2022 International Russian Automation Conference (RusAutoCon)

Developing Methods for Combinational Circuit Generation

2022 International Russian Automation Conference (RusAutoCon) | 978-1-6654-6655-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/RusAutoCon54946.2022.9896390

V.V. Zunin
HSE University
Moscow, Russia
vzunin@hse.ru

A.Yu. Romanov
HSE University
Moscow, Russia
a.romanov@hse.ru

R.A. Solovyev
Institute for Design Problems in Microelectronics of Russian Academy of Sciences
Moscow, Russia
roman.solovyev.zf@gmail.com

***Abstract*—This article provides an overview of the current state of research in the field of reliability of combinational circuits. The developed methods for combinational circuits generation and their subsequent implementation in the form of software are described. The implemented algorithms were used to generate combinational circuits in the structured Verilog format; the circuits generated were converted to circuits using the Nangate open standard cell library. For schemes, their parameters were calculated and stored in a structured form for subsequent conversion into a dataset in csv format. Using the developed software, a dataset of combinational circuits was generated; it can be used to conduct research on predicting the reliability of combinational circuits based on limited information about them. The dataset generated was tested using two machine learning methods.**

Keywords—combinational circuits dataset, reliability, soft errors, digital synthesis.

I. Introduction

Combination circuits are circuits whose outputs are uniquely determined by a combination of input signals [1], [2]. They can be built using only logic elements and do not require memory. Combination circuits are ubiquitous in many different digital devices, and their reliability is an important factor in the design of digital systems [3].

At present, the traditional methods for ensuring the reliability of combinational circuits are multiple redundancy methods, but other methods are developed as well [4]. To select an appropriate and sufficient protection scheme against single failures, it is necessary to evaluate the reliability indicators of the generated combinational circuit [5], [6]. This means that it is necessary to develop a set of methods and software for fast and accurate assessment of the reliability indicators of combinational circuits, which determines the relevance of this work. At the same time, most analytical methods require either a lot of time to assess reliability indicators [7], or have a low accuracy of this estimate.

On the basis of the conducted research, a hypothesis was put forward that the use of artificial intelligence methods for assessing the reliability of combinational circuits would allow quickly (and with high accuracy) to assess the reliability of combinational circuits; and the operation of the neural network would not depend on the size of the combinational circuit itself, providing an additional performance boost when analyzing combinational circuits with a large number of inputs, outputs, and logic elements. But in order to develop methods for assessing the reliability of circuits using artificial intelligence methods, it is necessary to create a dataset with combinational circuits, their parameters and reliability indicators calculated by known and justified methods. To create a dataset, one need the software that allows generating combinational circuits close to the real ones using various generation methods, as well as the ability to calculate the parameters of combinational circuits and their reliability indicators. At the same time, the dataset should provide the completeness of coverage of all possible variants of combinational circuits, since this is very important for ensuring the high accuracy of the neural network trained on it. In addition, the generated dataset can be expanded in the future in order to increase the coverage of possible tasks to be solved, including the development of methods for optimizing combinational circuits [8].

II. Review of Benchmarks of the Combination Schemes

The main sets of benchmarks for combinational circuits, presented in the Verilog format, are as follows: ISCAS85 [9], an extended version of ISCAS85 – ISCAS89 [10], LGSynth89 [11], LGSynth91 [12], IWLS 2005 [13].

The ISCAS85 benchmark contains 11 different circuits in Verilog format. This set has a larger variation in the size of circuits and a very small number of them. Its extended version, the ISCAS89 benchmark, contains 31 combinational circuits. Both presented benchmarks, in addition to description of the circuits in Verilog and other formats, contain the main parameters of the circuit (the number of inputs, outputs, logic elements by type, etc.). This allows using benchmarks without the need to additionally calculate their main parameters.

The LGSynth89 benchmark consists of 83 circuits, which can be divided into three types: state machines (41 circuits), two-level (24 circuits), and multi-level (18 circuits). Its next version, the LGSynth91 benchmark, significantly expands the previous version of the circuit set (up to 205), not only by increasing the number of circuits but also by adding a new type of circuit: sequential layered circuits. Both benchmarks have only a basic description of the circuits (number of inputs, outputs, logic elements and states of the finite automaton), which requires additional calculations when they are used to verify various methods for assessing the reliability of circuits, but the variety of circuits allows testing various methods in a more comprehensive and detailed manner.

Thus, existing combinational circuit benchmarks, consisting of combinational circuits in Verilog format and their basic parameters, can be used to check the correctness of the work of various algorithms but are poorly suited for use in machine learning, since they contain a limited number of circuits. Therefore, there is a need for a labeled dataset of large-sized combinational circuits (hundreds of thousands of pieces with a wide range of inputs and outputs) and a description of their extended parameters, which can be used in machine learning tasks.

III. Generation algorithms

To generate a variety of combinational circuits, it is necessary to use both various methods for generating combinational circuits and the ability to parameterize the generated combinational circuits with varying degrees of randomness. Therefore, various methods were developed and applied, as outlined in the following sections.

A. Combinational Circuit Generation using Random Truth Tables (CCGRTT)

The simplest method of generation of a combinational circuit is the synthesis of a truth table of the circuit and its subsequent processing. Several steps need to be taken to perform the schema generation [14]:

1. random truth table generation;
2. building PDNF and/or PCNF;
3. representation of the resulting logical expression in the given basis;
4. combinational circuit synthesis in Verilog.

B. Combinational Circuits Generation by the Random Connection of Gates (CCGRCG)

This method is a variation of the random generation of a directed graph that allows parameterizing the generation process and the final scheme.

For the generator to function correctly, it is required to submit the input data that matches the formalized rules. The generator receives four required parameters. They are: the number of inputs and outputs; the number of levels of logical elements; the number of logical elements at each level. The result of the algorithm is a directed graph, which is a combinational circuit that can subsequently be converted into a structural Verilog format.

The work of the algorithm can be divided into several main parts.

In the first phase, the directed graph object is created and then all inputs of the combinational circuit are added to it. The method of adding combinational circuit elements is such that the number of vertices, located at each level, is always known. Adding vertices (logical elements) to the graph occurs sequentially from level to level. The choice of the type of a logical element occurs depending on the logical basis chosen by the user, which can be specified in the required way with a restriction on its functional completeness. As a result, it is possible to control the number of logical elements at the considered level, to which elements will be added, and at the previous level, the elements of which are randomly selected to connect with the elements of the current level. As a result, when adding vertices responsible for inputs, the range of the current level is shifted, thereby preparing the algorithm for the second stage of circuit synthesis.

In the second phase, the connections of logical elements are added. First, a logical element is randomly selected from the available list. After that, elements from previous levels are selected, the number of which depends on the parameters of the logical expression. Thus, the top of the graph is created and connected with the selected elements of the previous levels, forming a combinational circuit; with this approach, cyclic dependencies are possible, which allow increasing the diversity of the final set of combinational circuits. The connection process continues until all the elements of the current level are connected with the required number of logical elements. As a result of processing each level of the logical circuit by the method presented, a combinational circuit in the form of a directed graph is generated.

The third phase is to connect the outputs to the circuit generated. To perform this task, one has to select the elements belonging to the last level of the combinational circuit and connect them to the outputs. In situations where the number of combinational circuit outputs exceeds the number of elements at the last level, the outputs can be reconnected to these elements.

As a result of the algorithm, a directed graph (which is a combinational circuit) is generated; it is saved in the Verilog format.

C. Combinational Circuits Generation by Random Vertex Connection (CCGRVC)

Like the previous algorithm implemented, this algorithm is a variation of random graph generation and has a higher degree of parameterization compared to the previous one.

The generation of a combinational circuit by the method proposed occurs in several stages.

The first stage of the algorithm is preparatory. The generator takes three required parameters and one optional parameter as input. Mandatory are: the number of inputs and outputs, a dictionary that contains the basis and the number of each logical operation. An additional parameter is a flag that determines what happens when the number of outputs exceeds the number of existing vertices. In one case, the output vertices remain unconnected to the others, and thus, the resulting signal at them remains unknown. In another case, the outputs, left without a connection to the vertices, will be connected to the input vertices with repetitions. Also, within this stage, a directed graph object with input and output vertices and additional variables are created to perform the subsequent stages of the algorithm.

The second stage begins with counting the total number of all operations, after which the main cycle of adding new vertices and edges of the graph is performed. Each iteration of the loop begins by checking the dictionary with the number of each of the operations and removing from it those whose number became zero during their use. The next step is to randomly select an operation from the dictionary passed to the input and reduce the number of available elements of this type. Next, parent vertices are selected, and their number depends on the selected logical operation; they themselves must be non-repeating. In addition, an auxiliary dictionary is updated with all the vertices and their levels, respectively. These actions are repeated until the dictionary with the available logical operations becomes empty. This means that all available logical operations were added, and they are interconnected. The use of any logical element as a parent expands the variety of possible combinational circuits; for example, due to the occurrence of cyclic dependencies.

The final stage is the connection of the vertices with the output vertices. The most important are the vertices with the highest level, so it is among them that random selection will take place first – to connect with the exits. This process is performed until all the output vertices are connected to one of the graph vertices.

As a result of the algorithm, the graph generated is saved in the Verilog format.

D. Combinational Circuits Generation based on Genetic Algorithm (CCGGA)

One of the promising methods for generating combinational circuits is the genetic algorithm [15]. It is used in various fields ranging from the optimization of various circuits [16], [17] o the automation of the design of analog circuits [18]. The genetic algorithm can be used to search for new configurations of graphs [19], and for a combinational scheme, a graph is quite a suitable form of representation. Also, the genetic algorithm is an optimization method [20], [21], which generates intermediate results as it moves towards the goal. This allows setting the goal of generating combinational circuits in the form of an objective function based on the given parameters and thus, obtain not only the final result in the form of a circuit but also intermediate combinational circuits, which, with each iteration, will approach the target. It is an adaptive search method that is used to solve optimization problems. Genetic algorithms use an analogue of the mechanism of genetic inheritance, as well as the natural selection.

The first step in the work of the genetic algorithm is the generation of an initial population from a given number of chromosomes. After that, the fitness of each of the chromosomes is calculated based on a given algorithm. If the stopping criterion is not reached (the number of iterations has reached its maximum or the calculated fitness of one of the chromosomes meets the specified condition), a new population is generated as follows:

1. calculation of parental chromosomes using one of the selection methods;
2. crossbreeding the selected parents with the given parameters;
3. mutation of offsprings;
4. selection of a new population;
5. calculating the suitability of each of the offspring chromosomes.

Thus, due to the fact that at every step of the genetic algorithm it is possible to use one of the given methods, a variety of different combinational schemes is achieved with different levels of randomness introduced into the schemes.

The features of combinational circulation generator based on genetic algorithm are:

1) *Parameters of the CCGGA*

The main parameters for generating combinational circuits using the developed genetic algorithm are as follows:

- population size;
- number of genetic algorithm cycles;
- population suitability threshold for algorithm completion;
- parameters for parent selection:
    - type of parent selection method;
    - selection parameters;
- crossbreeding parameters:
    - type of crossbreeding method;
    - crossbreeding parameters;
- mutation parameters:
    - type of mutation method;
    - mutation parameters;
- parameters of new population selection:
    - type of new population selection method;
    - new population selection parameters.

The specified set of parameters allows full customizing the operation of the genetic algorithm and generating combinational circuits.

2) *Selection of parents from the population*

The initial stage of the work of the CCGGA is the selection of parents from a given population. For this stage, five main types of parent selection were implemented [15]: panmixia, inbreeding, outbreeding, tournament method, roulette method.

Each of the implemented selection methods can be fully parameterized using the generation parameters class described above. Thus, parents will be selected from the population according to the given type of parent selection and will be used for crossbreeding and creating a new population.

3) *Recombination*

After parent selection, crossbreeding is necessary. Recombination is understood as the production of a descendant chromosome on the basis of previously selected parents. The most appropriate type of recombination for the problem of generating combinational circuits is crossbreeding.

The following methods were chosen as crossbreeding methods: crossbreeding at several reference points, uniform, triadic, permutation crossbreeding, as well as crossbreeding with a decrease in replacement.

4) Mutation

After crossbreeding, it is necessary to mutate the resulting descendants. Mutation allows one to introduce random changes in the chromosome and, in standard applications, get out of a local extremum. For problems of generation of combinational schemes, mutations introduce a random scatter from an initially generated to crossbreed population. To carry out the mutation, the following methods were implemented: binary mutation, density mutation, mutation-attachment and insertion with previous deletions, exchange of places, mutation-deletion.

5) Selection to a new population

After crossbreeding, it is necessary to select for a new population. As a selection method, a basic selection was implemented, which was the selection of individuals based on the adaptation index. This method is the main one and allows selecting the most suitable schemes for a given condition.

E. Combinational Circuits Generation based on the User-Defined Schemes (CCGUDS)

As a separate method of creation of new combinational circuits, one can single out their manual creation. This method assumes that the user develops a combinational circuit in the Verilog format, and the program analyzes the provided combinational circuit and generates its parameters.

This method allows one not only to independently create combinational circuits but also add third-party combinational circuits; for example, from benchmarks, which will be processed, and their parameters will be generated.

IV. Generation of combination schemes

The developed methods for generating combinational circuits were implemented in the form of software in the high-level language C# [22]. To transfer the generated circuits to the Nangate open library format, third-party software Nadezhda [23] was used, which allows one to transfer the circuit from the usual Verilog to the Nangate library format, as well as to evaluate the reliability of the resulting combinational circuit. After converting the scheme to a given format, it is analyzed, and its parameters are calculated. The main parameters were the following:

- number of inputs;
- number of inputs;
- length of the critical path;
- number of logic elements in the circuit;
- number of paths in the circuit;
- number of elements of each type;
- number of edges of each type;
- reliability of the combinational circuit expressed as a percentage.

As a result, for each generated scheme, its optimized analogue is synthesized in the Nangate library format, as well as the scheme parameters in a file in JSON format [24]. Thus, based on the set of generated schemes, a dataset is generated in csv format, in which all parameters of each of the schemes are recorded [22].

V. Approbation of the methods developed

To determine the suitability of the developed methods and combinational circuits generated as a result of their work, for use in machine learning, two algorithms were developed [22], which allow predicting the reliability of combinational circuits based on training on the dataset generated. The XGBoost algorithm [25], [26]. was chosen as the first idea for implementing a method for assessing the reliability of combinational circuits using machine learning. This algorithm is a machine learning algorithm based on a decision tree and using the gradient boosting framework. It has gained a lot of popularity as the algorithm chosen by many winning teams in many machine learning competitions. As the second method for calculating reliability, a regression neural network was chosen, which analyzes the parameters of the circuit and calculates its reliability.

To train the selected algorithms, a dataset of combinational circuits was generated using the developed software [23]. The dataset with the generated combinational circuits has a size of 100 thousand records with the following parameters:

- the number of inputs and outputs does not exceed 20;
- number of logical elements in each circuit does not exceed 1 thousand;
- number of circuits generated by the CCGRTT method: 20 thousand;
- number of circuits generated by the CCGRCG method: 20 thousand;
- number of circuits generated by the CCGRVC method: 20 thousand;
- number of circuits generated by the CCGGA method: 40 thousand.

The dataset was divided into 80% and 20% for network training and testing, respectively. The ISCAS85 benchmark [9] was chosen as a set of combinational circuits for the final validation of the algorithms, each circuit of which was brought to the required format, and all its parameters were calculated. Fig. 1 shows the main results reflecting the quality of the algorithms. The average deviation between the predicted values and the reference values was: 3.41% and 59.92% on the generated dataset and on the ISCAS85 benchmark, respectively, when using XGBoost; and 3.4% and 5.2%, respectively, when using a regression neural network. More accurate predictions on the dataset are explained by the fact that the dataset contains fully generated data, while the dataset for validation contains real schemas used in various fields of activity. The lower accuracy of prediction using XGBoost, compared to a neural network on real circuits, is explained by the ability of a neural network to better adapt when training on disparate data.

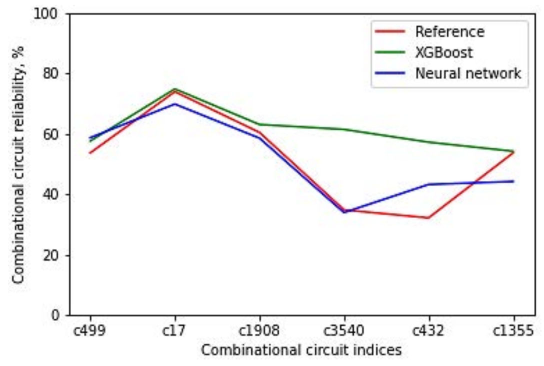

Fig. 1. Using a dataset to predict the circuit reliability.

VI. Conclusion

This work presented a description of algorithms for generating combinational circuits with different levels of parameterization by the user. Applying the methods developed and implemented in the form of software, a dataset of combinational circuits was generated, which was tested using the two most popular machine learning methods. As a result, the average deviation between the predicted values and the reference values was 3.4% and 5.2% on the generated dataset and on the ISCAS85 benchmark, respectively. Thus, the obtained results of predicting the reliability of combinational circuits using machine learning methods allow us to state that the refinement of methods for generating combinational circuits, as well as the increase and optimization of the existing dataset of combinational circuits, will increase the accuracy of the predictions, as well as increase the coverage of a larger set of the circuits.

References

[1] S. Roy and C. Tilak, “On synthesis of combinational logic circuits,” International Journal of Computer Applications, vol. 127, no. 1, pp. 21–26, 2015. DOI:10.5120/IJCA2015906311.

[2] A. Youssef, B. Majeed and C. Ryan, “Optimizing combinational logic circuits using Grammatical Evolution,” NILES 2021 - 3rd Novel Intelligent and Leading Emerging Sciences Conference, Proceedings, pp. 87–92, 2021. DOI:10.1109/NILES53778.2021.9600092.

[3] S. L. Harris and D. M. Harris, Digital Design and Computer Architecture: RISC-V Edition. Morgan Kaufmann, 2021.

[4] A. L. Stempkovskiy, D. V. Telpukhov, R. A. Soloviev, and N. V. Telpukhova, “Probabilistic methods for reliability evaluation of combinational circuits,” Problems of Advanced Micro- and Nanoelectronic Systems Development (MES), no. 4, pp. 121–126, 2016.

[5] H. Asadi, M. B. Tahoori, M. Fazeli and S. G. Miremadi, “Efficient algorithms to accurately compute derating factors of digital circuits,” Microelectronics Reliability, vol. 52, no. 6, pp. 1215–1226, Jun. 2012, DOI:10.1016/J.MICROREL.2011.12.031.

[6] J. Han, H. Chen, E. Boykin and J. Fortes, “Reliability evaluation of logic circuits using probabilistic gate models,” Microelectronics Reliability, vol. 51, no. 2, pp. 468–476, Feb. 2011, DOI:10.1016/j.microrel.2010.07.154.

[7] M. R. Choudhury and K. Mohanram, “Reliability analysis of logic circuits,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 3, pp. 392–405, Mar. 2009, DOI:10.1109/TCAD.2009.2012530.

[8] A. Youssef, B. Majeed and C. Ryan, “Optimizing combinational logic circuits using Grammatical Evolution,” 3rd Novel Intelligent and Leading Emerging Sciences Conference, Proceedings, pp. 87–92, 2021. DOI:10.1109/NILES53778.2021.9600092.

[9] H. Fujiwara, “A neutral netlist of 10 combinational circuits and a target translator in fortran,” Proc. of the International Symposium on Circuits and Systems, 1985.

[10] F. Brglez, D. Bryan, and K. Kozminski, “Combinational profiles of sequential benchmark circuits,” Proceedings - IEEE International Symposium on Circuits and Systems, vol. 3, pp. 1929–1934, 1989. DOI:10.1109/ISCAS.1989.100747.

[11] S. Yang, “Logic synthesis and optimization benchmarks,” in 1989 MCNC International Workshop on Logic Synthesis, 1988, 45 p.

[12] S. Yang, “Logic synthesis and optimization benchmarks user guide: Version 3.0,” in Microelectronics Center of North Carolina (MCNC), 1991, 45 p.

[13] C. Albrecht, “IWLS 2005 Benchmarks,” 2005.

[14] G. D. Hachtel and Fabio. Somenzi, Logic synthesis and verification algorithms. Kluwer Academic Publishers, 1996.

[15] W. Banzhaf, P. Nordin, R. Keller and F. Francone, Genetic programming : an introduction on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers, 1998.

[16] S. Papadopoulos, R. J. Mack and R. E. Massara, “A hybrid genetic algorithm method for optimizing analog circuits,” Midwest Symposium on Circuits and Systems, vol. 1, pp. 140–143, 2000. DOI:10.1109/MWSCAS.2000.951605.

[17] D. V. Telpukhov, “Development of methods for genetic synthesis of fault-tolerant logic circuits,” Problems of Advanced Micro- and Nanoelectronic Systems Development (MES), no. 1, pp. 45–49, 2018. DOI:10.31114/2078-7707-2018-1-45-49.

[18] I. Canturk and N. Kahraman, “Comparative analog circuit design automation based on multi-objective evolutionary algorithms: An application on CMOS opamp,” 38th International Conference on Telecommunications and Signal Processing (TSP), Jul. 2015, pp. 1–4, DOI:10.1109/TSP.2015.7296478.

[19] O. Romanov and O. Lysenko, “The evolutionary computation method for the synthesis of networks-on-chip quasi-optimal topologies,” IEEE 34th International Scientific Conference on Electronics and Nanotechnology (ELNANO), 2014, pp. 403–407, DOI:10.1109/ELNANO.2014.6873434.

[20] X. He et al., “Sparse-TPU: adapting systolic arrays for sparse matrices,” in Proceedings of the 34th ACM International Conference on Supercomputing, pp. 1–12, Jun. 2020. DOI:10.1145/3392717.3392751.

[21] Y. G. Gao and D. Y. Song, “A new improved genetic algorithms and its property analysis,” 3rd International Conference on Genetic and Evolutionary Computing, WGEC 2009, pp. 73–76, 2009. DOI:10.1109/WGEC.2009.150.

[22] Combinational Circuits Dataset. Zenodo, 2022. DOI:10.5281/zenodo.6783568.

[23] Nadezhda - Reliability Enhancement Logic tool for Integrated Circuits design. [Online]. Available: https://alphachip-tools.ru/nadezhda.php

[24] Json.NET - Newtonsoft. [Online]. Available: https://www.newtonsoft.com/json

[25] T. Chen and C. Guestrin, “XGBoost,” in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, Aug. 2016. DOI:10.1145/2939672.2939785.

[26] B. Pavlyshenko, “Machine learning, linear and Bayesian models for logistic regression in failure detection problems,” Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016, pp. 2046–2050, 2016. DOI:10.1109/BIGDATA.2016.7840828.