ISSN 0002-306X. Proc. of the RA NAS and NPUA Ser. of tech. sc. 2024. V. LXXVII, N1

UDC 621.3.049.77 **MICROELECTRONICS**

DOI: 10.53297/0002306X-2024.v77.1-58

V.Sh. MELIKYAN, A.A. GALSTYAN, S.A. GHUKASYAN, A.A. GHAZARYAN, E.E. KARAPETYAN

DEVELOPMENT OF PARAMETERIZED MODEL OF LOGIC ELEMENTS AT CLOCK TREE SYNTHESIS

Clock synthesis, routing optimization, placement and logic optimization are the three primary phases of physical design implementation. Since clock network synthesis uses at least 30% of the entire power budget, it is one of the crucial steps. Power consumption for high-performance blocks can reach 50% of the entire power. Not only would a high-quality clock tree will fix timing violations, but it will also minimize power usage and routing resource use. A new neural network based parameterized model is proposed in this paper, which can be used to obtain not only the list of logic elements, but it also can predict the circuits timing behaviour. Different ICs using SAED 14 and 32 nm technologies are designed using the proposed method.

Keywords: clock tree synthesis, automation, physical design, timing critical, neural network.

Introduction. There are three primary branches of clock distribution: the conventional clock network, the multisource clock network, and the mesh network (Fig.1). The mesh network is deeper in the global clock trunk than the multisource clock network, while the multisource clock network is shallower. This is the difference between the two types of clock networks.

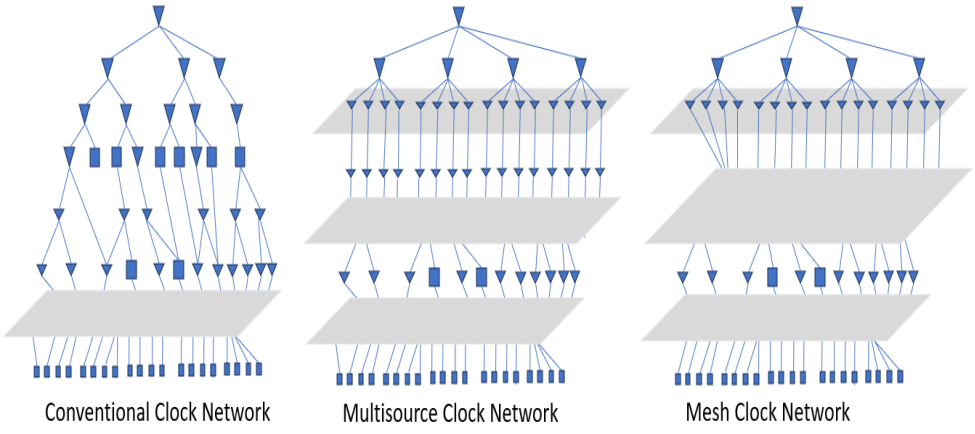

Fig. 1. Three main branches of clock distribution

The clock structure will include a global driver and a local sink (Fig.2)[1]. Mesh routing distinguishes between conventional clock networks and mesh networks, often known as multi source clock networks. Higher local sink depth is a benefit of multisource clock dispersion. In comparison to a full mesh network, it trades off routing resources, and consequently power as well. The reduced on-chip variance is a benefit of the mesh network. The drawback of the mesh or multisource clock distribution is that proper clock network delay propagation requires the use of spice simulation to extract the cell, net delay, and output transition time at the mesh routing. This is due to the inaccuracy with which static timing analysis (STA) can determine the mesh driver's cell output delay, mesh net latency, and mesh net output transition.

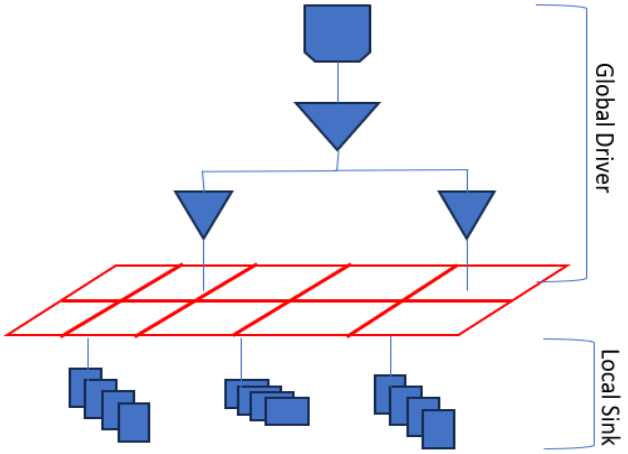

Fig. 2. Global and Local sinks

The distinction between the mesh clock distribution and the multisource method is that the latter may skip step 7 since step 1 contains additional tap drivers. Traditionally, a custom method has been used to complete the global driver stages 1 through 5 (Fig.3), because the EDA tool is not yet ready to handle the implementation [2-6]. The drawback of a tailored solution may be the requirement for numerous iterations to reach the desired time convergence and latency.

This is because of the nature of the estimated versus expected implementation error gap. For global clock drivers, some designs may have used a bespoke super-clock cell [7,8] in order to obtain a greater driving range and reduced power consumption.

In different physical implementation tools, basically two CTS flows are supported, Classic CTS and Concurrent Clock & Data (CCD). During the first flow, first runs CTS then data path optimization. The clock tree is built while ignoring

the data paths, the goal is to minimize skew. In the second CCD flow, CTS and data path optimization perform concurrently and it is recommended for timing-critical designs. The clock tree is built with full knowledge of data path timing the goal is to meet setup/hold timing.

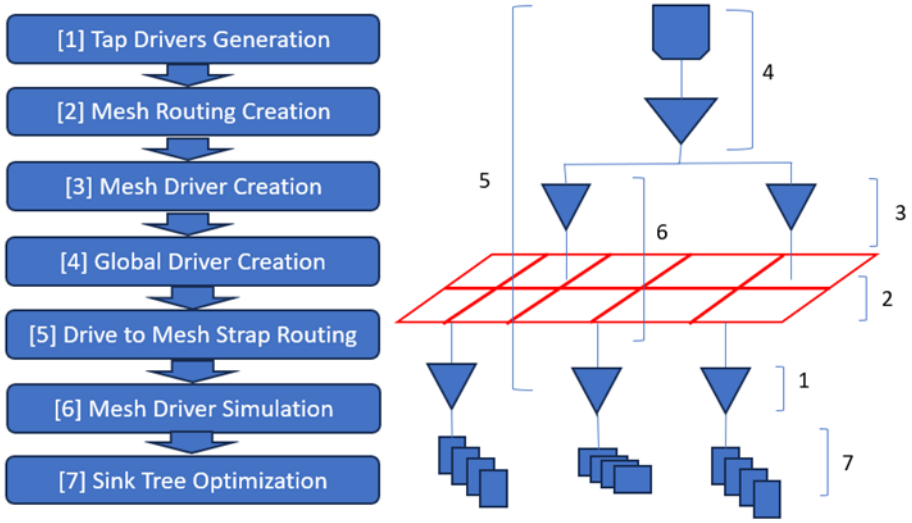

Fig. 3. Clock tree optimization steps

Various approaches have been used to optimize power, delay and skew to make clock distribution networks suitable for low-power and high-performance designs. An extensive study of different clock distribution networks, their timing performance is carried out. Different architectures for clock distribution networks are employed to meet various design requirements [9-14].

Thus, having a parameterized model to choose the right cell known as super clock cell from Logic Libraries will shrink selecting time for Physical Implementation EDA tools the clock tree will be balanced and time convergence, latency, power dissipation will be in the desired range. Having a parameterized model also shrinks time-to-time, as iterations in already mentioned clock tree optimization steps will reduce. Choosing super clock cells from Logic Libraries is a difficult task, neural network principles are used during this work. The proposed model can be used for all clock tree strategies during optimization.

Neural networks (Fig.4) are computational models inspired by the human brain, designed to recognize patterns, and make decisions.

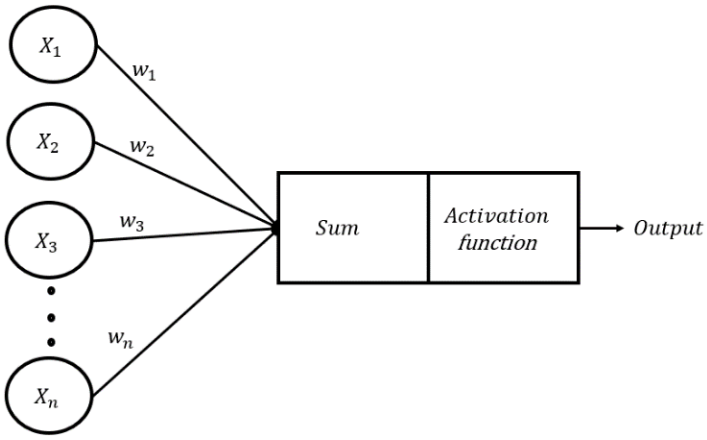

Fig. 4. Graphical abstraction of a neural network

Keras, a high-level neural network API [15], simplifies the implementation process by providing an easy interface for building and training models. Algorithms within neural networks, like backpropagation, optimize weights to minimize errors during training, enhancing the model's ability to generalize patterns in data.

Keras API incorporates diverse algorithms for training neural networks. One fundamental algorithm is backpropagation, wherein the model (Fig.5) refines weights based on the gradient of the loss function concerning those weights. This optimization process aims to minimize the disparity between the predicted and actual outputs during the training phase.

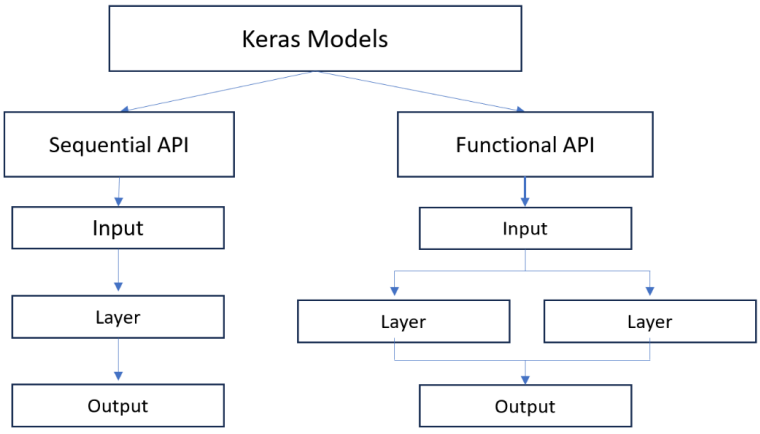

Fig. 5. Two models of Keras

Moreover, Keras supports various optimization algorithms, such as Adam and Stochastic Gradient Descent (SGD), influencing the weight updating. These algorithms contribute to the improvement of convergence speed and efficiency during

the training process. Additionally, Keras provides activation functions like ReLU, Sigmoid, and Tanh, introduction of non-linearity to the model. This characteristic enables the model to grasp intricate patterns and relationships within the data.

Having an automated flow to generate a cell list for the clock tree synthesis provides several benefits:

- Automation: A flow allows for automation of the process, reducing the change of human error and saving time.
- Scalability: As the design complexity increases, manually maintaining a cell list becomes impractical.

And also, using neural networks for predicting the clock tree synthesis results in the early stages of design with cells offers several benefits:

- Speed and efficiency: Neural network can quickly process large amounts of data and make predictions faster than traditional methods.
- Complex pattern recognition: Neural networks excel at recognizing complex patterns within datasets. In the context of CTS, where the interactions between different cells and components are intricate, neural networks can capture these relationships and provide more accurate predictions.
- Adaptability: Neural networks can adapt to different design scenarios and change in requirements.
- Early design insights: Neural networks can provide early insights into potential CTS challenges and optimizations.

The proposed approach. A new methodology is described aimed to fasten the process and help the tool to fix the selected problem during the CTS stage.

The CTS creating and analyzer program which is implemented using the proposed algorithm is described below.

The CTS creating and analyzer work can be separated into three main parts:

1. Library analyzer – which will select the most optimal cells from std cells library for CTS.
2. Clock tree analyzer –AI - based system, which will predict the results after CTS, which used the selected cells.
3. CTS – The CTS process in physical implementation tool, where we use cells required in the library analyzer stage.

Graphical implementation for the above mentioned (Fig. 6).

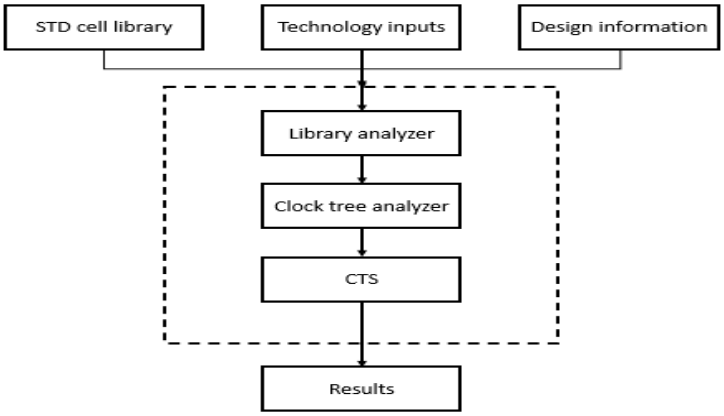

Fig. 6. Graphical approach to the proposed algorithm

As mentioned, the first stage is the library analyzer. The Library Analyzer, implemented as a Perl script, examines the .lib file, identifying and systematically reporting on the presence of all symmetric cells. As a result, Library analyzer yields a comprehensive list of symmetric cells as a consequence of its analysis. Notably, the defined threshold for inclusion is set at 30%, signifying those cells surpassing this symmetry criterion for utilization in our Clock Tree Synthesis (CTS) processes. Also, library analyzer filters the cells from one VT because, it is essential to achieve synchronous and reliable operation in integrated circuits. This uniform VT is instrumental in mitigating skew, which refers to timing discrepancies among different segments of the clock tree. By ensuring a consistent VT, clock signals, propagate at comparable speeds throughout the circuit reducing timing variations and enhancing the overall performance and stability of the system. The main steps for library analyzer are: (Fig. 7).

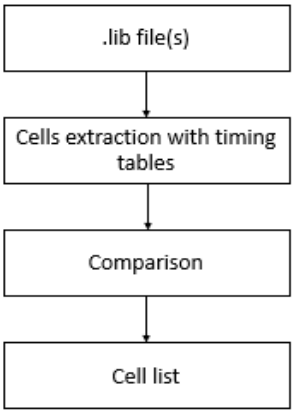

Fig. 7. The library analyzer's workflow

In the second stage, we analyze our design for the CTS side by using the analyzer. The program is python-based where the user should do the steps mentioned below (Fig.8):

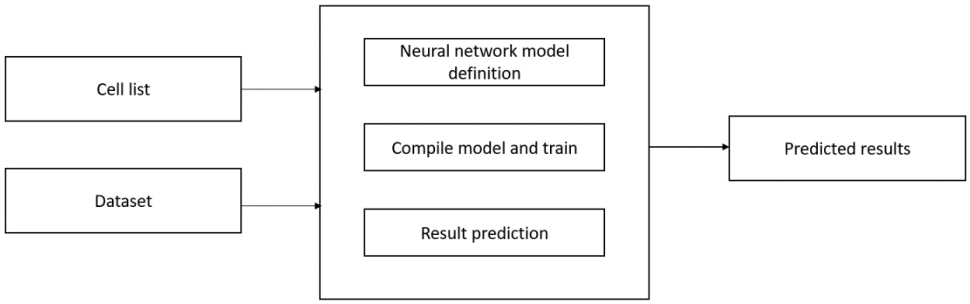

Fig. 8. Part of the used database

1. Load and split dataset which has been initially created. The part of the initially created database shown in Table. 1.
2. Normalize the input features.
3. Define the neural network model: The script uses the "Sequential" model from the "keras.models" module. The model consists of three Dense layers with different activation functions. The first layer has 64 units and uses the "relu" activation function. The second layer also has 64 units with "relu" activation. The third and final layer has a single unit with a linear activation function.
4. Compile the model: The model is compiled using the "mean_squared_error" loss function and the "adam" optimizer. This prepares the model for training.
5. Train the model: The model is trained using the "fit()" function, which takes the training sets ("X_train" and "y_train"), the number of training epochs, the batch size, and the "verbose" parameter to control the progress output.
6. Evaluate the model: After training, the model is evaluated on the testing set using the "evaluate()" function. The mean squared error (MSE) between the predicted values and the true values is calculated and printed.
7. Make predictions on new data: Finally, the model is used to make predictions on new input data. A new data point is created (in this example, with arbitrary values), normalized using the same scaler used for training and passed to the model to obtain the predicted slack value. The predicted slack is then printed.
8. In the final stage, the real CTS in physical implementation tool is implemented, with the use of predicted cells.

Table 1

Dataset of different technologies

Tech nology	Metal stack	Gate count	Macro count	STD cells utilization	Utili zation	Cells	Slack (*ps*)	Setup (*ps*)	Hold (*ps*)
12	10	152903	5	56.07	0.91	LVT_INV_S_3; LVT_INV_8; LVT_INV_10	-0.38	-35	-1
14	11	152424	5	55.35	0.91	LVT_INV_S_4; LVT_INV_6	-0.12	-10	-2
12	9	140880	5	75.77	0.84	LVT_INV_8; LVT_INV_10	-0.4	-26	-1
16	11	168452	3	68.23	0.75	LVT_INV_3; LVT_INV_4; LVT_INV_6; LVT_INV_8 MUX2_MM_2	-0.1	0	0

Results. The library analyzer provides an automated solution for cells selecting process. The solution is technology independent and fully automated which means that it gives an opportunity to do the selection automatically instead of doing it manually. For the design parameter prediction, a CTS analyzer has been developed which is a neural network-based workflow. The CTS analyzer gives an opportunity to check the design slack, setup, hold, and dynamic power consumption results in early stages. The CTS analyzer's operational (predicted results raw) and results from design (real results row) outcomes are comprehensively presented in a tabular format (Table 2).

Table 2

The predicted versus real results

Design	Predicted results				Real results			
	Slack (*ps*)	Setup (*ps*)	Hold (*ps*)	Dynamic Power(*mWt*)	Slack (*ps*)	Setup (*ps*)	Hold (*ps*)	Dynamic Power (*mWt*)
Design1	-15	-7	-2	0.967	-12	-8	0	0.923
Design2	-13	-9	-25	1.125	-11	-11	-26	1.101
Design3	-8	0	-6	0.865	-9	-2	-7	0.894

Conclusion. In this paper, a digital standard cell library parameterized neural network model was developed, from which the most optimal cells mostly fast and symmetrical, are selected for the construction of the clock tree. The proposed model is able to achieve timing and power improvement compared to the default clock tree synthesis. It is suitable for full chip clock planning, as reducing iterations for spice simulation tool to backannotate the driver cell delay. Next, parameters are

pre-determined, the maximum deviation of which is 20% compared to the real one. The application of the model makes it possible to reduce the design time, as it gives a chance to get an idea about the parameters of the circuit in the initial stages of design. Since customization is allowed, it can be part of the non-default permuton exploration in design space optimization (DSO.AI) [16] tool, for future studies.

REFERENCES

1. **Chong A.B.** Hybrid Multisource Clock Tree Synthesis // 2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS). - Dubai, United Arab Emirates, 2021. – P. 1-6, doi: 10.1109/ICECS53924.2021.9665516.
2. ObstacleAvoiding and Slew-Constrained Clock Tree Synthesis with Efficient Buffer Insertion **/ Y. Cai, C. Deng, C. N. Sze, et al //** IEEE Transactions on Very Large-Scale Integration (VLSI) Systems. - Jan. 2015. - Vol. 23, no. 1. - P. 142-155.
3. Practical Full Chip Clock Distribution Design with a Flexible Topology and Hybrid Metaheuristic Technique **/ E.K. Teh, M.A. M. Zawawi, M.F. P. Mohamed, et al** // IEEE Access. – 2021. - Vol. 9. - P. 14816-14835, doi: 10.1109/ACCESS.2021.3053052.
4. **Rajaram A., and Pan D.Z.** Robust chip-level clock tree synthesis for SOC designs // 2008 45th ACM/IEEE Design Automation Conference. - Anaheim, CA, USA, 2008. - P. 720-723, doi: 10.1145/1391469.1391654.
5. **Fumihiro M., and Midori T.** Clock Tree Synthesis Based on RC Delay Balancing // Proceedings of The IEEE Custom Integrated Circuits Conference. - May 1992. - P.28.3.1-28.3.4, doi: 10.1109/CICC.1992.591862
6. **Dong-Jin L., and Igor L.M.** Obstacle-aware Clock-Tree Shaping During Placement // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.- Feb. 2012. - Vol. 31. - P. 205-216, doi: 10.1109/TCAD.2011.2173490
7. **Zhu H. and Kursun V.** 2-Phase high-frequency clock distribution with SPLIT-IO dual-Vt repeaters for suppressed leakage currents // 2015 IEEE International Symposium on Circuits and Systems (ISCAS). - May 2015. - P. 2932-2935, doi: 10.1109/ISCAS.2015.7169301.
8. **Kumar Gundu A. and Kursun V.** Low Leakage Clock Tree with Dual-Threshold-Voltage Split Input–Output Repeaters // IEEE Transactions on Very Large-Scale Integration (VLSI) Systems.- July 2019. - V. 27, no. 7. - P. 1537-1547.
9. **Vishnu P.V., Priyarenjini A.R., and Kotha N.** Clock Tree Synthesis Techniques for Optimal Power and Timing Convergence in SoC Partitions // 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT). - Bangalore, India, 2019. - P. 276-280, doi: 10.1109/RTEICT46194.2019.9016727.
10. A Clock Tree Synthesis Scheme Based on Flexible H-tree / **H. Wang, Q. Lei, Y. Yang, et al //** 2022 2nd International Conference on Electrical Engineering and Mechatronics Technology (ICEEMT). - Hangzhou, China, 2022. - P. 249-252, doi: 10.1109/ICEEMT56362.2022.9862608.

11. Clock Tree Resynthesis for Multi-Corner Multi-Mode Timing Closure / **S. Roy, P.M. Mattheakis, L. Masse-Navette, et al //** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. - April 2015. - V. 34, no. 4. - P. 589-602, doi: 10.1109/TCAD.2015.2394310.
12. Top-level activity-driven clock tree synthesis with clock skew variation considered / **T.J. Wang, S.H. Huang, W.K. Cheng, et al** // 2016 IEEE International Symposium on Circuits and Systems (ISCAS). - Montreal, QC, Canada, 2016. – P. 2591-2594, doi: 10.1109/ISCAS.2016.7539123.
13. **Melikyan V., Martirosyan M., Melikyan A., Piliposyan G.** 14nm Educational Design Kit: Capabilities, Deployment and Future // Small Systems Simulation Symposium. - Niš, Serbia, 12th-14th February 2018. - P. 37-41.
14. 32/28nm Educational Design Kit: Capabilities, deployment and future / **Goldman, R. Bartleson, K. Wood, T. Kranen, et al** // Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia), IEEE. - 2013. - P. 284-288, doi: 10.1109/PrimeAsia.2013.6731222.
15. **Gulli A and Pal S.,** Deep learning with Keras.- Packt Publishing Ltd; 2017.
16. Design Space Optimization AI, https://www.synopsys.com/implementation-and-signoff/ml-aidesign/dso-ai.html

National Polytechnical University of Armenia.The material is received on 01.04.2024.

Վ.Շ. ՄԵԼԻՔՅԱՆ, Ա.Ա. ԳԱԼՍՏՅԱՆ, Ս.Ա. ՂՈՒԿԱՍՅԱՆ, Ա.Ա. ՂԱԶԱՐՅԱՆ, Է.Ե. ԿԱՐԱՊԵՏՅԱՆ

ՍԻՆՔՐՈԱԶԴԱՆՇԱՆԱՅԻՆ ԾԱՌԻ ՍԻՆԹԵԶԻ ԴԵՊՔՈՒՄ ՏՐԱՄԱԲԱՆԱԿԱՆ ՏԱՐՐԵՐԻ ՊԱՐԱՄԵՏՐԱՑՎԱԾ ՄՈԴԵԼԻ ՄՇԱԿՈՒՄԸ

Սինքրոազդանշանային ծառի, միջմիացումների և տեղաբաշխման լավարկումը ֆիզիկական նախագծման երեք հիմնական փուլերն են: Քանի որ սինքրոազդանշանային ծառն օգտագործում է ամբողջ էներգիայի առնվազն 30%-ը, հետևաբար՝ այն ֆիզիկական կարևոր փուլերից մեկն է: Բարձր արդյունավետությամբ ինտեգրալ սխեմաների համար էներգիայի սպառումը կարող է հասնել ամբողջ էներգիայի 50%-ին: Լավարկված սինքրոազդանշանային ծառը ոչ միայն կուղղի ժամանակի խախտումները, այլև կնվազեցնի և՛ էներգիայի, և՛ միջմիացումների ռեսուրսի օգտագործումը: Առաջարկվում է նեյրոնային ցանցի վրա հիմնված նոր պարամետրացված մոդել, որը կարող է օգտագործվել ոչ միայն տրամաբանական տարրերի ցանկը ստանալու համար, այլև կանխատեսում է սխեմաների ժամանակային պարամետրերը սինքրոազդանշանային ծառի նախագծման ընթացքում: Պարամետրերը գնահատելու համար առաջարկված մեթոդով նախագծվել են տարատեսակ ինտեգրալ սխեմաներ՝ ՍԱՌԻԴ 14 և 32 նմ տեխնոլոգիաներով:

Առանցքային բառեր. սինքրոազդանշանային ծառի սինթեզ, ավտոմատացում, ֆիզիկական նախագծում, կրիտիկական ժամանակ, նեյրոնային ցանց:

В.Ш. МЕЛИКЯН, А.А. ГАЛСТЯН, С.А. ГУКАСЯН, А.А. КАЗАРЯН, Э.Е. КАРАПЕТЯН

РАЗРАБОТКА ПАРАМЕТРИЗОВАННОЙ МОДЕЛИ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ ПРИ СИНТЕЗЕ ДЕРЕВА СИНХРОСИГНАЛА

Оптимизация синхросигналов, межсоединений и размещения — три основных этапа физического проектирования. Поскольку дерево синхронизации потребляет не менее 30% общей энергии, следовательно, это один из важных этапов. Для высокоэффективных блоков потребление энергии может достигать 50% от общего объема энергии. Оптимизированное дерево синхронизации не только исправит нарушения синхронизации, но также снизит энергопотребление и использование ресурсов межсетевого соединения. В данной статье предлагается новая параметризованная модель на основе нейронной сети, которую можно использовать не только для получения списка логических элементов, но и для прогнозирования временных параметров схем при построении дерева синхросигналов. С использованием предложенного метода оценки параметров были спроектированы различные интегральные схемы с технологиями САУД 14 и 32 *нм*.

Ключевые слова: синтез дерева синхросигнала, автоматизация, физическое проектирование, критическое время, нейронная сеть.