (SVA) based Model Checking methodology is employed for the Formal Verification of RISC Processor. The microarchitecture implemented of the RISC Processor is shown in Figure 3. SVA is crafted depending on the test plan after assessing the specifications of the RISC Processor. The tool being used is Synopsys VC Formal Verification Tool. All the temporal design activities are captured in the form of SVA and fed to the tool. The tool gives the advantage. This approach's advantage is that verification needs not to be dependent on the limited scenarios scripted, but the tests will be exhaustive, covering the entire design space. To weed out, illegal scenarios assume statements can be integrated into SVA. The tool also gives liberty to select the type of engine solver to be selected and the type of bounded proof depth required to assess the integrity of the design. Few operational integrity bugs passed in the simulation were detected by the Formal

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

approach, which was critical. Thus, FV proved to be a potent arsenal against simulation resistant bugs, which, if escaped to silicon, could lead to faulty behaviour of hardware. A detailed formal methodology flow along with metrics to be evaluated is shown in Figure 4. The novelty of the methodology lies in adopting a metric-driven approach with analysis like sanity linting checking, assertion density evaluation, coverage checking, connectivity checking and mutation analysis.

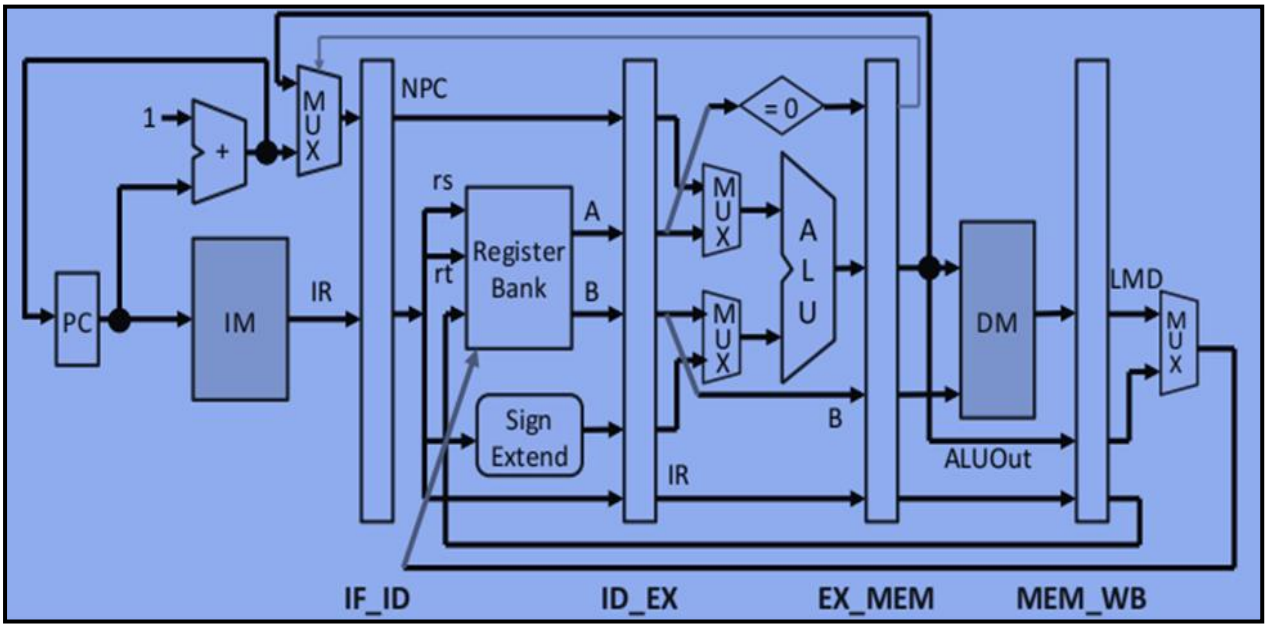

Figure 3 RISC Architecture Block Diagram

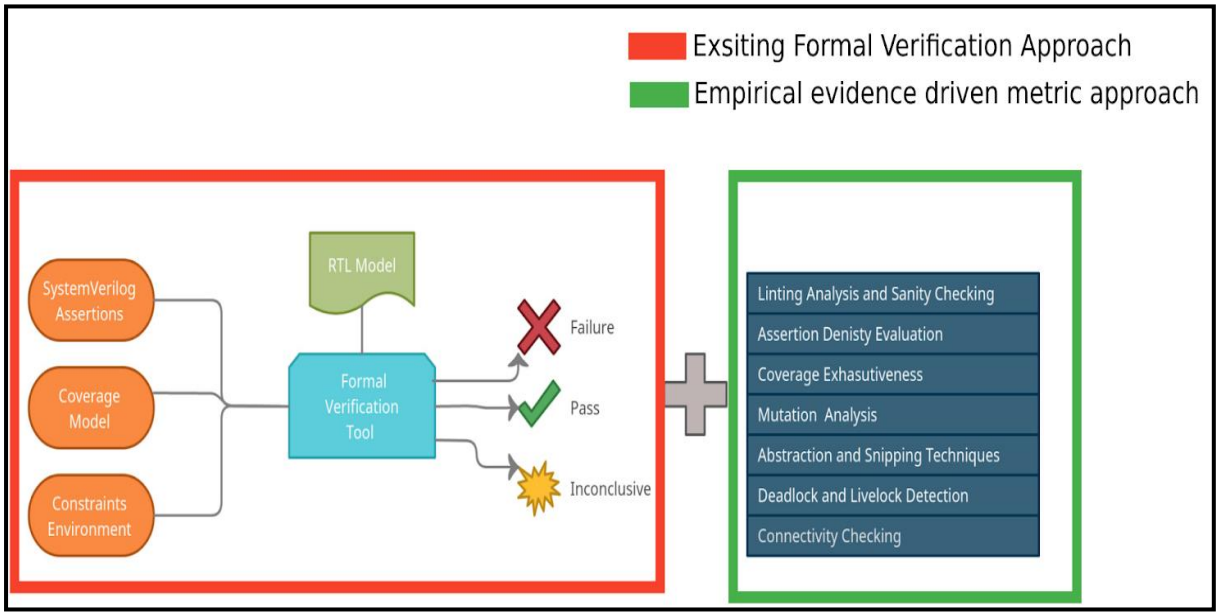

Figure 4 Proposed Formal RTL Signoff Methodology

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

Results Analysis and Discussion:

This section provides important insights regarding all the individual parameters which are required to perform the formal analysis on RISC Processor Verification results. The rationale behind their essentiality is also equally highlighted intertwined with relevant empirical visualization.

Linting Analysis and Sanity Checking:

Linting Analysis is the first step in RTL Verification. As shown in Fig 5, many of these issues did not get detected in simulation-based techniques. They escaped to It can help in doing sanity checking by eliminating issues such as Non-Synthesizable constructs, Unintentional latches, Unused declarations, Driven and undriven signals, Race conditions, Incorrect usage of blocking and non-blocking assignments, Incomplete assignments in subroutines, Case statement style issues, Set and reset conflicts and Out-of-range indexing range issues. The detection of these Linting errors is critical for performing preliminary sanity checking and weeding out Linting issues embedded in the HDL Design Model. If they are not thwarted at this stage, these will percolate down to further stages, causing critical issues at Synthesis Stage, resulting in corrupted gate-level netlist leading to structural faults in silicon, causing faulty chips to get manufactured. 26 Linting were detected in Formal Analysis, which escaped simulation-based approach, thus aiding in nipping the bug at preliminary stages of sanity checking by performing root cause analysis.

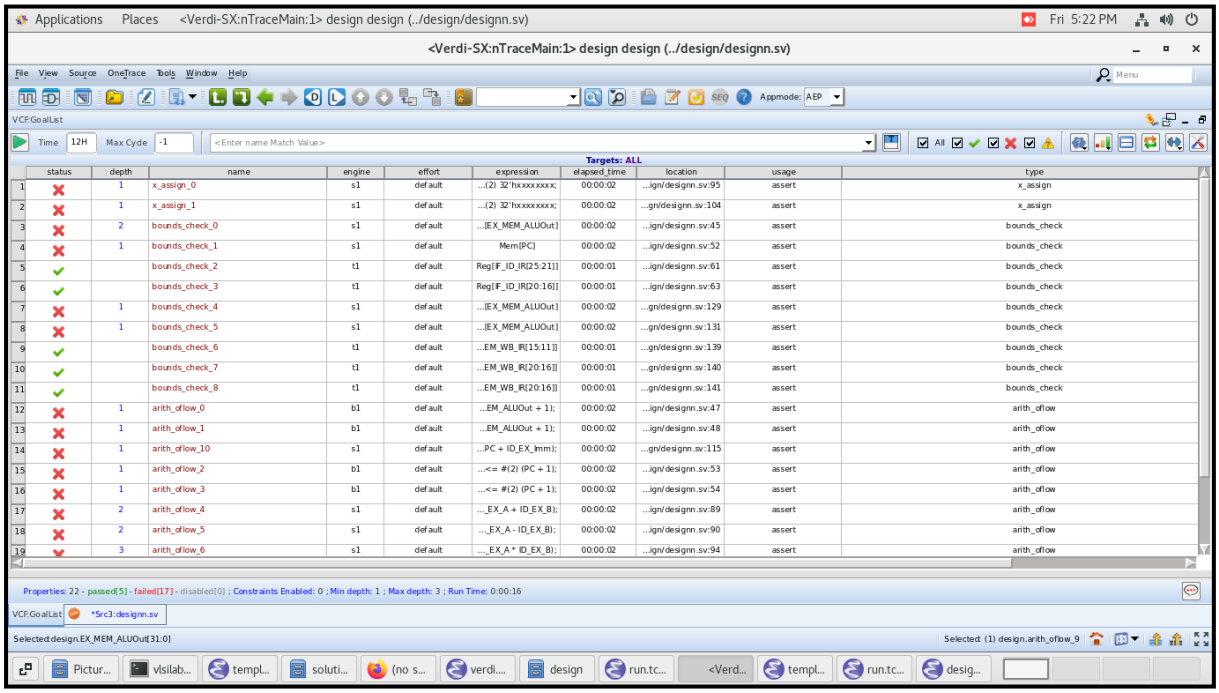

Figure 5 Linting and Sanity Checking Results

RTL Schematic:

The VC Formal Tool is used to generate the RTL Schematic view shown in Figure .6 of the entire processor core, depicting all the functional blocks like Memory Unit used to store

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

instructions binary data, Instruction Fetch Decode Unit, Arithmetic Logic Unit and Memory Writeback Unit. All the different functional units are examined for their connectivity checking, ensuring the data propagation is not hindered, and forward progression with seamless data updates policy to the memory after computation from ALU is executed. This is extremely critical to ensure the connectivity of all the submodules ensuring there are no data progress issues from percolating from memory modules to execution units.

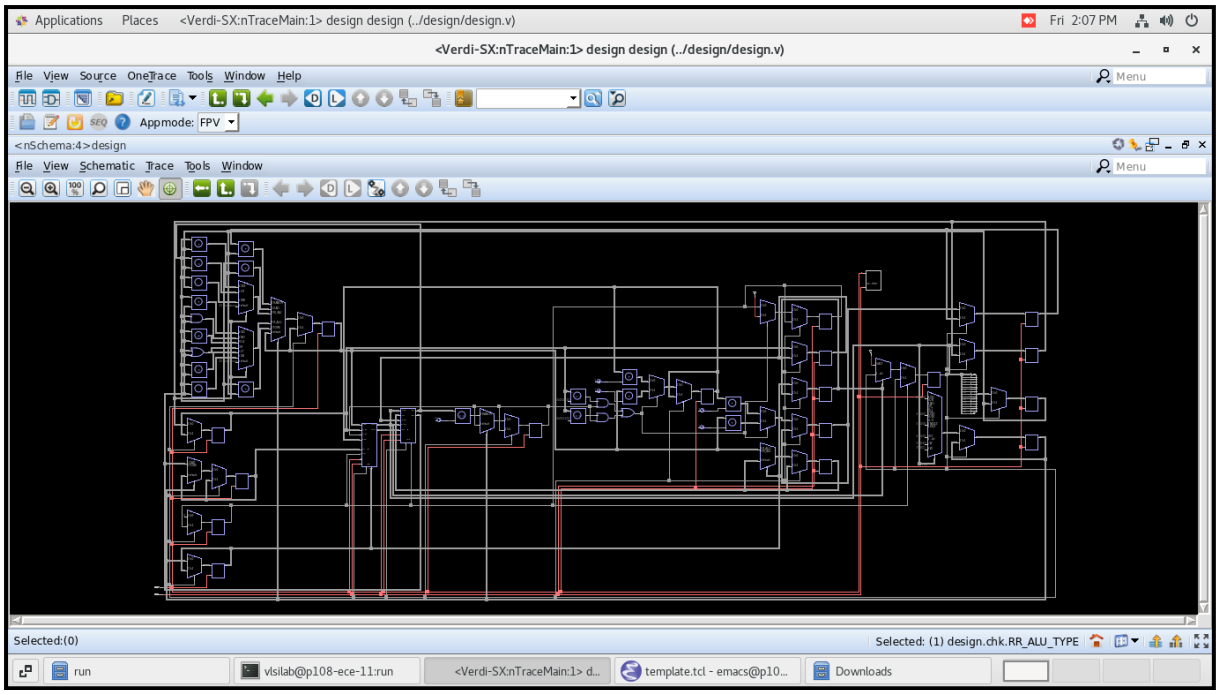

Figure 6 RTL Schematic of RISC Processor

Coverage Analysis:

The Formal Coverage Analyzer (FCA) application mode is used to run unreachability analysis and find coverage properties in design that cannot be covered with any stimulus. This helps to find exceptions that can be used to achieve coverage closure much faster. But, before using these exceptions for closing coverage, they need to be analyzed carefully to ensure that they are not masking any potential bugs in the design. As shown in Figure 7, the FCA mode metrics can perform coverage analysis of the vital performative metrics like Line, Condition, Branch, Toggle, Finite State Machine (FSM) state, FSM transition, SystemVerilog Covergroups. A total of 96 cover statements are scripted to exhaustively assess the checking of all internal register state values and plug the cover holes by triggering more stimulus.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

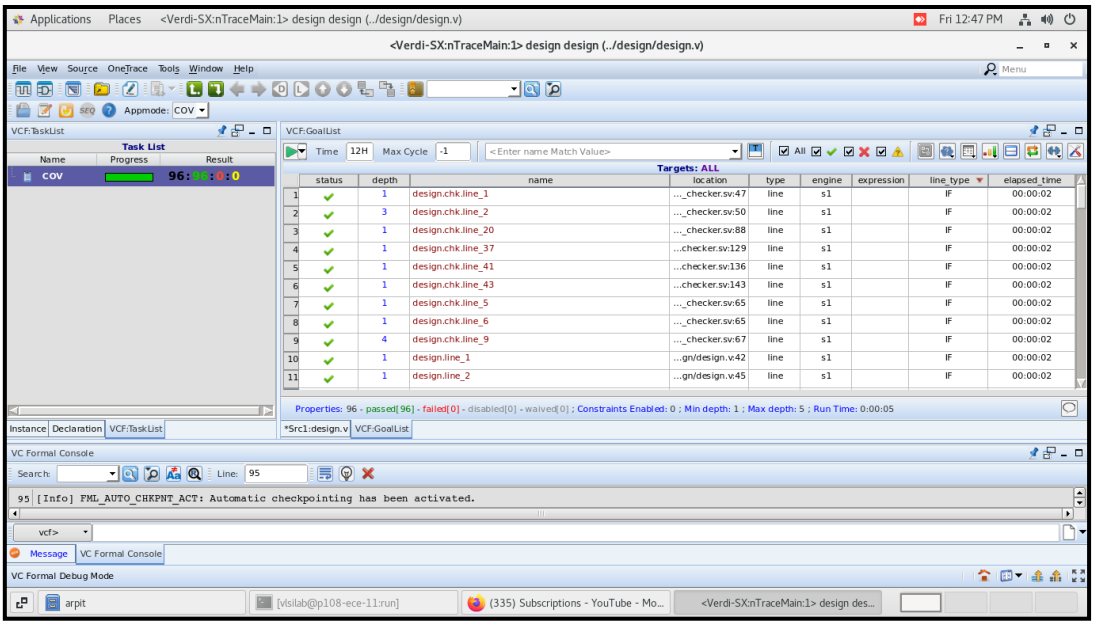

Figure 7 Coverage Analysis

Property Density Report:

Figure. 8 shows that Property density coverage provides structural coverage of the design code in the COI of all the properties. Bounded depth coverage reachability analysis helps identify whether the required design code is covered within the specified proof depths. It is empirical proof of all the internal registers, assertions statements and line of RTL codes that are covered to detect any verification blindspots, which would aid us in crafting additional SVA's to plug those corner cases scenarios left unverified. This kind of feedback mechanism approach eventually helps in adding a layer of verification confidence helping in building robust RTL Models devoid of bugs.

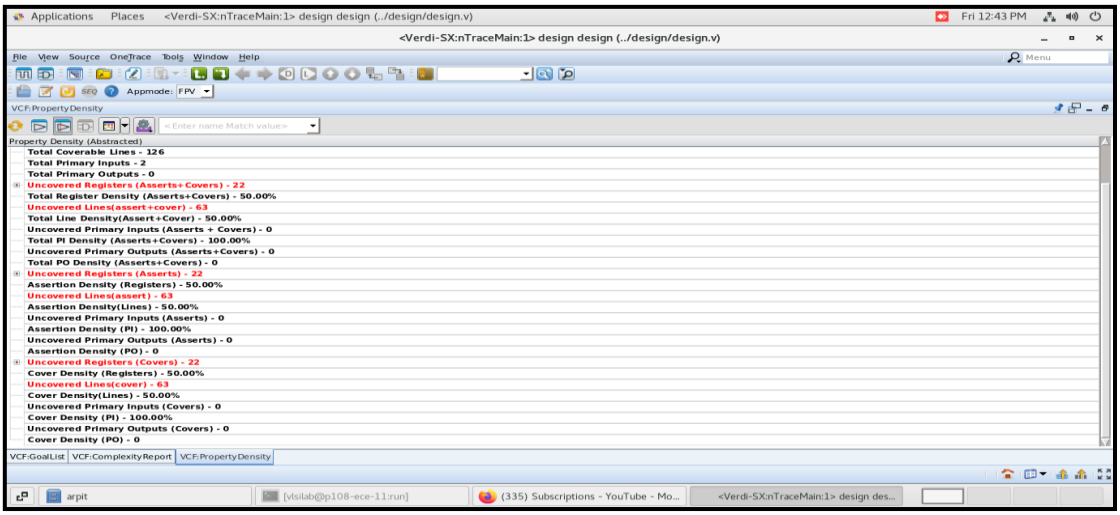

Figure 8 Property Density Report

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

Deadlock and Livelock Detection:

The two kinds of forwarding progress issues that are being tackled are Deadlock and Livelock. Deadlock is a critical flaw in an RTL Design in which the FSM state transition to other states is not possible, and the design gets stuck in a particular deadlock state with no recovery mechanism. Livelock is a phenomenon in which the state transitions only loops between certain states recursively and cannot manoeuvre out of the trapped states. The processor core has 3 stage pipeline architecture, so it becomes critical to detect any stalls or glitches encountered in the forward progress of data due to deadlock issues. The timing diagrams during debug analysis are critically investigated to ensure the temporal specifications align with expected behaviour. As shown in Figure 9, the temporal timing behaviour is critically examined for adherence to the specifications document.

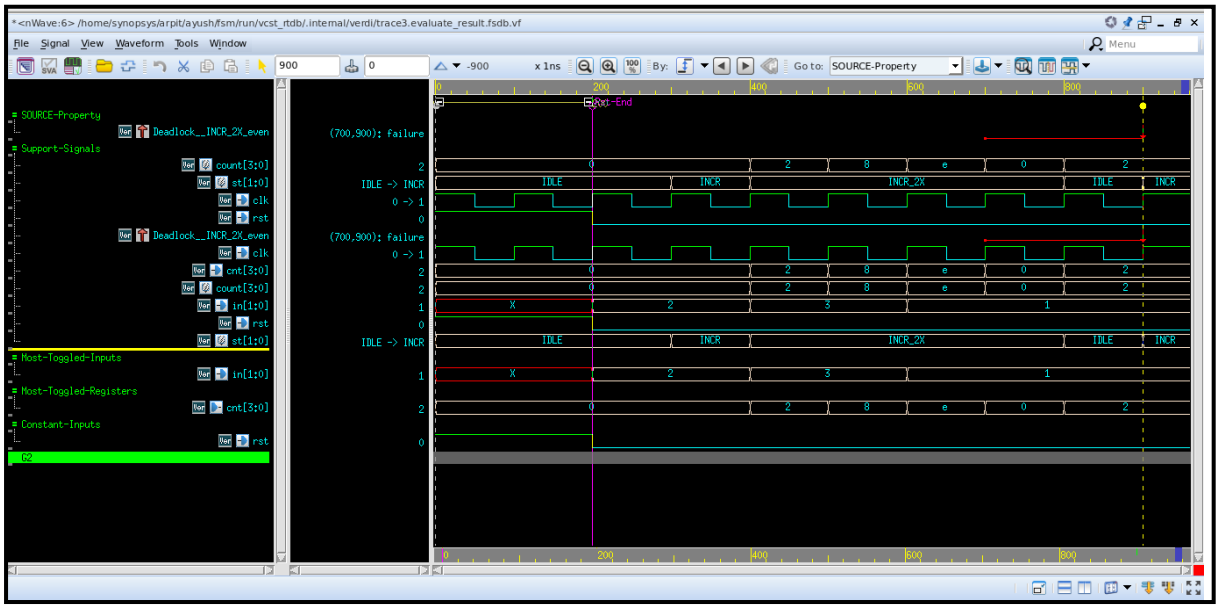

Figure 9 Temporal Behaviour of RISC Processor

Mutation Analysis in RISC Processor:

Mutation Analysis is an essential technique being adopted in Formal Verification to assess the quality of SystemVerilog Assertion and Covergroups and determine their robustness in detecting artificial bugs implanted inside Design Verification environments. Tweaks are made in the RTL functionality by modifying the RISC Processor HDL Design Code via fault injection mechanisms. In the first phase, incorrect code is inserted into the RTL Code of Instruction Decode block, followed by 2nd phase, in which code corruption in ALU Block. The modified faulty HDL Code is then evaluated by the Assertions and Covers statements built, and their utility was proven as SVA Library based Formal Testbench successfully detected faults inside the modified RTL. The same is depicted in Figure 10, which highlight observable interjections at the terminal nodes for checking the propagated fault. All the

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

flagged failed assertions have successfully demonstrated their ability to catch bugs via Synopsys VC Formal Tool. A similar process is also tried by fault injecting mechanisms in the Verification environment by tweaking the Assertion and Coverage library and assessing mismatches if any false positives get triggered. Thus, the fault injection-based Mutation Analysis mechanism helped strengthen the verification and design environments by further refining Assertions checkers and Coverage points.

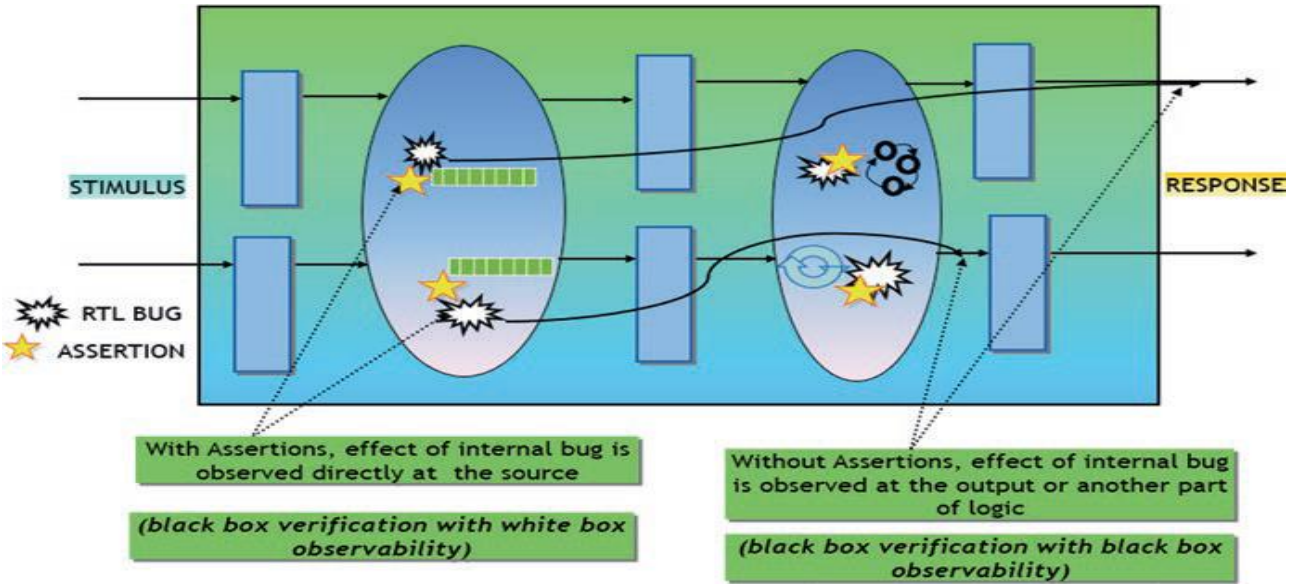

Figure 10 Fault injection analysis

Conclusion:

This paper conclusively demonstrates the effectiveness of deploying Formal Verification for RTL Signoff of RISC Processor compared to a dynamic simulation environment. It has aided in the rapid verification of the RTL Model with exhaustive coverage analysis coupled with corner-case bug detections and timing analysis metrics. All the opcodes, along with their operations intended as per specifications, are validated.

References:

1. M.Graphics,"https://blogs.sw.siemens.com/verificationhorizons/2020/10/27/prologue-the-2020-wilson-research-group-functional-verification-study/",[Online: accessed 1-4-2021]
2. P. Ashar, "A paradigm shift in verification methodology," 2016 Formal Methods in Computer-Aided Design (FMCAD), Mountain View, CA, USA, 2016, pp. 6-6, DOI: 10.1109/FMCAD.2016.7886652.
3. Carl Seger, An Introduction to Formal Hardware Verification, University of British Columbia, Vancouver, BC, Canada, 1992
4. N. Een, A. Mishchenko and R. Brayton, "Efficient implementation of property directed reachability," 2011 Formal Methods in Computer-Aided Design (FMCAD), Austin, TX, USA, 2011, pp. 125-134.
5. Aaron R. Bradley. 2011. SAT-based model checking without unrolling. In

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11,S Iss 3, Dec 2022

Proceedings of the 12th international conference on verification, model checking, and abstract interpretation. Springer-Verlag, Berlin, Heidelberg, 70–87.

6. B. Ustaoglu, S. Huhn, F. Sill Torres, D. Große and R. Drechsler, "SAT-Hard: A Learning-Based Hardware SAT-Solver," 2019 22nd Euromicro Conference on Digital System Design (DSD), Kallithea, Greece, 2019, pp. 74-81, doi: 10.1109/DSD.2019.00021.
7. M. Siegel, "Achieving earlier verification closure using advanced formal verification," Formal Methods in Computer-Aided Design, Lugano, Switzerland, 2010, pp. 275-275.
8. Hu, Yalin. "Exploring formal verification methodology for FPGA-based digital systems." *Sandia National Laboratories, New Mexico, California* (2012).
9. N. Fern and K. Cheng, "Evaluating Assertion Set Completeness to Expose Hardware Trojans and Verification Blindspots," 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 2019, pp. 402-407, DOI: 10.23919/DATE.2019.8714883.
10. N. Fern, I. San, Ç. K. Koç and K. Cheng, "Hardware Trojans in incompletely specified on-chip bus systems," 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2016, pp. 527-530.
11. "IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language," in IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012), vol., no., pp.1-1315, 22 Feb. 2018, DOI: 10.1109/IEEESTD.2018.8299595.
12. R. M. Sarikhada and P. K Shah, "Speed up the validation process by formal verification method," 2020 IEEE International Conference for Innovation in Technology (INOCON), Bengaluru, India, 2020, pp. 1-4, DOI: 10.1109/INOCON50539.2020.9298384.
13. N. Bombieri, R. Filippozzi, G. Pravadelli and F. Stefanni, "RTL property abstraction for TLM assertion-based verification," 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2015, pp. 85-90, DOI: 10.7873/DATE.2015.0121.
14. T. Ghasempouri, A. Danese, G. Pravadelli, N. Bombieri and J. Raik, "RTL Assertion Mining with Automated RTL-to-TLM Abstraction," 2019 Forum for Specification and Design Languages (FDL), Southampton, UK, 2019, pp. 1-8, DOI: 10.1109/FDL.2019.8876941.
15. B. Alizadeh, P. Behnam and S. Sadeghi-Kohan, "A Scalable Formal Debugging Approach with Auto-Correction Capability Based on Static Slicing and Dynamic Ranking for RTL Datapath Designs," in IEEE Transactions on Computers, vol. 64, no. 6, pp. 1564-1578, 1 June 2015, DOI: 10.1109/TC.2014.2329687.
16. R. Sharafinejad, B. Alizadeh and T. Nikoubin, "Formal Verification of Non-Functional Strategies of System-Level Power Management Architecture in Modern Processors," 2020 IEEE 14th Dallas Circuits and Systems Conference (DCAS), Dallas, TX, USA, 2020, pp. 1-6, DOI: 10.1109/DCAS51144.2020.9330633.
17. P. Aggarwal, D. Chu, V. Kadamby and V. Singhal, "Planning for end-to-end formal using simulation-based coverage," 2011 Formal Methods in Computer-Aided Design (FMCAD), Austin, TX, USA, 2011, pp. 9-16.