International Journal of New Technology and Research (IJNTR)
ISSN:2454-4116, Volume-4, Issue-2, February 2018 Pages 122-124

Verification of Identity and Syntax Check of Verilog and LEF Files

D. A. Simonyan

***Abstract*— The Verilog and LEF files are units of the digital design flow [1][2]. They are being developed in different stages. Before the development of the LEF file, the Verilog file passes through numerous steps during which partial losses of information are possible. The identity check allows to make sure that during the flow the information has not been lost. The syntax accuracy of the Verilog and LEF files is checked as well.**

The scripting language Perl is selected for the program. The language is flexible to work with text files [3].

The method developed in the present paper is substantial as the application of integrated circuits today is actual in different scientific, technical and many other spheres which gradually finds wider application bringing about large demand.

***Index Terms*— LEF vs Verilog files validation, digital design flow, VLSI, data parsing, perl scripting, results reporting.**

I. Introduction

A. The description of the method.

The goal of the proposed method is to create a .pl format script file through which, it will be possible to check the identity of the Library exchange format files, check the syntax peculiarities of the Verilog and Library exchange format files. The script should be able to check most of the important characteristics of the files. The program should develop a report file where the results of the check will be marked.

The program will include four parts:

- Reading and checking the Verilog file;
- Reading and checking the LEF file;
- Checking identity of the Verilog and LEF files;
- Creating the report file.

In the Verilog file will be checked the presence of the block name after the module keyword [4]. Each pin should have one of the following directions input, output, inout or wire. The endmodule condition should be present in the file.

In the LEF file, the LEF version will be checked, as well as the presence of the definition of the “SYMMETRY” condition in macro, the presence of the definition of the “FOREIGN” condition in macro, the presence of the definition of the “SIZE” condition in macro, the presence of the definition of the “DIRECTION” condition in macro, the presence of the definition of the “USE” condition in macro, the presence of the definition “End Pin Name” at the end of each block, the presence of the definition “END MACRO_NAME” at the end of macro, the presence of the definition “END LIBRARY” at the end of the LEF file.

The User should have an opportunity to obtain the results a few seconds after using the script.

It is necessary to develop a system which will store the report data in the textual file to be read in future.

II. The Relationship of the Verilog and LEF files

The process of automated digital design is rather comprehensive. In the process of the digital automated design flow, many tools are used, among them the behavior description language. One of the behavior description languages known to us is the Verilog which is an initial description of the integrated circuit by the RTL language. The information on the design pins is stored in Verilog file [5].

In the automated digital design process, obtaining the Verilog description is followed by the operation with the LEF (Library Exchange Format) files of the integrated circuit. Before the development of the LEF file, the Verilog file passes many steps at which the partial losses of information become possible. The check of the identity allows to make sure that during the flow, the information has not been lost. The syntax accuracy of the Verilog and LEF files is also checked.

LEF is one of the formats of presenting chips which is submitted to the customer by the organization, producing chips in a final package which also includes the GDSIL, Verilog and a number of other data. An important problem is the checking of the package quality and completeness, which is the correspondence of each type of the package data. With all other data, the process of identity checking based on the requirements set to those data. That is, it is necessary to check the identity of the contents of the LEF and Verilog files with the automatic software tools.

D. A. Simonyan, IIAP NAS RA, Yerevan, Armenia

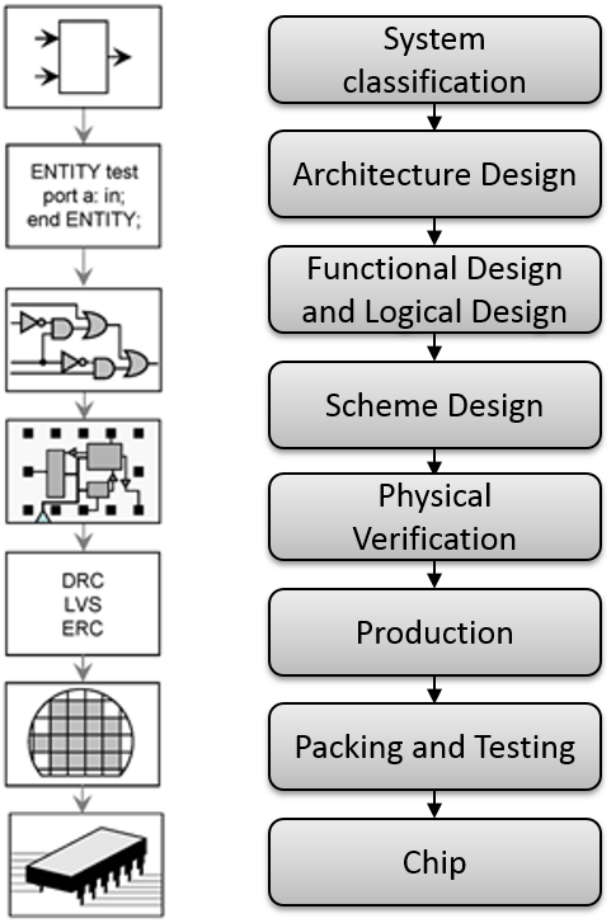

Fig. 1 Physical design flow

The physical design process for special purpose IC is shown in Fig. 1. During the process, the RTL initial up to GDSII files are obtained after which the chip is sent to be produced.

In Fig.2, the view of the program solution of the method by blocks is introduced

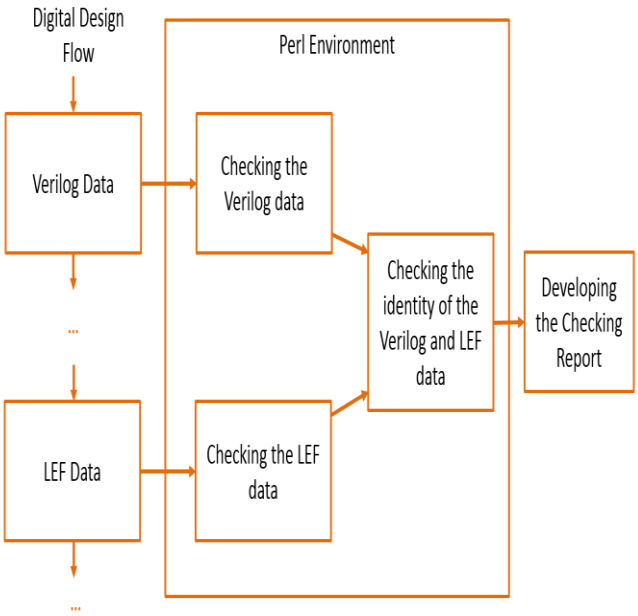

Fig. 2 The graphical imaging of the program

As a checking program, the scripting language Perl has been selected which is able to work with text files.

The addresses of 2 input files are given to the program. The script reads each file and assigns its contents to the corresponding variables. After reading the Verilog file, a number of important checks are carried out. Then the program reads the LEF file, carries out the corresponding checks. It is also important to check the correspondence of the information on the pins. It is carried out by locating the information on pins from the Verilog and LEF files in appropriate arrays and checking the identity of the arrays.

Then, in the selected address, the report file is created in which information on checking is presented.

The checking carried out in the program. The syntax checking of the LEF and Verilog file. To read the files in the Perl language, the "open" function is applied which allows to obtain an array from the Verilog file in the language Perl. After opening the Verilog file certain conditions are checked.

The hierarchic modeling principle in Verilog is carried out by modules. The module is the main component of the Verilog language, from which the design is constructed. In Verilog it is named by the keyword module. The module description should end by the appropriate keyword "endmodule".

Each module should have a name and a list of terminals, describing the inputs and outputs of the module.

The program carries out the following important checks in the Verilog file. It checks:

- the presence of the block name after the keyword module;
- the fact if all the pins have one of the mentioned directions - input, output, inout, or wire;
- the presence of the condition, "endmodule", meaning the end of the module.

The LEF file is also opened by the program by means of the function "open" of the Perl language. The contents of the LEF file is filled into the array which allows to carry out checks in each line of the file. After opening the file and assigning it to the array certain conditions are checked.

The LEF file contains the Library information on the design class.

The program accomplishes the following checks in the LEF file. It checks:

- the LEF version;
- the presence of the Symmetry Condition definition in macro;
- the presence of the Foreign Condition definition in macro;
- the presence of the Size Condition definition in macro;
- the presence of the Direction Condition definition in macro;
- the presence of the Use Condition definition in macro;
- the presence of the "End Pin_ Name" definition at the end of the block of each pin;
- the presence of the "End Macro_Name" definition at the end of macro;
- the presence of the "End Library" definition at the end of the LEF file.

The result of each checking is ascribed to the "hash" variable which allows to quickly achieve the checking results.

Let us consider the checking of the identity of the LEF and Verilog file pins. The program is checking the identity of all

pins. All the pins inside the module on the Verilog file read are stored in the array, to compare with the described pins in the LEF file later.

Likewise, the pins described in the LEF file are stored in the array. A two-sided checking is carried out:

- all the pins described in Verilog should be present in the LEF file;
- all the pins described in the LEF file should be present in the Verilog file.

In case of an error, the program is able to record the missing pins in special variables which will be accessible when developing the report results.

Now let us speak about the report development. The script collects the checking results in the variables. In the last step of the program, a file in which the checking results are mentioned, is created. It gives information of the checking result is positive or negative.

It is mentioned in the report that if there is a problem, where it is which allows the user to open the mentioned file and find it easily.

III. Conclusion.

A program is developed which automatically checks: the Verilog file:

- the presence of the block name after the keyword module;
- any pin should have one of the mentioned directions; input, output, inout or wire;
- the presence of the endmodule condition in the file.

The “Library Exchange Format” file:

- the LEF version;
- the presence of the Symmetry Condition definition in macro;
- the presence of the Foreign Condition definition in macro;
- the presence of the Size Condition definition in macro;
- the presence of the Direction Condition definition in macro;
- the presence of the Use Condition definition in macro;
- the presence of the “End Pin_ Name” definition at the end of the block of each pin;
- the presence of the “End Macro_Name” definition at the end of macro;
- the presence of the “End Library” definition at the end of the LEF file.

The identity of the Verilog and LEF files:

- The description of the pins in the Verilog file should be present in the LEF files. The Verilog and LEF files and the data identity is also checked.

So, the method described method checks the presence of the syntax peculiarities of the Verilog and LEF files as well as the identity and transfers the information to the user in the form of readable report in the file with a textual format.

References

[1] M. Gordon, “The semantic challenge of Verilog HDL”, In Tenth Annual IEEE Symposium on Logic in Computer Science, 1995. LICS'95 proceedings, June 1995, pp. 136 – 145

[2] K.D. Muller-Glaser, K. Kirsch, K. Neusinger, "Estimating essential design characteristics to support project planning for ASIC design management", In IEEE International Conference on Computer-Aided Design. ICCAD-91. Digest of Technical Papers., pp. 148 - 151

[3] D. Widhalm, S. Tauner, M. Horauer, "Augmenting pre-silicon simulation by embedding a scripting language in a SystemC environment", In IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), 2016, pp. 1-6

[4] D.R. Smith, "Hardware synthesis from encapsulated Verilog modules", In Proceedings of International Conference on Application Specific Systems, Architectures and Processors, 1996. ASAP 96, 1996, pp. 284 - 292

[5] Tsu-Hua Wang, Chong Guan Tan, "Practical code coverage for Verilog", In IEEE International Verilog HDL Conference, 1995, pp 99 – 104

[6] S. Darfeuille, "PERL scripts-based netlist analysis tool for the detection of ESD ‘big buffer’ configurations", In Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012, pp. 269 – 272

[7] H. Fatih Ugurdag, "Experiences on the road from EDA developer to designer to educator", In East-West Design & Test Symposium, 2013, pp. 1 - 4