Circuit Centric Quantum Architecture Design

Preprint · December 2019

DOI: 10.13140/RG.2.2.19404.77446

5 authors, including:

Utkarsh Azad
International Institute of Information Technology, Hyderabad
6 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Ankit Papneja
Indian Institute of Technology (ISM) Dhanbad
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Rakesh Saini
Indian Institute of Technology (ISM) Dhanbad
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Bikash K. Behera
Bikash's Quantum (OPC) Pvt. Ltd.
132 PUBLICATIONS **439** CITATIONS
SEE PROFILE

Circuit Centric Quantum Architecture Design

Utkarsh@,[1, *] Ankit Papneja@,[2, †] Rakesh Saini@,[2, ‡] Bikash K. Behera,[3, 4, §] and Prasanta K. Panigrahi[4, ¶]

[1]*Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology Hyderabad, Hyderabad 500032, Telangana, India*

[2]*Department of Physics, Indian Institute of Technology, (ISM), Dhanbad 826004, Jharkhand, India*

[3]*Bikash's Quantum (OPC) Pvt. Ltd., Balindi, Mohanpur 741246, Nadia, West Bengal, India*

[4]*Department of Physical Sciences, Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India*

With the development in the field of quantum physics, several methods for building a quantum computer have come up. One such method is the use of superconducting qubits. The interactions amongst qubits are of greater importance for the designing of a sophisticated quantum computer. The paper discusses the dependence of the circuit size, and circuit depth on the interaction and connection between different qubits present in quantum hardware. Here, we present a procedure which helps one in determining the optimal interactions between different qubits of a quantum chip to execute a given circuit in the most efficient way possible. In this paper, we illustrate it with an example of a 5-qubit structure. In general, this procedure can be used for any arbitrary user-defined circuit. Given the allowed interactions, one can accurately determine the configuration for which the circuit runs in the least number of clock cycles with the lowest gate operations and noise-based errors.

I. INTRODUCTION

The history of quantum computers[1] go back to 1980s when Paul Benioff came up with an idea of a Turing machine that uses the properties of quantum mechanics. At a later time, it was proposed that such a machine can be used for performing computational tasks which are out of the reach of a classical computing device. Since then, several methods for the construction of quantum computers have been introduced over the years, such as ion traps, photonic, NMR, atomic, NV centre, topological etc. Among the above potential hardware platforms, superconducting quantum circuits have become the most popular for constructing a scalable quantum computing system. Here, the information is stored in quantum degrees of freedom of nanofabricated, anharmonic oscillators constructed from superconducting circuit elements. The superconducting qubits[2–5] are macroscopic in size and lithographically defined as compared to other technologies where information is stored in microscopic quantum systems. Among several other advantages of superconducting qubits, it is the ease of production of such qubits. Major tech giants like Google, IBM, Intel etc have developed their quantum computers based on the superconducting qubits technology, owing to the fact that they have very high compatibility with the existing fabrications. These are known to be the most mature qubit technology at the present date. The areas of improvement in this field are already known to us. All current architecture designs require a 2D arrangement[6,7] of qubits with the nearest neighbour interactions, compatible with powerful quantum error correction using the surface code. Several realistic experiments in the field of quantum computation, including quantum simulation[8–11] developing quantum algorithms[12–14], testing of quantum information-theoretical tasks[15,17], quantum cryptography[18], quantum error correction[19,20] have been performed owing to the development of quantum chips.

With the advancements in the field of quantum technology, several chips have been developed and made available to the public via cloud-based services[21–23]. However, except for the few free services, currently, users are charged based on the execution time of their circuits. As we explain in Section II, for any given circuit, the coupling present in the hardware determines the execution time by transforming crucial factors like depth and size. Hence, for the noisy intermediate-scale quantum (NISQ) processors these factors determine their performance.

It is speculated that, in coming years, with the involvement of industry, more cloud-based services will be made available to the public. Hence, economical and practical usage of this wide availability of the choices would help users in knowing beforehand the most optimal hardware for their circuit. In this paper, we discuss a novel procedure to allow the user to select the hardware with the most optimal permissible interactions that allows the most efficient run for the given circuit.

Structure – In Section II, we present the general procedure. Section III illustrates the procedure with the help of an example on a range of 5-qubit hardware (Fig. 1) for a chosen 5-qubit circuit (Fig. 7). In Section IV, we discuss and interpret our results.

II. PROCEDURE

Given a quantum hardware, and details about its topology, one can know all permissible and valid interactions between the pairs of qubits. If each qubit q_i is

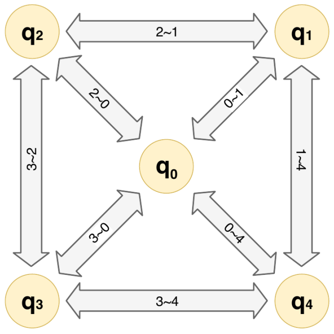

FIG. 1. A general coupling map of a 5-qubit quantum hardware. Here, bidirectional arrows represent all the permissible interaction between the qubits.

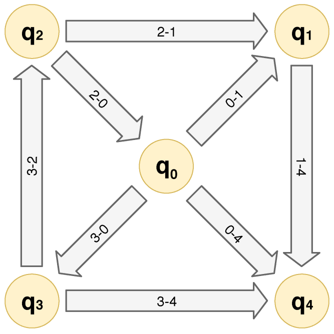

FIG. 2. A coupling map example for the 5-qubit quantum hardware. Here, directional arrows: $q_i \to q_j$ represent the permissible interaction with q_i as the source qubit and q_j as the target qubit.

represented by its label i, we can store this information as list containing the set $\{i, j\}$ if the interaction $q_i \sim q_j$ is permissible (Fig. 1) i.e., exactly one of the interactions $q_i \to q_j$, or $q_j \to q_i$ is the valid one.

Next we define a decision variables $x_{ij} \in \{0, 1\}$. If x_{ij} is 1 then the interaction $q_i \to q_j$ is valid else the interaction $q_j \to q_i$ is valid (Fig. 2). Hence, there is a constraint between the decision variables x_{ij} and x_{ji}, i.e., exactly one of them is true in a given hardware layout. Now, consider the following matrix made from encompassing all possible x_{ij} for a given hardware:

$$C = \begin{pmatrix} x_{00} & x_{01} & \cdots & x_{0(n-1)} \\ x_{10} & x_{11} & \cdots & x_{1(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n-1)0} & x_{(n-1)1} & \cdots & x_{(n-1)(n-1)} \end{pmatrix} \tag{1}$$

In C, all $x_{ii} = 0$, and $\forall i, j$ such that $q_i \nsim q_j$. Also due to our imposed constraints, exactly one of the decision variables – x_{ij} or x_{ji} is valid at a time. Hence, this matrix C is analogous to an adjacency matrix, and it represents the couplings for a given hardware and is called a coupling map. All the non-zero elements x_{ij} in the coupling map represent the allowed interaction on

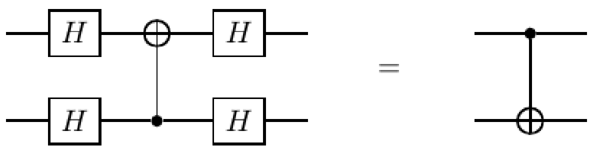

FIG. 3. If x_{ij} is false, and $q_i \sim q_j$ for a quantum hardware, then the direction of controlled gates $CX[i, j]$ (right) can be reversed by replacing it with its equivalent circuit (left)[18].

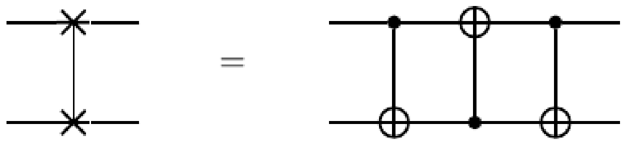

FIG. 4. Standard implementation of $SWAP$ instruction involves decomposing it into three CNOTs gates.

the hardware where q_i is the source qubit and q_j is target qubit.

Now, any quantum circuit QC can be represented as a temporal sequence of instructions I_{set}. Each instruction $I \in I_{set}$ is either a unitary gate $U[i]$ being applied on the qubit q_i or a controlled gate $CX[i, j]$ applied on qubit q_j with qubit q_i as a control. If we do not consider the tedious task of circuit optimization yet, we can see that all the unitary instructions that are unitary gates are by default valid because they can be directly decomposed into basis gates that implementable on the hardware. On the other hand, the controlled instruction $CX[i, j]$ are valid only if x_{ij} is 1 in the coupling matrix C of given hardware. To fix the invalid controlled instruction $CX[i, j]$ we have the following options: (1) if x_{ji} is valid then perform the procedure listed in Fig. (3) to invert the instruction to $CX[j, i]$ else (2) perform swaps of either qubit q_i with q_k, qubit q_j with qubit q_l or both depending upon which of the following decision variables: $\{x_{kj}, x_{il}, x_{kl}\}$ is true (Fig. 4). In general, the latter option is implemented via randomized heuristics and induces stochasticity in the process. This fixing step is usually done by the compiler and usually results in the addition of extra gates which increases circuit depth, which is undesirable.

In Ref. (5), we list down the pseudocode for using this procedure to make any circuit compatible for given quantum hardware. Step by step procedure is as follows: We generate all possible bitstrings of length $len(I)$, where I is the list of qubit indices among which interaction is possible. We then initialize the element in the list *coupling* of length $2^{len(I)}$ with $\mathbb{O}_{N,N}$, where N is the total number of qubits present in the hardware, and $\mathbb{O}$ is an all-zeros matrix. Now, using every bitstring, we initialize decision variables x_{ij} to generate coupling matrixes which are stored in the list *coupling*. Next, we find the non-zero elements for every coupling matrix to generate the

Algorithm 1: Procedure

```
input  : List of qubit indices that are permitted to
         interact: $I = [(q_i, q_j), \ldots, (q_k, q_l)]$
         Compiler: $C$
         Quantum Circuit: $QC$
         Number of qubits: $N$
output: Compiled quantum circuit: $compiledCircuit$
         Properties: $circuitProp$
1  combinations ← $\{0,1\}^{len(I)}$;
2  couplings ← $[\mathbb{O}^1_{N,N}, \ldots, \mathbb{O}^{2^{len(I)}}_{N,N}]$;
3  foreach $i$ ← combinations do
4      couplings $[index(i)]$ =
        GenerateCouplingMatrix($I, i$);
5  end
6  foreach $i$ ← couplings do
7      coupleList ← NonZero($i$) ;
8      coupleMap ← GenerateCouplingMap(coupleList) ;
9      compiledCircuit ← Compiler($QC$, coupleMap);
10     circuitProp ← GenerateDAG(compiledCircuit);
11     data ← Push(compiledCircuit, circuitProp);
12 end
13 return data;
```

FIG. 5. Pseudocode for our procedure

coupling map C. This C is then given to the compiler along with the circuit QC for compilation. Various quantum computing libraries from the companies like IBM[21], Rigetti[?] , Xanadu[24] provide built-in compilers.

Here, we have used the circuit transformation functionalities that include transpiler, circuit_to_dag provided in the IBM Qiskit. In order to gather information about more circuit properties such as size, depth, etc, we convert our quantum circuit into a directed acyclic graph (DAG)[25]. In this representation, each node corresponds to a gate operation, and each directed edge represents the qubit dependency. Fig. (6) shows an example DAG for the truncated original circuit. In a given DAG, we define the first layer to be a set of all nodes that are not preceded by any other node. Similarly, we can define the second layer to be the set of all nodes that are preceded only by the nodes of the first layer. By definition, the total number of nodes in DAG is equal to the size of the circuit, and the number of layers provides the depth of the circuit. The topological sorting for the DAG gives us a linear ordering of operations (i.e. nodes) such that for every qubit dependency represented by the directed edge $n_i \rightarrow n_j$, node n_i comes before n_j. In general, this can be used to decompose the circuit into independent blocks for circuit optimization or parallel execution on the hardware.

III. ILLUSTRATION

To illustrate the procedure, we consider a 5-qubit circuit as shown in Fig. (1). We illustrate the circuit in

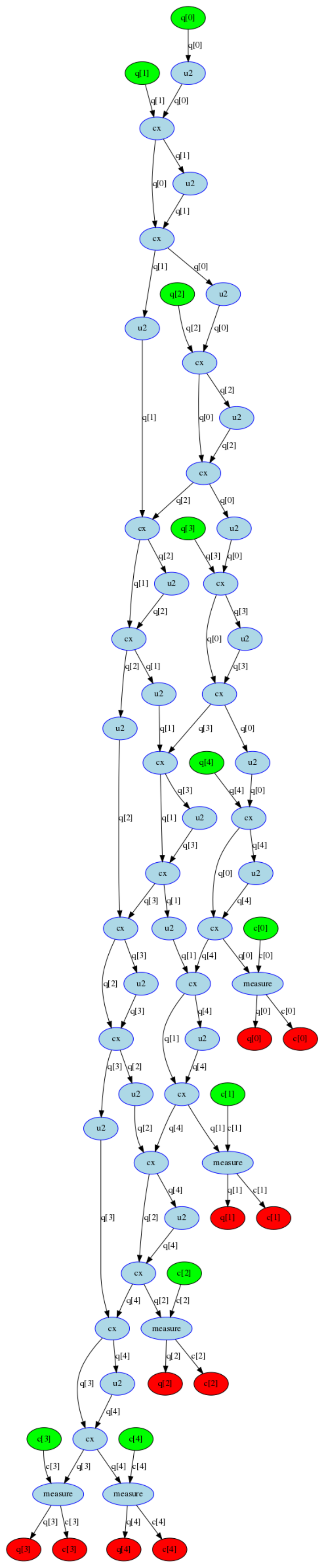

FIG. 6. DAG for the truncated original circuit. In this representation, each node represents a gate operation, and each directed edge represents the qubit dependency.

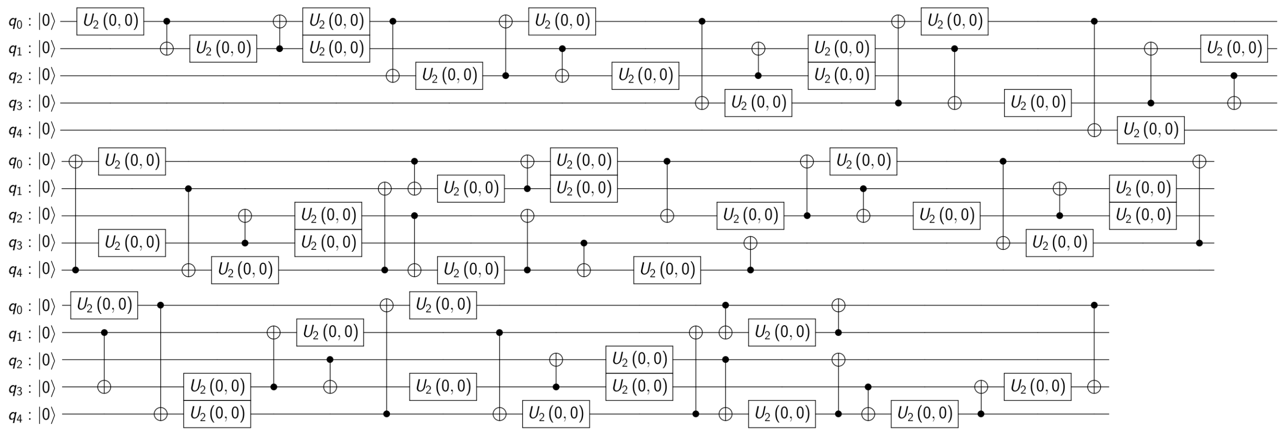

FIG. 7. Original circuit for coupling C_0, i.e., the ideal all-to-all connectivity configuration, in a folded representation. This circuit has a depth of 46 with 43 unitary gates and 43 controlled-not gates. For concise representation, the classical register and measurements are not shown.

Coupling no.	Circuit size	U gates	CX gates	Circuit depth
0	91	43	43	46
44	144	87	51	64
108	143	86	51	64
138	143	86	51	64
142	142	85	51	64
172	143	86	51	64
176	142	85	51	64
202	146	89	51	64
206	145	88	51	64
236	142	85	51	64
240	141	84	51	64

TABLE I. Coupling numer, circuit size, number of gate operations for the transformed circuits with the least depth.

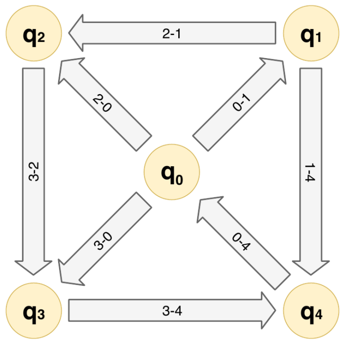

FIG. 8. Coupling map C_{240} for a 5-qubit hardware. For this coupling: $x = [1, 1, 1, 0, 1, 1, 1, 1]$.

Fig. 7 as an input along with the list of eight permissible interactions between the qubits:

$$I = [(0,1),(0,2),(0,3),(0,4),(1,2),(1,4),(2,3),(3,4)] \tag{2}$$

This means, in total there are $2^8 = 256$ different configurations possible for the 5-qubit hardware. In the context of circuit execution, we want to minimize on two major cost inducing factors: time, and noise-based errors. In general, the former is represented by the circuit depth, i.e., the number of clock cycles utilized in the parallel execution of the circuit on a quantum processor, whereas, the latter one is represented by both depth, and circuit size. Hence, we aim to find the optimum configuration, i.e., the transformed circuit for this configuration either has the lowest depth, or it has the lowest size.

In our case, the input circuit (Fig. 7) is based on the template which contains the controlled-not operations for all qubit pairs $CX[i, j]$, $\quad \forall i, j \in \{0, \ldots, N\}$. In principle, one can visualize an input circuit generated randomly using a template with any desired property defining the application of either unitary $U[i]$ or controlled $CX[i, j]$ operations. Hence, the circuit transformation observed for the configuration is directly dependent on this template.

As explained in the previous section, in the QISKit platform, we can calculate the size, depth etc of the circuit by converting it into a DAG. However, the depth, in this case, fluctuates with every execution instance because their circuit transformer (also called transpiler) utilizes stochastic swaps. To overcome this, we seeded all the stochastic steps in transpilation process. The seed value was chosen such that it maximizes the number of equivalent optimal configurations, and minimizes the circuit depth. For the initial circuit (Fig. 7), the circuit size was 91, depth was 46, and the number of controlled unitary instructions were 43 each. However, in the case of transformed circuits, the least achievable depth was 64. We list down the data regarding the properties of transformed circuit in Table: (I), with respect to the coupling number 0, which corresponding to the most ideal all-to-all connectivity configuration. The coupling map C_{240} corresponding to the coupling number 240 for which transformed circuit has the least size and depth is:

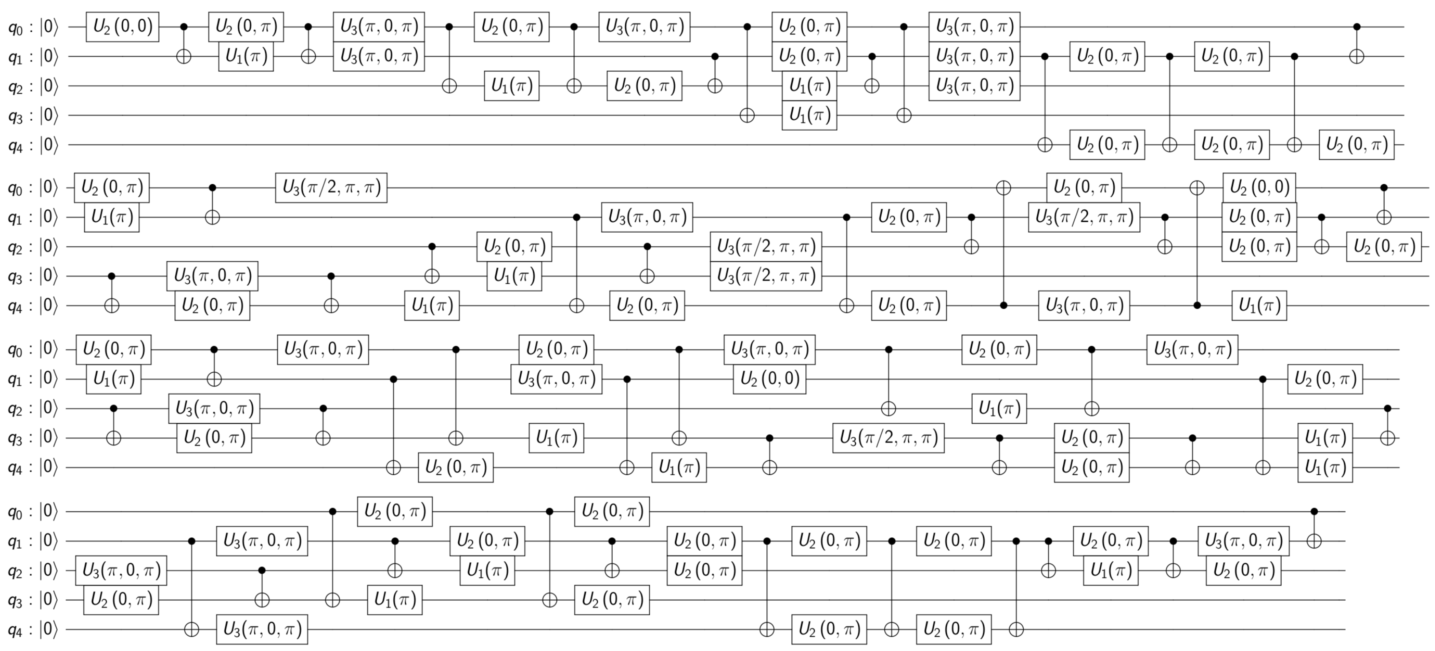

FIG. 9. Transformed circuit for the coupling C_{240} with $x = [1, 1, 1, 0, 1, 1, 1, 1]$, in a folded representation. This circuit has a depth of 64 with 84 unitary gates and 51 controlled-not gates. For concise representation, the classical register and measurements are not shown.

Circuit depth	Equivalent couplings	Average circuit size
64	10	143
65	20	146
66	27	148
67	21	150
68	26	151
69	36	153
70	30	154
71	29	156
72	20	158
73	16	160
74	13	162
75	6	164
76	2	164

TABLE II. The number of couplings for which the depth of transformed circuit, along with their average circuit sizes (rounded up to nearest integer).

$$C_{240} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3}$$

In Fig. (8), we represent a graphical representation based on this coupling map C_{240}. Similarly, we can generate graphical representation for every C_i. To do that we just use the bit string corresponding to the number $(i-1)$ to generate the coupling map C_i. For example, to generate C_{44}, we use $(43)_2 = [00101011]$ where each bit corresponds to the value of the decision variable represented by $x = [x_{01}, x_{02}, x_{04}, x_{04}, x_{12}, x_{14}, x_{23}, x_{34}]$.

IV. CONCLUSIONS

The computational capability of the current generation quantum devices is considerably restricted due to poor qubit quality, limited qubit connectivity and shorter coherence time. Hence, properties like circuit depth and circuit size become the limiting factor for the circuits one would want to run. In the Tables. I and II we have accumulated our results for running the circuit in Fig. 7 for the 5-qubit hardware in Fig. 1. We found that for any given circuit, multiple couplings could be equivalently optimal. For example, the depth of the transformed circuit in 10 different couplings comes out to be 64. Moreover, the average circuit size also came out to be the least for this group of couplings. However, using a stricter scheme for comparison, it is possible to rank different couplings within an equivalent circuit depth too. To do this, we consider circuit size and the number of unitary or controlled gates for the transformed circuits. For example, in our case, on considering circuit size, the most optimal coupling within the above group comes out to be C_{240} with the least circuit size of 141 operations.

In general, the circuit execution on NISQ devices is affected by three kinds of noise[26]: (1) Memory errors, (2) Gate implementation errors and (3) Measurement errors. Their effect on the result of circuit execution can be understood in the following way. Firstly, memory errors are due to thermalization, amplitude decay and decoherence of qubits. In the parallel execution of a circuit on a quantum chip, these affect the state of the system at the end of every clock cycle. Therefore, the overall effect, in this case, is proportional to the circuit depth. However, currently, only sequential execution of instructions is supported by these devices. In this case, circuit size determines the overall effect of these errors. Similarly, the gate implementation errors are due to the imprecision and systematic errors[27] introduced in designing microwave pulses. In general, the unitary operation is based on realizing single-qubit rotation using these pulses, and controlled operation is based on cross-resonance effect. Hence, the effect of these errors is also directly proportional to circuit size i.e., the total number of gate instructions. Lastly, the measurement errors are due to depolarization of qubits during the read-out process. This is analogous to a bit-flip error in classical computers. Therefore, the success probability of meaningful circuit execution decreases down with higher circuit depth and circuit sizes.

DATA AVAILABILITY

The code created to run these simulations and related supplementary data could be made available to any reader upon reasonable request.

ACKNOWLEDGMENTS

U., A.P., and R.S. would like to thank IISER Kolkata and Bikash's Quantum (OPC) Pvt. Ltd. for providing hospitality during the course of the project work. B.K.B. acknowledges the support of Institute fellowship provided by IISER Kolkata.

* utkarsh.azad@research.iiit.ac.in; @The authors have equally contributed to the manuscript

† papneja.ankit55057@gmail.com; @The authors have equally contributed to the manuscript

‡ rs474390@gmail.com; @The authors have equally contributed to the manuscript

§ bkb18rs025@iiserkol.ac.in

¶ pprasanta@iiserkol.ac.in

[1] Quantum computing. Retrieved from `https://en.wikipedia.org/wiki/Quantumcomputing` (2019)

[2] A. O. Niskanen, k. Harrabi, F. Yoshihara, Y. Nakamura, S. Lloyd., and J. S. Tsai, Quantum Coherent Tunable Coupling of Superconducting Qubits, Science, **316**, 723 (2007).

[3] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, A quantum engineers guide to superconducting qubits, Appl. Phys. Rev. **2**, 021318 (2019).

[4] J. Majer, J. M. Chow , J. M. Gambetta , J. Koch, B. R. Johnson, J. A. Schreier, R. J. Schoelkopf, Coupling superconducting qubits via a cavity bus, Nature **449**, 443 (2007).

[5] Y. Makhlin, G. Scohn, and A. Shnirman, Josephson-junction qubits with controlled couplings, Nature **398**, 305 (1999).

[6] H. Mukai, K. Sakata, S. J. Devitt, R. Wang, Y. Zhou, Y. Nakajima, and J. S. Tsai, Pseudo-2D superconducting quantum computing circuit for the surface code: the proposal and preliminary tests, arxiv:1902.07911 (2019).

[7] S. Lloyd, and B. M. Terhal, Adiabatic and Hamiltonian computing on a 2D lattice with simple two-qubit interactions, New J. Phys. **2**, 023042 (2016).

[8] D. Aggarwal, S. Raj, B. K. Behera, and P. K. Panigrahi, Application of quantum scrambling in Rydberg atom on IBM quantum computer, arXiv:1806.00781 (2018).

[9] P. K. Vishnu, D. Joy, B. K. Behera, and P. K. Panigrahi, Experimental demonstration of non-local controlled-unitary quantum gates using a five-qubit quantum computer, Quantum Inf. Process. **17**, 274 (2018).

[10] G. R. Malik, R. P. Singh, B. K. Behera, and P. K. Panigrahi, First Experimental Demonstration of Multi-particle Quantum Tunneling in IBM Quantum Computer, DOI: 10.13140/RG.2.2.27260.18569 (2019).

[11] Manabputra, B. K. Behera, and P. K. Panigrahi, A Simulational Model for Witnessing Quantum Effects of Gravity Using IBM Quantum Computer, arXiv:1806.10229 (2018).

[12] D. G. Martin, and G. Sierra, Five Experimental Tests on the 5-Qubit IBM Quantum Computer, J. App. Math. Phys. **6**, 1460 (2018).

[13] R. Jha, D. Das, A. Dash, S. Jayaraman, B. K. Behera, and P. K. Panigrahi, A Novel Quantum N-Queens Solver Algorithm and its Simulation and Application to Satellite Communication Using IBM Quantum Experience, arXiv:1806.10221 (2018).

[14] A. Dash, S. Rout, B. K. Behera, and P. K. Panigrahi, Quantum Locker Using a Novel Verification Algorithm and Its Experimental Realization in IBM Quantum Computer, arXiv preprint arXiv:1710.05196 (2017).

[15] A. R. Kalra, N. Gupta, B. K. Behera, S. Prakash, and P. K. Panigrahi, Demonstration of the no-hiding theorem on the 5-Qubit IBM quantum computer in a category-theoretic framework, Quantum Inf. Process. **18**, 170 (2019).

[16] A. Baishya, S. Sonkar, B. K. Behera, and P. K. Panigrahi, Demonstration of Quantum Information Splitting Using a Five-qubit Cluster State: An IBM Quantum Experience, DOI: 10.13140/RG.2.2.21435.05925 (2019).

[17] D. Alsina, and J. I. Latorre, Experimental test of Mermin inequalities on a five-qubit quantum computer, Phys. Rev. A **94**, 012314 (2016).

[18] B. K. Behera, A. Banerjee, and P. K. Panigrahi, Experimental realization of quantum cheque using a five-qubit quantum computer, Quantum Inf. Process. **16**, 312 (2017).

[19] J. Roffe, D. Headley, N. Chancellor, D. Horsman, and V. Kendon, Quantum Sci. Technol. **3**, 035010 (2018).

[20] R. K. Singh, B. Panda, B. K. Behera, and P. K. Panigrahi, Demonstration of a general fault-tolerant quantum error detection code for (2n+ 1)-qubit entangled state on IBM 16-qubit quantum computer, arXiv preprint

arXiv:1807.02883 (2018).

[21] Qiskit: An Open-source Framework for Quantum Computing, 10.5281/zenodo.2562110 (2019).

[22] Amazon Web Services, Inc. (2019). Amazon Braket - Amazon Web Services. Available at: https://aws.amazon.com/braket/

[23] R. Smith, M. J. Curtis and W. J. Zeng, A Practical Quantum Instruction Set Architecture, arXiv:1608.03355 (2016).

[24] J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, and N. Killoran, PennyLane: Automatic differentiation of hybrid quantum-classical computations, Ville Bergholm, arXiv:1811.04968 (2018).

[25] A. M. Childs and E. Schoute and C. M. Unsal, Circuit Transformations for Quantum Architectures, 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)

[26] H. Chaudhary, B. Mahato, L. Priyadarshi, N. Roshan, Utkarsh, and A. D. Patel, A Software Simulator for Noisy Quantum Circuits, arXiv:1908.05154 (2019)

[27] D. Willisch, M. Nocon, F. Jin, H. De Raedt and K. Michielsen, Gate-error analysis in simulations of quantum computers with transmon qubits, Phys. Rev. A **96**, 062302 (2017).