Synthesis of a VHDL-model of gas discharge circuit of gas laser

Vera Golyan
Software Engineering Department
Kharkiv National University of Radioelectronics
Kharkiv, Ukraine
vira.golan@nure.ua

Nataliia Golyan
Software Engineering Department
Kharkiv National University of Radioelectronics
Kharkiv, Ukraine
nata2012.nn@gmail.com

Kyrylo Halchenko
Information and Network Engineering Department
Kharkiv National University of Radioelectronics
Kharkiv, Ukraine
kyrylo.halchenko@nure.ua

Abstract— **The paper suggests synthesis of a VHDL – model of a gas laser installation, efficiently distributing RTL-circuits to create a new list of connections with a minimum set of components. Models of GDC lasers consist of three subtasks: a logic conclusion, optimization received RTL structures and its distribution on structural elements technology mapping.**

Keywords— VHDL- models, LSIC, CAD, SPARTAN-II, LAZER, GDC, RTL

I. INTRODUCTION

Computer-aided designs use the conducts of the designed objects for prognostication. The use of design for the construction of the laser settings is today more actual in connection with all by the greater necessity of the computer-aided manufacturing. For example, descriptions of the modern laser setting can be certain and predicted design facilities. On models execute the controlled experiments in those cases, when experimentation on the real objects not possible

II. THE USE OF MODELS OF GDC LASERS FOR THE CONSTRUCTION OF THE LASER SETTINGS

The use of design for the construction of the laser settings is today more actual in connection with all by the necessity of the computer-aided manufacturing. Computer-aided designs use the conducts of the designed objects for prognostication. For example, descriptions of the modern laser setting can be certain and predicted design facilities. Designing long scale integrations (LSIC) is impossible without the availability of powerful CADs. Developing CAD means of a gas laser gas-discharge circuit is critical problem [1]. Modern CAD have is functional-block structure, in accordance with which different stages of a design cycle are performed by corresponding subsystems. This has led to the appearance of individual systems frequently developed by a variety of manufacturers majoring in the , performance only one of stages (for example, a synthesis stage). In addition, CAD is becoming a through one: all the design stages – microcircuits modeling synthesis, implementation and programming are carried out in a common medium. It permits to verity devices at all the design stages, whish enables to reduce the probability of emergence of errors. Therefore at a definite stage of CAD development there appeared a need for developing standardized - languages of equipment description, one of which is VHDL [2]. VHDL is a powerful language which allows to describe the behavior of digital circuits, as well as to perform and also to spend the hierarchical functional-structural description of the big integrated systems and at the same time has all signs of the programming language of high level – allows to create the types of the data, has a wide set of arithmetic and logic operations [3].

A. Purpose and possibilities of synthesis GDC a VHDL-model

Problem of the synthesis of system is effective distribution of the RTL-circuit with the aim of creating a new list of connections with a minimum number of circuit components used. Each component of the new list will correspond to the physical hardware block in the FPGA (elements of configured logical blocks, anticipated carry logic).

Hierarchical designs are synthesized in an ascending regime when components of the lower level are synthesized up to those of a upper level.

The devise model in the language of describing VHDL equipment, must be adapted for synthesis and implementation on FPGA chip of the XILINX firm.

978-1-7281-1814-7/19/31.00 © 2019 *IEEE*

Параметры семейства FPGA Spartan-II

Logic Cells	System Gates (Logic and RAM)	CLB Array (CxR)	Total CLBs	Maximum Available User I/O	Total Distributed RAM Bits	Total Block RAM Bits
972	30000	12x18	216	132	13824	24K

Fig. 1. FPGA Spartan-II parametres

The problem of a choosing on hardware platform is of great important for the designer.

The right choice will enable:

- to reduce material cost in the implementation of the device
- to attain optimum funcharing and speed of operation

According to the technical requirement the GDC model must be adapted for synthesis and implementation on the FPGA of the Xilinx firm. In choosing FPLD of most of the focus is on the relation between its cost and productivity. Besides it is necessary to take into account the crystal are which will be occupied by the synthesized device.

Structure of Spartan-II chip

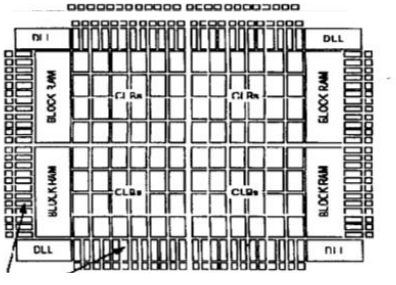

Fig. 2. Matrix Spartan-II

FPGA (Field Programmable Gate Arrays) were first developed by the Xilinx firm in 1985.

Tuning a FPGA on the specified functioning is carried out each time before the start of its operation. The required setup program is preliminary recorded in ROM (RAM). Loading information from ROM and FPGA automated initialization after turning on power supply (for this the FPGA contains the required logic circuits). One can also carry out FPGA tuning under the control of a microprocessor or microcontroller.

Family FPGA Spartan-II has a record low cost per a gate on the density of packing up to 200 thousand gates. In a crystal there is four blocks of the RAM each having 4KBits, besides it in possible to implement 16 bits of memory on each 4 input functional generator . The Spartan-II generator combine lines of flexible regular architecture which include a CLB matrix surrounded by programmable input – output blocks inter connected by a hierarchy of high – speed, multi – sided resources by a history of high-speed, multi – sided resources of interconnections.

Devices Spartan-II (fig. 3) have more high efficiency in comparison with the previous families FPGA. Projects can work with system frequency of synchronization to 200 MHz, including input/conclusion blocks (Input/Output-I/O).Besides chips Spartan-II are distinguished by variety of advantages:

- Rather low cost of a crystal;
- The big dimension of the chip (to 200 000 system gates);
- High speed.

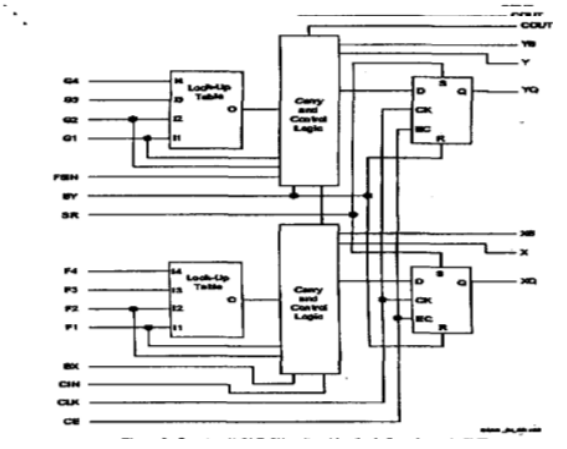

Fig. 3. Devices Spartan-II

B. Flow-chart of adjusting of mixture, renewals pressures are in GRK of lazer

Programmed by the user valve matrix Spartan-II shown on fig. 2, covers: configured logic blocks (configurable logic blocks - CLBs) and input-output blocks (IOBs). CLB blocks serve for creation of functional logic elements, and blocks I/O create the interface between contacts of a microcircuit and CLB blocks.

The adjusted logic block (CLB). Base building element CLB of the block is the logic cell (Logic Cell - LC). The logic cell includes 4-vhodovoj the functional generator, logic of the accelerated carrying over and remembering element. The exit of each functional generator in each logic cell is connected to a target line of the CLB-block and to a trigger D-input.

Each CLB-block contains four logic cells organized in the form of two identical sectors (Slice) in series Spartan-II. In drawing 3 one sector is represented in more details. In addition to four base logic cells, the CLB-block of series Spartan-II contains logic who allows to combine resources of functional generators for realization of functions from five or six inputs. Look-Up Tables (LUT). Functional generators are realized in the form of 4 Look-Up functional tables (LUT). Except use as the functional generators, each LUT-element can be used also as synchronous memory of type RAM dimension 16x1 bit. Moreover, from two LUT-elements within the limits of one

sector (Slice) it is possible to realize synchronous RAM-memory dimension 16x2 a bat or 32x1

C. Choice of tool means

For project synthesis the software product of firm Synplicity Synplify 7,0 as it has following advantages in comparison with programs of synthesis of other manufacturers is chosen:

- High speed of synthesis;
- Visual representation of results of synthesis (netlista);
- Presence of libraries under modern element base of the largest world manufacturers.

The analyzed project is synthesized in a library set of primitive things.

The list of connections after a stage of a logic conclusion is made, basically, from abstract logic elements, such as adders, counters, multipliers, and also gates and synchronous triggers. These circuit elements further it is necessary to place COTTON VELVET in structural components of used technology. This process also is called as technological distribution.

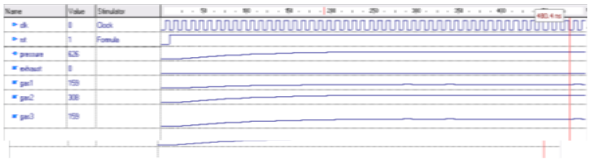

Fig. 4. Result of modeling.

To be convinced that the system corresponds to the specification, it is necessary to verify it , i.e. modeling.

III. TESTING

Software testing is a process of research, testing of a software product aimed at checking the correspondence between the actual behavior of a program and its expected behavior on a final set of tests selected in a certain way (ISO / IEC TR 19759:2015) [4].

Software testing is the process of technical research assigned to identify information about the quality of a product relative to the context in which it is to be used. The testing technique includes the process of finding errors or other defects, as well as testing the components of the software.

Testing techniques also include the process of finding errors or other defects and testing software components for evaluation purposes. Evaluating of next parameters available:

- compliance with the requirements that guided the designers and developers;
- correct answer for all possible inputs;
- performance of functions within a reasonable time;
- usability,
- compatibility with software and operating systems,
- compliance with the customer's tasks.
- since the number of possible tests even for simple software components is almost infinite, therefore, the testing strategy is to conduct all possible tests taking into account the available time and resources, as well as covering the main part of the system functionality with tests. As a result, the software tested by standard execution of the program in order to identify defects, so for testing the system developed in this work, a method of manual, that is, manual testing was chosen.

Since the number of possible tests even for simple software components is almost infinite, therefore, the testing strategy is to conduct all possible tests taking into account the available time and resources. As a result, the software tested by the standard execution of the program to detect bugs (errors or other defects).

Software testing can provide objective, independent information about the quality of the SOFTWARE, the risks of failure, both for users and for customers.

Manual testing of the user interface was carried out to detect errors in the interface functionality, unhandled exceptions when interacting with the interface, loss or distortion of data transmitted through the interface elements, as well as errors in the interface, such as the absence of elements.

Manual testing is a part of the testing process at the stage of quality control in the software development process. It is carried out by the tester without the use of software, to check the program or website by modeling user actions. Ordinary users can also act as test subjects, informing developers about the found errors.

In addition to manual testing, the logic of the model covering by integration tests written with the help of the library test library. Initially, it was planned to test using Boost::Test , but this library was heavy enough for such a project, then it was decided to develop a light and small library for testing with which you can test some parts of the application.

Automated testing is used to reduce testing time and cost. The computer can perform a recorded sequence of steps faster than a human can. Also, run tests overnight to get results in the morning. However, the labor saved on performing automated tests should be spent on writing test programs. Depending on the type of applications to be tested and the automation tools chosen, this may require more workers than a manual approach. In addition, some test tools produce a very large amount of data, potentially creating results interpretation tasks that can take a long time.

Drivers and program libraries must be tested with test programs. In addition, testing a large number of users (performance testing and load testing) is usually done by software rather than by hand.

During testing, BDD technology was used, which tested the behavior of some individual parts of the program, for example,

a physical model. The system was tested in four different environments, namely Windows 32/64 bit, Linux32/64 bit. Example of integrational tests, tests the model and are responsible for solving the system shown on figure 5.

```
DECLARE_OOP_TEST( SimpleSystemOfEquations)
{
        Matrix m( 3, 3,
                {
                        3, 2, 5,
                        2, -1, 3,
                        1, 2, -1
                }
        );


        Matrix::Row freeMembers( { -1, 13, 9 } );

        Gauss resolver( m, freeMembers );

        Matrix::Row roots{ 13, -5, -6 };

        assert( roots == resolver.getRoots() );

}
DECLARE_OOP_TEST( SystemNotResolved )
{

        Matrix m( 3, 3,
                {
                        0,      1,      1,
                        1,      -1,     0,
                        3,      -1,     2
                }
        );

        Matrix::Row freeMembers({ 2, -2, 2 });

        Gauss resolver(m, freeMembers);

        assert(resolver.resolveSystem().empty());

}
```

Fig. 5. – Sample tests

The test results are shown in table 1.

TABLE I. RESULTS OF SOFTWARE TESTING

Testing Stage ID	Quantity of defects	Quantity of fixed defects	Description
1	2	2	Detected and corrected cause of defect in the calculation of the physical model of the product.
2	3	3	Interface fix.
3	1	1	Discovered and fixed a bug not correctly display divergency graph data.

The table shows that there were three stages of testing, during each of which certain defects were found and corrected. This allowed us to check:

- the correspondence between the actual behavior of the program and its expected behavior on the final set of tests performed in a certain way,
- to identify situations in which the behavior of the program is incorrect, undesirable or does not meet the specification,
- to perform a process containing all the life cycle activities, both dynamic and static, relating to the planning, preparation and evaluation of the software product and the related results of work in order to determine compliance with the described requirements, namely,
- to demonstrate that they are suitable for the stated purposes and to determine defects.

IV. CONCLUSION

The possibilities of the syntheses VHDL - models of GDC lasers, which will consist of three subtasks: a logic conclusion, optimization received RTL structures and its distribution on structural elements technology mapping.

The analyzed project is synthesized in a library set of primitive things.

The list of connections after a stage of a logic conclusion is made, basically, from abstract logic elements, such as adders, counters, multipliers, and also gates and synchronous triggers. These circuit elements further it is necessary to place COTTON VELVET in structural components of used technology. This process also is called as technological distribution.

REFERENCES

[1] Orazio Zvelto, Principles of lasers, vol. 3. Moscow: Mir, 1990.

[2] Piotr Bibilo, Modeling and verification of digital systems in VHDL. 2012.

[3] J. R. Armstrong, Modeling of digital systems in VHDL language. Moscow: Mir, 1992.

[4] "ISO/IEC TR 19759:2015", ISO, 2019. [Online]. Available: https://www.iso.org/standard/67604.html.