Federal State Autonomous Educational Institution of Higher Education

**«NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS»**
**TIKHONOV MOSCOW INSTITUTE OF ELECTRONICS AND MATHEMATICS**

Nikita, Indyuchenko

БПМ 212

# PROJECT REPORT

Development of a unified system for storing and processing digital circuits of the dataset

PROJECT SUPERVISOR: Vladimir Zunin

MOSCOW, 2024

## Summary

**Project goal:** development of a unified system for storing and processing a dataset of digital circuits.
**Project tasks:** full coverage of the available software with unit tests;
visualization of datasets in the form of graphs;
documenting the source code;
optimization of the construction of the current graph and circuit.

## Planned results

**Project:** Software that takes as input a set of data and type of generation algorithm, on the basis of which the generation of combinational circuits takes place.
**Scientific:** Participation in conferences, preparation of scientific articles.

## Project overview

Combinational circuits are circuits whose outputs are uniquely defined by a combination of input signals. They can be built using only logic elements and do not require memory. Combinational circuits are used ubiquitously in many different digital devices, and their reliability is an important factor in the design of digital systems.

Currently, any combination scheme needs to be evaluated for operability. The main problem is that estimation methods take quite a long time to calculate the result for circuit analysis. Therefore, it is necessary to accelerate the current methods and supplement them with new ones that will gain in time and at the same time the loss in accuracy of the result will be minimal. In addition, the user should be able to use existing data evaluate the reliability performance of combinational circuits. At the same time, most analytical methods are either time-consuming to estimate reliability indicators or have low accuracy of this estimation.

It is hypothesized that the use of artificial intelligence techniques to assess the reliability of combinational circuits will make it possible to quickly and accurately assess the reliability of combinational circuits. But for this purpose it is necessary to create a dataset with combinational circuits, their parameters and reliability indices calculated by known analytical methods. To create a dataset it is necessary software that allows to generate combinational schemes close to the real ones with the use of various methods of generation, as well as the ability to calculate the parameters of combinational schemes and their reliability indices. At the same time, the dataset should provide full coverage of all possible variants of combinational schemes, as it is very important for ensuring high accuracy of the neural network trained on it.

## Proposed solution

The first task was to get rid of the dependence of the current program on the operating system. The solution to this problem was to rewrite the code from C# to C++.

The next step was to write a method for evaluating reliability, such as "Single-valve observability". Description of the algorithm:

1. counting the number of valves in the circuit
2. fixing of one valve
3. calculating the probability of the entire circuit, assuming that an error occurred on one valve, and the rest are ideal
4. follow steps 2 and 3 for the next valve
5. summation of all found probabilities and evaluation of the scheme's operability

The third task was to create a connection between the user, the generator and the cloud storage. The main points for implementation:

1. creating a web page and interface using JavaScript
2. sending data to the generator
3. saving the generated dataset to Synology drive
4. sending the URL to the web page

The fourth task was to write documentation, for this it was necessary to take several steps:

1. studying the design guide code comment
2. creating a Doxygen file that will store the entire config for the project
3. commenting on header files
4. creating a Cmake file that will generate documentation when building a project based on the configurator file and source code

The last task was to test the program for performance. Several checks have been performed:

1. creating Unit tests that, with the correct circuit parameters, the generator does its job
2. writing conditions on the web part, checking for data formats before sending to the generator
3. creating a "feedback loop" between the user and the generator. The appearance of the loading line
4. Setting a limit on the number of requests. Protection against DDOS attacks

**Conclusion and results**

The result of the project was the creation of a web interface for the ability to set the necessary parameters for generating schemes remotely, and a description of the program structure using the Doxygen program was also added.

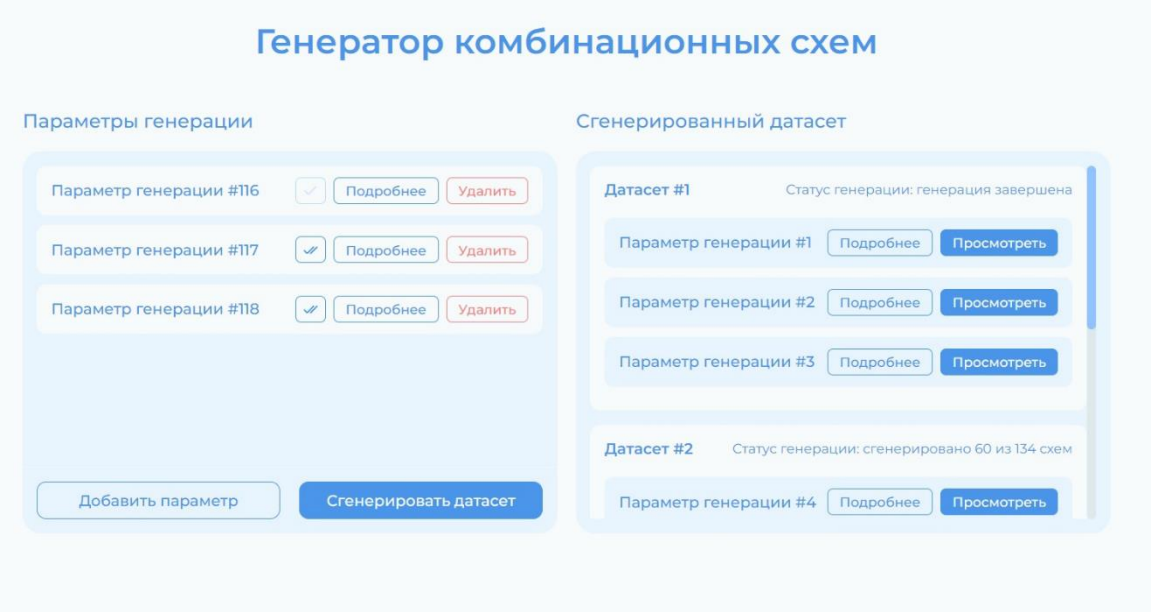


Figure 1: the user's home page for filling in data

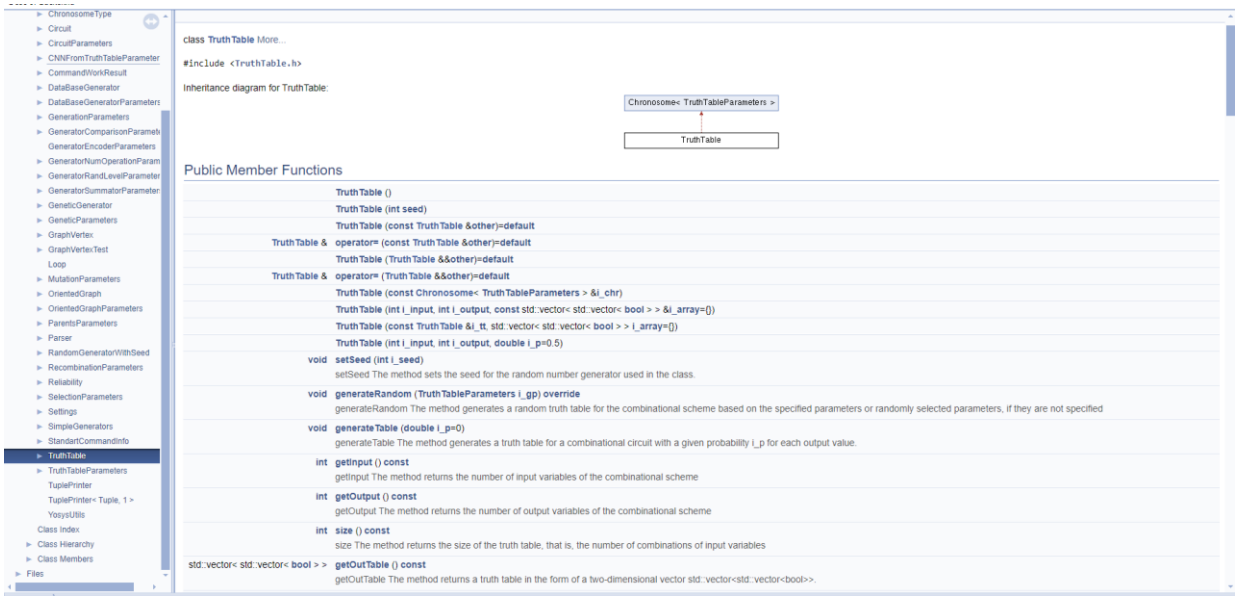


Figure 2: Description of parameters and methods of the TruthTable class

In addition, the code has been redesigned, and now the project is cross-platform, which helps the developer to run the generator on the most popular operating systems such as Windows, Linux, macOS.

There was also an attempt to translate graph entities into XML format for the ability to train data on existing libraries. However, this idea has not achieved a successful result, so this task remains relevant for the future.

**References**

[1] Borkar S. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. IEEE Micro

2005;25(6):10–6.

[2] https://doi.org/10.1016/j.microrel.2010.07.154

[3] V.V. Zunin, A. Yu. Romanov, R.A. Solovyev. Developing Methods for Combinational Circuit Generation

[4] Combinational circuits [Electronic resource] URL:https://github.com/RomeoMe5/CAD_Combinational_Circuits

[5] L. Fan and C. Wu, "FPGA technology mapping with adaptive gate decompostion", ACM/SIGDA FPGA International Symposium on FPGAs, 2023.

[6] Han J et al. Reliability evaluation of logic circuits using probabilistic gate models. Microelectron Reliab (2010), doi:10.1016/j.microrel.2010.07.154