International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 09 | Sep 2022 www.irjet.net p-ISSN: 2395-0072

Formal Verification Of An Intellectual Property In a Field Programmable Gate Array

Prashanth M[1], Sujatha K[2]

[1]*Department of Electronics and communication, BMS college of engineering, Bangalore, India*
[2] *Associate Professor, Dept. of Electronics and communication, BMS college of engineering, Bangalore, India*

Abstract - *The verification method has fallen behind, nevertheless, as a result of increased IC production capabilities. The verification phase of the circuit design flow, according to ITRS, has taken the longest. Verification engineers now outnumber design engineers in terms of the number of active engineering projects. Verification is now the IC design industry's bottleneck since the ratio for sophisticated designs might approach 2:1 or 3:1. Verification will become the main obstacle for the future growth of the IC design business if significant advancements are not made.*

Our suggested formal verification will compare the reference model with the implementation model using Jasper Gold formal verification tool. With the necessary design changes made without sacrificing functionality, we were able to achieve an assertion pass rate of 80%.

Key Words: **Formal verification, Intellectual property, counter example, Sequential equivalence checking, Combinational equivalence checking.**

1. INTRODUCTION

Formal verification is the process of utilizing mathematical tools to confirm the accuracy of the design. Timing checks are not carried out by formal verification tools, which instead utilize a variety of techniques to verify the design. Since these tools don't require a test bench or stimulus, once an RTL code is ready, formal verification can be carried out, A bug is easier to correct the sooner it is discovered[1]. Formal verification techniques find bugs that are missed by standard verification methods. Moreover, formal verification typically identifies flaws rather more quickly than standard methods do in cases when they are detectable. A design first should undergo formal verification before being functionally tested through simulation and emulation. Essentially, the DUT is a network of flip-flops and logical gates. Equations may be used to express this network of gates and flip-flops. The tool then independently evaluates each checker (i.e., SVA assertion), looking for any possible input sequences that might show the checker to be untrue. If such a sequence is discovered, a waveform illustrating this erroneous situation is shown. This waveform is known as a counter example in formal language (CEX)[5].

The goal of formal verification is to detect the bugs in the early stage of the design and in less time, we can verify the design specifications. The main types of equivalence checking in formal verification is logical equivalence checking and sequential equivalence checking. Logical equivalence checking, sometimes referred to a process known as combinational equivalence checking, procedure of determining if two architectures share the same combinational logic between registers. The number of registers in the two designs under comparison need to be equal as well. This method is used to confirm that two designs with various levels of abstraction, like a gate-level netlist, are functionally similar and a layout netlist are functionally equivalent[3].

The technique of confirming that whether two designs specifications are equivalent or not and produce the similar results when given the similar inputs is known as sequential equivalence checking. The sequential logic of two designs, which may have different implementations, is compared by the SEC as shown in figure 1.1.

Some more logic, including scan-based logic, power control circuits, etc. Verification of such modifications is required. Regular verification processes take a long period, which extends the time to market. The changed design is compared to the golden design using sequential equivalence testing to ensure that they are functionally equal.

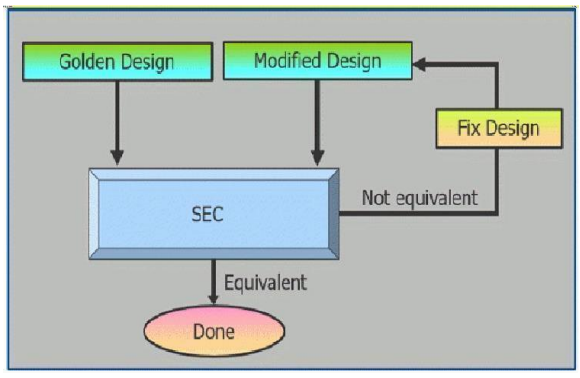

Fig -1: Sequential Equivalence Checking

The ideal candidates for formal verification are these short, control-oriented pieces that are repeated frequently. All of Written assertions and assumptions for the design and interaction using SVA. In order to ensure that every component is covered during formal verification of the

design., cover points are also defined in system Verilog. Additionally, as part of the subsystem level verification, the assertions created for these modules are reused. Equivalence checking includes sequential equivalence checking gives application of sequential equivalence checking in clocked gated circuits. Compared to general sequential equivalency checking, sequential logic synthesis frequently results in significantly simpler equivalence checking difficulties (SEC). When sequential equivalence checking can be transformed into a combinational equivalence check.

2. FORMAL VERIFICATION METHODOLOGY

Gathering the specifications and needs of the desired product is the first step in the development cycle of any semiconductor chip. To obtain the desired behavior from the hardware unit, design engineers construct the register transfer logic. The Design Under Test (DUT) is examined for its functional and structural properties according to the suggested approach before the verification activities are actually implemented, as shown in Fig. 1. A proper inspection strategy must be established on basis of analysis and taking into account the resources accessible for the purpose of verification.

Formal Verification has the ability to verify every piece of digital hardware, but it is necessarily constrained by the design state space explosion problem. Therefore, the DUT has to be examined for its suitability for formal analysis based on the scheduled/available time for verification and resource variables. The hardware designs that are most suited for the use of formal techniques are referred to as "formal friendly" designs.

The following architectures are not suitable for formal verification: FPU's, multipliers, AXI, PCI bus protocols, GPUs, SPUs, filters, and designs that perform sophisticated algorithms The earlier efforts, however, demonstrate that formal methods may still be used to verify these designs. Since human manipulation is required to, for example, minimize the proof complexity, excellent formal verification knowledge and a high level of effort are required to complete the verification.

A semi-formal approach can be devised when the DUT is too vast to be taken into account for end-to-end formal verification. [11]Using both formal and simulation methods, a hybrid verification strategy verifies the DUT. Traditional simulation approaches will be used to verify RTL designs which are too vast and/or complicated to be taken into consideration for formal verification owing to time and/or resource constraints. To address various challenges in formal verification, a systematic approach must be taken. The obstacles include formal test planning, producing properties, confirming the accuracy of the property collection, and complexity management.

The creation of a verification plan is an essential and significant step in successfully verifying the DUV. The Design and Verification Engineers define the aims of the verification process in an industrial setting. The design's characteristics or abstract criteria that must be verified must be listed. To gather the functional coverage, these characteristics are successively mapped to the produced attributes.

The development of the property is a crucial component of the formal verification. The formal tool uses model checking to ascertain whether the design specifications satisfy the property hypothesis after capturing the properties from the Executing the formal test plan is the methodology's main component. It takes less time and is simpler to set up the setting for formal verification than simulation.

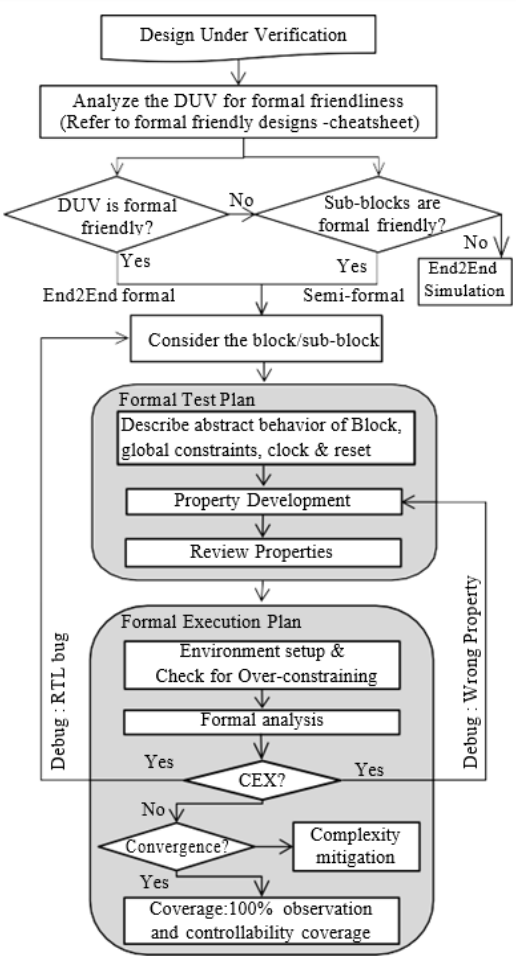

Fig -2: Formal verification methodology

Executing the formal test plan is the methodology's main component. It takes less time and is simpler to set up the setting for formal verification than simulation. This stage involves defining the DUV's reset behavior and triggering event (clock edge). Global restrictions (such as turning off scan mode I/test mode) are also defined for the DUT. After completing the first few stages, the setup has to be examined for any unintentional over constraining. This may be done by

compiling the stimuli coverage, which displays the RTL code lines that are omitted from formal analysis because of the restrictions. Examining should be done on the portion of the RTL code that was omitted by the inadvertent limiting.

In comparison to the methods described thus far, the formal analysis and debugging Counter examples need greater time and resources. Modern formal tools come with cutting-edge debugging and analysis features that may be used to significantly cut the amount of effort spent troubleshooting. It is necessary to take the necessary actions as shown in Fig. 2 after identifying the causes of the property failure.

The collection of coverage from the property runs is the last phase in the flow. The quality of the characteristics is closely correlated with the quality of the formal verification. Review the attributes and gather observable coverage are necessary to prevent trash in-garbage out. The formal tools produce structural and functional coverage data during the formal analysis. While the structural coverage is focused on the coverage of RTL lines, the functional coverage is concerned with the outcomes of property pass/fail tests. If the analysis depth is sufficient to analyze all potential design behaviors in the case of "bounded" proofs, it is still possible to retrieve the coverage information.

3. PROJECT METHODOLOGY

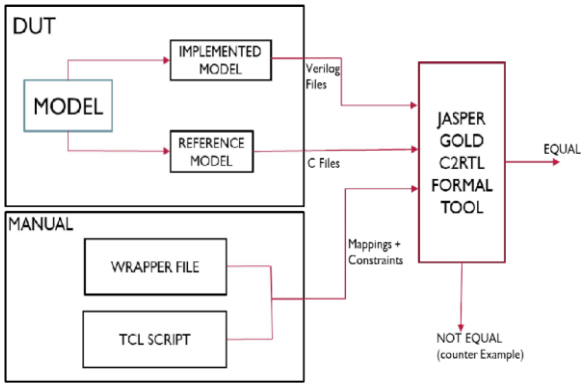

Fig -3: Project Methodology

DUT is the IP, which is under test for verification. Reference or specification model is the golden reference model, which may be written in C/C++/Sysc/Verilog language. The implemented design is usually written in Register transfer level.

Wrapper file includes all the input mappings between C and RTL model signals and TCL script consist of constraints which is written as assertions. Both DUT and manual script is fed to the tool where the tool checks for all the possible combinations of input to match the assertions. If it fails then Counterexample will be generated.

Design under test is the turbov IP which will be verified in Jasper gold C2RTL tool. Implemented model is written in RTL language and reference model is written in C language. Where both models are fed with same set of inputs and output is verified for the same.

As in C language we can't define input and output logic wrapper file is written to define the input and output signal logic for the design. In TCL script the wrapper file is called which is the reference model file and the same signals of C model and RTL model is defined. Here the assumptions which is my input to the DUT is defined and the functionality which is to be verified is written as assertions.

4. RESULTS

While synthesizing the RTL model to the jaspegold formal tool encountered with the combinational loop errors.

In order to overcome the combinational loop errors written the Verilog code SCFIFO component instantiations. In addition, written the Test script to check the functionality.

The results of the assertion pass rate before and after instantiating the Verilog model instantiation in the design is as shown.

Assertion pass rate before SCFIFO model Instantiation	
Total Tasks	2
Total properties	
Assumptions	0
Assertions -Proven -bounded_proven -CEX	96 35 (36.45%) 0 61 (63.54%)

Table1. Assertion results before SCFIFO model Instantiations

When the TCL script ran in Jasper gold formal tool before instantiating the SCFIFO modules in the design the too was black boxing the out-of-range signals. Because of that the assertion pass rate was 63.54% as shown in table 1.

Assertion pass rate after SCFIFO model Instantiation	
Total Tasks	2
Total properties	
Assumptions	0
Assertions -Proven	96 70 (80.20%)

-bounded_proven	0
-CEX	19 (19.79)

Table 2. Assertion results after SCFIFO model Instantiations

When the written Verilog code for SCFIFO models is instantiated in the design without sacrificing the functionality the assertion pass rate increased to 80.20 as shown in table 2.

5. CONCLUSIONS

In the proposed work we have presented an efficient method to verify the functionality of the design. Here we are comparing the reference design which is C model with the implemented model which is RTL model. Initially the SCFIFO modules black boxed certain signals because of that assertion pass rate was 36.45%. By developing Verilog models for such four SCFIFO instantiations the assertion pass rate increased to 80.20% without sacrificing the original functionality of the design.

REFERENCES

[1] J. Wang, J. Shao, Y. Li and J. Ding, "Survey on Formal Verification Methods for Digital IC," *2009 Fourth International Conference on Internet Computing for Science and Engineering*, 2009, pp. 164-168, doi: 10.1109/ICICSE.2009.46.

[2] A Comparison of Assertion Based Formal Verification with Coverage drove Constrained Random Simulation, Experience on a Legacy IP, Jentil Jose, Sachin A. Basheer WIPRO technologies.

[3] R. Drechsler and G. Fey, "Formal verification meets robustness checking — Techniques and challenges," *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2010, pp. 4-4, doi: 10.1109/DDECS.2010.5491833.

[4] H. Savoj, A. Mishchenko and R. Brayton, "Sequential Equivalence Checking for Clock-Gated Circuits," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 2, pp. 305-317, Feb. 2014, doi: 10.1109/TCAD.2013.2284190.

[5] K. Gracie and M. Hamon, "Turbo and Turbo-Like Codes: Principles and Applications in Telecommunications," in *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1228-1254, June 2007, doi: 10.1109/JPROC.2007.895197

[6] S. Nandan and P. P. Deepthi, "Performance Improvement of Turbo Codes Using Soft Input Decryption," 2018 Fourth International Conference on Computational Intelligence and Communication Networks, 2012, pp. 394-397, doi: 10.1109/CICN.2018.160.

[7] A. H. Saleh, K. M. Saleh, and S. Al-Azawi, "Design and simulation of CRC encoder and decoder using VHDL," 2018 1st International Scientific Conference of Engineering Sciences - 3rd Scientific Conference of Engineering Science (ISCES), 2018, pp. 221-225, doi: 10.1109/ISCES.2018.8340557.

[8] H. Macherano, A. Zinchenko and "Combinational Equivalence Checking for adder Circuits," in IEEE Transactions on Computer-Aided Design ofIntegrated Circuits and Systems, vol. 33, no. 2, pp. 305-317, Feb. 2014, doi: 10.1109/TCAD.2013.2284190.

[9] Cadence eLearning courses.

[10] "IEEE Standard for System, Software, and Hardware Verification and Validation," in IEEE Std 1012-2016 (Revision of IEEE Std 1012-2012/ Incorporates IEEE Std 1012-2016/Cor1-2017) , vol., no., pp.1-260, 29 Sept. 2017, doi:10.1109/IEEESTD.2017.805546