Coverage measurement and analysis during verification of DUT

Armen Boiadzhian¹,², Zbigniew Lisik¹, Irina Hahanova²
Lodz University of Technology, Wolczanska 211/215, Lodz, Poland
Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, Ukraine

armen.boiadzhian@edu.p.lodz.pl

Abstract. Verification involves checking the compliance of the system with the requirements of its specification at each level of detail. Given a certain complexity inherent in real systems, formal methods are not applicable for such a resister-transistor level (RTL)-description check, since they are associated with extremely high computational complexity. In practice, in this case, the device's RTL code is checked by applying test stimulus to the RTL model and controlling the response to either the specification or the reference model. The applied techniques for writing and generating tests do not have a priori guarantees covering all possible situations that allow us to identify bugs in design.

1. Introduction
In the context of constantly changing requirements and continuous improvement of projects, such characteristics of technology as the ability to reuse tests and the ability to create tests that are resistant to changes in implementation are of great importance. Even with the development of various forms of coverage and new tools that support the measurement of coatings, the use of these indicators during the verification tends to be temporary, which was mainly due to a lack of clearly established methodologies for testing based on coverage [2].

The relevance of the topic is connected to a rapid growth of electronics industry. The number of electronic devices to verify is growing immensely. Figure 1 shows the estimated growth rates for the global electronics industry from 2016 through 2018, by region. In 2018, the European electronics industry was around 2% over the previous year. We can observe the growth of 3% in Australia and 5% in Asia and America. Moreover, we can see the average growth in electronic industry in the world which was 4% in the year 2018. The variety of the devices that emerge, accelerates the development of the verification methods [4].

1.1. Code coverage
Code coverage is the measurement of structures in the source code that were activated during modeling. One limitation of the code coverage metric is that it is possible to achieve 100% code coverage during regression execution, which means that the testbench provided an input that activated all the structures in the RTL source code, but there are still defects in the project. For example, an input signal might activate a line of code containing a defect, but the testbench did not generate the additional necessary signal that propagates the effects of the defect to some point in the testbench where it could be detected. In practice, there are cases where the testbench has reached 90% code coverage - however, only 54% of
the covered code will be investigated during the simulation. This means that a defect may exist in a line of code that was designated as covered, but the defect was not detected due to insufficient input to propagate the defect to the observation point [1].

Types of code coverage:

- Line coverage
- Statement coverage
- Branch coverage
- Condition coverage
- Event coverage
- Toggle coverage
- Finite State Machine coverage
- Controlled and Observed coverage

![Figure 1. Estimated growth rates for the global electronics (Statista, 2020 [7])](image)

1.2. Functional coverage

Functional coverage is the metric of how much design functionality has been covered by the verification environment.

The purpose of the functional test is to determine whether the project requirements are functioning as defined in the specification. A functional coverage can be associated with either a project specification or an implementation coverage space. The purpose of functional coverage measurement is to measure the verification progress with respect to the functional requirements of the project. That is, the functional coverage helps us determine if all of these functional requirements have been executed and then implemented during the simulation [6][7].

1.3. Code coverage vs Functional coverage

Different varieties of code coverage metrics tell us how well the input signal went to the device under test (DUT). For example, some lines are not covered or some signals are not switched, this means that testbench and testcases are not good enough for the project to reach these states.

Code coverage cannot show dependencies between different parts of RTL code. It can only tell us if the signal has switched or if the code has not been reached.
Functional coverage points are an indicator of the coverage of the functional state of the project. Functional state can be achieved by combining different parts of the code or different signals. Thus, this metric is best for measuring the completion of the verification. But, by definition, functional coverage is highly subjective. The quality of the functional coverage report is not worse than the performance of the functional coverage and its implementation [3].

For any coverage metric to have any value, it must be coupled with a good validation mechanism for all test cases. There is no point in reaching the design state or breaking the logic and not checking if the project is as expected in that state. See the comparison of the two methods mentioned above in Table 1.

**Table 1. Comparison of code coverage and functional coverage.**

| Code Coverage                                      | Functional Coverage                                      |
|---------------------------------------------------|----------------------------------------------------------|
| Tells how well HDL code has been exercised by the test bench | Measures how well the functionality of the design has been covered by the test bench. User defines the functionality to be measured through coverage |
| Verifies completeness of verification environment in terms of hitting expression lines etc. of RTL code | Verifies completeness of verification environment as per the requirement specification, and also functional coverage points |
| Design specification is not used                  | Design specification is used                              |
| Support in all languages                          | Verilog does not support functional coverage. To perform functional coverage, SystemVerilog, Specman E or Vera are necessary |

2. Materials and methods
The object of the research is the methods for creating an analysis of the completeness of testing when verifying digital devices, for finding holes in the verification environment and for further improving quality control. Verification environment and methods for analysing the completeness of testing in the verification of digital devices are the subject of research. RISC processor is taken as DUT (device under test), and coverage methods of verification are used as the method.

The functional behavior of any construct observed from any verification interface environment consists of both data and temporal components. So, with a high abstraction level, there are two main types of functional coverage measurement: cover properties and cover groups.

2.1. Cover groups
Cover groups record the number of occurrences of different values specified as coverage points. These coverage points can be referenced hierarchically using test cases or a testbench so that specific values or scenarios can be queried. They also provide a means to create cross-coverage. In contrast to the properties of the coverage, the coverage group can be used in class objects and structural code [1].

2.2. Cover properties
Cover properties use the same syntax that is associated with assertions in SystemVerilog. This syntax is used by both properties and sequences that can be used by SystemVerilog statements, namely, assert, assume, or cover. The advantage of this is that it takes the same amount of effort to ensure that the same properties can be used for both assertions and functional coverage for collection. Unfortunately, cover properties can only be placed in structure code (that is, modules, programs, or interfaces) and cannot be used in class-based objects [1].
2.3. **DUT - device under test**

The DUT for verification was reduced instruction set computer (RISC) processor, that stands for reduced instruction set computer processor, which architecture you can see in the Figure 2. During the verification different modules of the processor were examined.

![Figure 2. The structure of the device](image)

3. **Results**

Two interfaces are used to connect to the device under test and monitor the signal values. The first interface is also used by the verification environment to submit test cases. The second interface is
connected only in the monitor for monitoring the inputs and outputs of the control finite-state machine (FMS).

As a result, a functional coverage was developed on the base of the created verification environment for the processor with RISC architecture, which gives an idea and assessment of the coverage measurement during verification. Based on the conducted six tests, the coverage measurement of verification of the processor modules was carried out, the result of the analysis is presented in Figure 3. Blank spaces (coverage register data base and coverage flags) show uncovered points, the values which were not verified or invalid.

![Figure 3. Results of coverage analysis](image)

**Figure 3.** Results of coverage analysis

### 4. Discussion

As shown in Figure 4, finding bugs in the early stages of development greatly reduces the cost of their elimination. Thus, this work shows how valuable it is to carry out verification for the timely detection of bugs and their elimination, which in turn reduces the cost of development.

![Figure 4. Cost of fixing a problem in different periods of development](image)

**Figure 4.** Cost of fixing a problem in different periods of development [2]
5. Conclusions

As a result of the research, the functional coverage was investigated on the base of the verification environment created for the processor with RISC architecture, which gives an idea and assessment of the completeness of testing during verification.

During the development, the literature sources were analyzed, in particular, the methods of building a verification environment based on components, the correct use of Universal Verification Methodology (UVM) libraries were studied, a wide review with regard to the development of various verification environments was conducted, methods for evaluating the completeness of testing were studied and analyzed. As a result, due to the fact that the code coverage is automatic - at the level of the verification environment - and does not give an idea of the tested functionality, which is one of the key drawbacks, as the complexity of a large project size, functional coverage was used, due to its assumed advantages.

For this purpose, the hardware modules of matrix execution of processor instructions were studied, which enabled the simultaneous execution of four logical operations. Verification environment was implemented and the test was built on the basis of the processor. Then the completeness of testing during processor verification and its subsequent evaluation were assessed. The practical implementation of models and verification methods was integrated into the EDAPlayground modeling environment.

After analysing different verification methods we have proved the flexibility advantage of the functional verification method based on the cover points, as we had to examine six different modules of the processor. In conclusion, the method, indeed, helped us to find uncovered spaces in our verification process.

Acknowledgements

I wish to acknowledge assistance and encouragement from senior colleagues of Kharkiv National University of Radioelectronics, namely The Design Automation Department, and Lodz University of Technology, especially The Department of Semiconductor and Optoelectronics Devices. I would also like to express gratitude and appreciation for the editor of the above article.

References

[1] Piziali A 2004 Functional Verification Coverage Measurement And Analysis (Boston: Kluwer Academic Publishers) pp 153-204

[2] Spear C 2006 SystemVerilog for Verification: A Guide to Learning the Testbench Language Features (Marlboro: Springer) p 55

[3] Mayer A 2003 Principles Of Functional Verification (Boston: Elsevier Science) p 98

[4] Wilcox P 2004 Professional Verification: A Guide To Advanced Functional Verification (Boston: Kluwer Academic Publishers) pp 45-46

[5] Hahanov V, Hahanova I, Litvinova E and Guz O 2010 Design And Verification Of Digital Systems on Chips (Kharkiv: Novoye Slovo) p 55

[6] Hahanov V, Litvinova E and Guz O 2011 Logic associative computer (Kharkiv: Electronic simulation) pp 73-80

[7] Statista 2020 Global electronics industry: growth outlook by region 2018 | Statista [online] Available at: https://www.statista.com/statistics/268396/estimated-growth-rates-for-the-electronics-industry-by-region/ [Accessed 27 Feb. 2020]