Abstract—Since quantum computing technologies are still in their early development phase, quantum circuits are created mainly by manual placement of logic elements. This development approach has the drawback of becoming inefficient due to lack of human comprehension when analyzing large circuits that correspond to complex algorithms. Because, even a slight increase in the number of operations in a quantum algorithm, could lead to the significant increase in size of its corresponding quantum circuit. Therefore, the purpose of creating Quantum Circuit Synthesizer "Naginata" is to improve the development and debugging processes of quantum circuits by introducing dynamically scalable compositions for common operations such as: the adder, multiplier, digital comparator (comparison operator), etc., turning them into building blocks for quantum programs, as well as providing a stable platform for creating more of these compositions. This way, our quantum synthesizer is opening an opportunity to implement quantum algorithms using higher-level commands. The programmer could implement a quantum algorithm with these generic blocks, and the quantum synthesizer would create a suitable circuit for this algorithm, in a format that is supported by the chosen quantum computation environment. With the help of simple command logging and using coding for building quantum circuits, this approach has the potential to significantly simplify and enhance the development and debugging processes of quantum circuits. The proposed approach for implementing quantum algorithms could have a potential application in the field of machine learning. In this regard, we provided an example of creating a circuit for training a simple neural network. Neural networks have a significant impact on the technological development of the transport and road complex, and there is a potential for improving the reliability and efficiency of their learning process by utilizing quantum computation.

Keywords— quantum synthesizer, quantum circuit synthesis, quantum machine learning, quantum algorithms, neural networks, quantum computing, Grover's algorithm

I. INTRODUCTION

For research in the field of quantum informatics, it is considered promising to solve the problem of comparing a large number of digits in order to find the maximum or minimum using the properties of Grover's algorithm. This will significantly reduce the number of qubits used and the number of iterations required to solve this type of problem.

The use of the Grover algorithm for machine learning is a promising direction in the development of quantum computing, since this approach can reduce the number of epochs (iterations of the learning process) required to train a neural network.

The application of Grover's quantum algorithm for machine learning can significantly speed up the process of training large neural networks.

The method of finding solutions to equations or inequalities with many unknowns using the Grover search algorithm can be useful in the areas of mathematical modeling and analysis of the response of complex systems to changing operating conditions by identifying design defects that are difficult to predict. In addition, such techniques can be useful for improving the efficiency and accuracy of decision support systems and the development of artificial intelligence.

The goal of project "Naginata - Quantum Circuit Synthesizer" [1] is to create a prototype system for synthesizing complex quantum circuits. One of the applications of this system will be in parametric synthesis of quantum circuits for finding the weights of a neural network (perceptron) of a given topology. Quantum circuits synthesized by the program are exported to a *.qasm text file in the form of OPENQASM 2.0 code, compatible with IBM Quantum - a cloud quantum computing environment.

To create quantum circuits that implement complex algorithms, a large number of available quantum bits is required for building the circuit. In this regard, the IBM Quantum cloud service was chosen as a platform for debugging [2]. IBM Quantum allows users to perform quantum computing tasks on state-of-the-art quantum computer and quantum simulator prototypes.

Since all current generation quantum computing environments are significantly limited by the number of available quantum bits, some circuit optimization is often necessary to demonstrate complex circuits. One of the methods for optimizing circuits is to use quantum registers with the minimum required size. Thus, the synthesized circuits of typical blocks included in a quantum algorithm must change, based on the size of these registers.
This paper demonstrates the circuits, which lay in the basis of algorithms, created to scale these circuits, depending on the input parameters. For clarity, these circuits were implemented in the “Quirk” quantum simulator, which also supports a certain set of typical operations.

II. MODELS AND METHODS

Adder circuit

IBM Quantum Composer currently does not have built-in methods for performing arithmetic operations, therefore they need to be created. It is necessary to form scalable circuits for the adder, multiplier, and comparison operator. These operations must support the generation of their reversed forms - operations with a reverse sequence of logic gates. This is required for utilizing the reversibility principle in synthesized quantum circuits. As an example, reversed circuits are necessary for completing the oracle function in our implementations of Grover’s algorithm [3, 7].

The adder circuit, which does not require auxiliary qubits (ancilla) was taken from the built-in example in the Quirk quantum simulator [4].

The multiplication algorithm is based on the principle of binary long multiplication, combined with the adder (Fig. 1). Qubit multiplication is performed using Toffoli gates, the product is written to ancilla qubits, then the adder transfers ancilla values to the resulting product register, after which, using the reversibility principle, the ancilla register is cleared for later use. The quantum synthesizer implements parametric QASM code generation depending on the input register qubits and their size. The “multiplier_asymmetric” function generates circuits for multiplying different sized registers.

Multiplier circuit

In the process of calculations, it often becomes necessary to use auxiliary qubits, known as ancilla, for intermediate calculations, the values of ancilla qubits are later cleared by utilizing the reversibility principle. So, in order to reduce the number of quantum bits used, we place ancilla qubits in a separate register, which is shared by sequential operations. When the circuit is built (Fig. 3), this register is added to the circuit using the “plot_ancilla()” method applied to the exported object of the “Circuit” class. In our synthesizer, ancilla qubits are declared by assigning the value “True” to their “is_ancilla” property.

Digital comparator circuit

To compare values of different registers, the circuit of a classical digital comparator [5, 6] was interpreted as a quantum circuit (“if_equal” function) (Fig. 4).

Multiple-Control Toffoli gate circuit

To change the sign of the target states in the quantum register allocated for Grover’s algorithm, a Pauli-Z gate controlled by a set of qubits is needed. It can be obtained by adding a Hadamard gate on each side of a Toffoli gate’s target element (Pauli-X).

Usually, to create an element with a large number of control qubits, a set of Toffoli gates with ancilla qubits as a buffer is used (Fig. 5).
Our “multi_control_gate_3cx” circuits are generated similar to this concept, but they use a 4-qubit Toffoli implementation (Fig. 6) to reduce the number of required ancilla qubits.

The standard Toffoli gate has 2 control qubits. There is a Toffoli implementation (Fig. 6) with three control qubits. This implementation is obtained by visualizing the “mct” function for 4 qubits from the “Qiskit” framework [8, 9]. In “Naginata”, it is created using the ”Toffoli_4q” function, or by adding a “GName.CCCX” logic gate to a circuit.

Figure 6. 4-qubit Toffoli circuit

III. RESULTS

Example use case scenario of the described functions in development of a quantum algorithm (on the example of a perceptron training circuit)

As an example, these synthesis methods were used in an algorithm to find weights suitable for training the neural network on Fig. 7.

Now, parametric versions of the described circuits can be used as building blocks for building more complex algorithms. Let’s look at an example of finding weights in a statically defined neural network with the topology described on Fig. 7. It is implemented in the file “qnn_static_test.py”.

The goal of training this neural network is to find the coefficients Wi that satisfy inequality (1):

\[(I_1w_1 + I_2w_2)*w_3 \geq Ac.\]  \hspace{1cm} (1)

To simplify the analysis of the algorithm’s results, we implement only condition (2) [10-12]:

\[(I_1w_1 + I_2w_2)*w_3 = Ac.\]  \hspace{1cm} (2)

TABLE I. The order of splitting the measured bit strings into W values

| measured | w_1 | w_2 | w_3 |
|----------|-----|-----|-----|
| 010010   | 01  | 00  | 10  |
| 011100   | 01  | 11  | 00  |
| 100001   | 10  | 00  | 01  |
| 110100   | 11  | 01  | 00  |

At the input of the algorithm, the values are: \(I_1 = 11_{10}; I_2 = 10_{10}; Ac = 000110_{2} = 6_{10}\).

At the output of the algorithm, the following values were obtained as solutions: 010010, 011100, 100001, 110100. They correspond to peaks in the diagram on Fig. 8. Based on the order in which the registers are defined for this circuit in file “qnn_static_test.py” and the bit numbering order in the IBM Quantum environment, the obtained values should be divided into equal 2-bit registers corresponding to desired values of \(w_1, w_2, w_3\), in the manner indicated in Table I.

IV. CONCLUSIONS

A study was made of the possibility of developing and applying quantum algorithms in advanced software environments for modeling quantum computing in order to increase the efficiency of calculations related to the processing of large data arrays in the transport and road sphere. On the basis of the data obtained, a new oracle function has been developed that illustrates the possibility of using the Grover algorithm to solve particular problems. For the practical implementation, the Quirk quantum circuit simulator was chosen, which has an extended set of sensors that allow to visualize the transformations occurring in quantum circuits and a large number of complex quantum gates, which greatly simplifies the construction of new quantum circuits on their basis [10-17].
Demonstrated quantum circuits, lay in the basis of algorithms for scaling these circuits, depending on their input parameters. These algorithms were used in our quantum synthesizer prototype, called “Naginata”. For clarity, these circuits were implemented in the Quirk quantum simulator, which also supports a certain set of typical operations.

As a possible application example, of the developed synthesizer, the presented methods were used in an algorithm that finds weights suitable for training an example neural network (perceptron) on Fig. 7. The distribution diagram of the output values, which was obtained after running the synthesized circuit on IBM’s 100-qubit quantum simulator “simulator_mps” is provided on Fig. 8.

The prototype of the quantum synthesizer was developed with the possibility of its further modification and expansion of capabilities by adding new functions that implement typical operation blocks.

Among the ways of further developing this project, it is important to note the development of more advanced versions of the described typical blocks that require a minimal number of operations and ancilla qubits, as well as the addition of new blocks to expand the range of applications of the quantum synthesizer. This would make it possible to form necessary tools for the transition from creating quantum circuits at the level of logic gates to quantum programming using pre-prepared compositions of these logic gates.

The use of computational advantages of quantum computers in machine learning could significantly optimize artificial intelligence models and improve the accuracy of their training, which in turn would make these models more reliable and versatile. In particular, this can increase the speed and accuracy of object recognition systems that are actively being introduced into unmanned vehicles and driver assistance systems, as well as many other promising developments in the transport and road complex and other spheres of economy.

REFERENCES

[1] C. Pronin, Naginata-QCS. 2022. Accessed: Sep. 20, 2022. [Online]. Available: https://github.com/MexIvanov/Naginata-QCS.

[2] IBM Quantum Computing, Oct. 01, 2015. URL: https://www.ibm.com/ibm.com/quantum (accessed Sep. 18, 2022).

[3] L. K. Grover, “A fast quantum mechanical algorithm for database search,” Proceedings of the Annual ACM Symposium on Theory of Computing, vol. Part F129452, pp. 212-219, 1996, doi: 10.1145/237814.237866.

[4] Y. Hong, S. Jeon, S. Park and B. -S. Kim, "Implementation of an Quantum Circuit Simulator using Classical Bits," 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2022, pp. 472-474, doi: 10.1109/ARIA542R2.2022.9870019.

[5] R. Escobar, L. M. Prócel, L. Trojman, M. Lanuzza and R. Tacho, "High-Speed and Low-Energy Dual-Mode Logic based Single-Clack-Cycle Binary Comparator," 2021 IEEE 12th Latin America Symposium on Circuits and System (LASCAS), 2021, pp. 1-4, doi: 10.1109/LASCA51355.2021.9459151.

[6] Gao, M., Wang, J., Fang, S., Nan, J., and Daming, L. "A new nano design for implementation of a digital comparator based on quantum-dot cellular automata," International Journal of Theoretical Physics 60,7 (2021), pp. 2358-2367.

[7] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition. New York: Cambridge University Press, 2010. ISBN 978-1-107-00217-3 Hardback.

[8] E. h. Shaik and N. Rangaswamy, "Implementation of Quantum Gates based Logic Circuits using IBM Qiskit," 2020 5th International Conference on Computing, Communication and Security (ICCCS), 2020, pp. 1-6, doi: 10.1109/ICCSS49678.2020.9277010.

[9] H. Tian and Z. Chu, "A Novel Toffoli Gate Design Using Quantum-dot Cellular Automata," 2021 China Semiconductor Technology International Conference (CSTIC), 2021, pp. 1-3, doi: 10.1109/CSTIC52283.2021.9461426.

[10] C. B. Pronin, O. I. Maksimychev, A. V. Ostrooukh, A. V. Volosova, and E. N. Matukhina, “Creating Quantum Circuits for Training Perceptron Neural Networks on the Principles of Grover’s Algorithm”, 2022 Systems of Signals Generating and Processing in the Field of on Board Communications, SOSG 2022 - Conference Proceedings, pp. 1-5, 2022. doi: 10.1109/IEEECONF53456.2022.9744279.

[11] N. H. Nguyen, E. C. Behrman, M. A. Moustafa and J. E. Stock, "Benchmarking Neural Networks For Quantum Computations," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 7, pp. 2522-2531, July 2020, doi: 10.1109/TNNLS.2019.2933394.

[12] X. Hou, "Research of model of Quantum Learning Vector Quantization Neural Network," Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, 2011, pp. 3893-3896, doi: 10.1109/EMEIT.2011.6023910.

[13] S. Adarash and M. Möller, “Resource Optimal Executable Quantum Circuit Generation Using Approximate Computing," 2021 IEEE International Conference on Quantum Computing and Engineering (QCE), 2021, pp. 225-231, doi: 10.1109/QCE52317.2021.00040.

[14] Y. Kondo, R. Mori and R. Movassagh, "Quantum supremacy and hardness of estimating output probabilities of quantum circuits," 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS), 2022, pp. 1296-1307, doi: 10.1109/FOCS52979.2021.00126.

[15] J. Kusyk, S. M. Saeed and M. U. Uyar, "Survey on Quantum Circuit Compilation for Noisy Intermediate-Scale Quantum Computers: Artificial Intelligence to Heuristics," IEEE Transactions on Quantum Engineering, vol. 2, pp. 1-16, 2021, Art no. 2501616, doi: 10.1109/TQE.2021.3068355.

[16] N. Thumwanit, C. Lortaraprasert, H. Yano and R. Raymond, "Trainable Discrete Feature Embeddings for Quantum Machine Learning," 2021 IEEE International Conference on Quantum Computing and Engineering (QCE), 2021, pp. 479-480, doi: 10.1109/QCE52317.2021.00087.

[17] Q. Wang, R. Li and M. Ying, "Equivalence Checking of Sequential Quantum Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 9, pp. 3143-3156, Sept. 2022, doi: 10.1109/TCAD.2021.3117506.