Minimizing Elementary Gates in IBM Quantum Architectures using Deferred Swap Embedding

Syed Hammad Ahmed
Department of Computer Science
University of Central Florida
Orlando, USA
hammad.ahmed@knights.ucf.edu

Rickard Ewetz
Department of ECE
University of Central Florida
Orlando, USA
rickard.ewetz@ucf.edu

Gita Sukthankar
Department of Computer Science
University of Central Florida
Orlando, USA
gitars@eecs.ucf.edu

Abstract— With the advent of quantum computers, researchers have been exploring effective and efficient ways to embed quantum algorithms into physical quantum circuits. Implementation and execution of a logical quantum circuit on a physical quantum circuit necessitates various constraints to be satisfied. This requires one-to-one logical to physical qubit mappings as well as logical qubits being mapped to physically adjacent qubits for elementary CNOT instructions. In order to successfully execute all program instructions, swap instructions are inserted before CNOT instructions where required. These additional instructions must be minimized in order to ensure fidelity of the overall circuit. In this paper we propose the Deferred Swap Embedding (DSE) technique which ensures a minimal number of swap instructions are used to satisfy the constraints of any generic quantum architecture possessing a small number of qubits. Our solution also identifies the initial mapping that would lead to the minimal overall number of swap instructions. The results show that our proposed approach performs better than the current state-of-the-art solutions.

I. Introduction

Recent technological advancements in the field of quantum computing [1] have led to an increasing amount of research and development in both academia and industry. Several top technology companies have invested in developing more powerful quantum computers. Google recently launched a quantum computer with 72 quantum bits (qubits) [9]. IBM has developed over 2 dozen quantum computers ranging from 1 to 65 qubits [8]. Similarly the most powerful quantum machine at Rigetti is a 31-qubit machine [10]. These companies also provide access to their quantum computers via the cloud that empowers the technology community to run quantum programs and explore the technology.

IBM Quantum Experience (IBM Q) [8] is a cloud-based service that allows users to run programs on various physical IBM quantum computers with different architectures. The architecture of a quantum computer refers to the number of physical qubits and how these qubits are arranged and coupled together physically. A quantum program is a set of instructions that can be realized as a quantum circuit having quantum gates. In order to correctly implement the circuit on a physical quantum computer certain architectural constraints must be satisfied. The widely used CNOT gate is a binary gate which can only be executed if both operand qubits are associated with two connected physical qubits. These associations between the logical and physical qubits are known as qubit mappings and may be visualized as a set of key-value pairs of logical and physical qubits.

In classical computing, any logic circuit may be decomposed into universal logic gates including NAND and NOR gates. Similarly, in the quantum domain CNOT and U gates are widely used as elementary quantum gates. CNOT is a binary gate with source and target qubits; its execution on a given quantum architecture is possible only if the corresponding physical source and target qubits are connected from source to target to form compatible gates, as shown in Figure 2. If the respective physical qubits are not connected, a series of SWAP operations are required to bring the logical source and target qubits adjacent to ensure that these constraints are satisfied. Therefore, to ensure effective execution of a program, all instructions must satisfy these architectural constraints. The problem is that adding each additional instruction adversely affects the fidelity of the program which makes it more error-prone. To ensure the insertion of minimal SWAP gates we must efficiently choose an initial mapping of qubits as well as choose the SWAP operations which would minimize the total SWAP operations required for the execution of the complete program.

We propose the Deferred Swap Embedding (DSE) algorithm which guarantees the minimal number of swaps required for execution of a quantum program. For each of the initial mappings the total cost of making all instructions compatible for execution is calculated. The mapping which results in the minimum total cost is considered as the best initial mapping for the quantum program. The cost refers to the total additional elementary instructions required when SWAP instructions are inserted before incompatible CNOT instructions. While finding the minimal total cost for a mapping, DSE chooses series of swap instructions which minimize the total cost for the next instruction. In this way we *defer* our decision of selecting the best set of swap instructions until the next instruction. Lastly, the optimization module eliminates all redundant instructions which do not affect the overall quantum state. Hence, DSE enhances the compatibility of quantum programs by minimizing the number of swaps while maintaining high fidelity.

II. Preliminaries and Background

A. Quantum Program

A quantum program is a set of instructions which define a quantum circuit using a low-level programming language called quantum assembly language. The instructions, also called quantum gates, are applied on quantum bits called *qubits*. These instructions can be single-qubit, having one qubit as the operand, or multi-qubit, having 2 or more qubits as operands. The qubits that are written in a program are referred as *logical qubits* as compared to *physical qubits* which are physically embedded as part of the hardware of a quantum computer. To execute a quantum program on a quantum computer, each logical qubit must be physically realized as a physical qubit which is referred to as *mapping*.

A.1 Elementary Instructions

In classical computers there are certain *universal* logic gates which means any logic function can be performed with only these gates. Similarly, to implement quantum circuits we require gates not only satisfy universality of gates but also require them to be *error-tolerant*. The Controlled-NOT (CNOT) and single-qubit Pauli gates possess both these properties and are considered to be elementary gates [13].

A.2 SWAP Instruction

The swap instruction is a binary instruction that interchanges the states of the two operand qubits. As it is not part of the IBM QX architecture's set of elementary gates, a swap instruction is decomposed into 7 elementary gates as shown in Figure 3.

B. Quantum Architecture

The architecture of a quantum computer includes all physical implementation details including the number of physical qubits, n, and which qubits are physically connected. This is imperative since in order for instructions involving multiple qubits to be executed on an architecture, these qubits must be physically connected.

B.1 Coupling Map

The coupling map is a graph with physical qubits as nodes and the physical connections between corresponding qubits as edges. The edges need to be directed edges as the elementary gate CNOT is only executable on the quantum machine if its qubits form a directed connection from source to target.

B.2 Mapping

A mapping is a binding of logical qubits to physical qubits; it is required for execution of quantum instructions having those set of logical qubits as operands. We can denote a mapping using the following notation: $m = [0 \to 0, 1 \to 1, 2 \to 3, 3 \to 4, 4 \to 2]$. In this notation, $a \to b$ denotes that logical qubit b is mapped onto physical qubit a. As the physical qubits form a sequence from 0 to $n-1$, where the number of physical qubits n is 5, we can represent a mapping in an abridged notation as $m = [0, 1, 3, 4, 2]$. This 1-to-1 assignment remain unchanged throughtout the execution unless there are swap instructions in the program which interchange the states of the two qubits being swapped.

B.3 CNOT Constraints

A CNOT instruction having logical qubits l_s and l_t as source and target qubits, respectively, must have the corresponding physical qubits $p(l_s)$ and $p(l_t)$ connected in a specific directed way in the architecture i.e. a directed edge from $p(l_s)$ to $p(l_t)$ in the coupling map, as shown in Figure 1. This is referred to as a CNOT constraint and must be satisfied for all CNOT instructions for successful execution. We refer to a quantum instruction satisfying the CNOT constraints as a compatible instruction. Similarly, a program which satisfies CNOT constraints for all instructions is referred to as a *compatable program*.

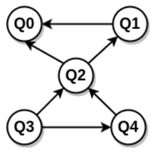


Fig. 1.: Coupling map for IBM QX4 architecture. [11]

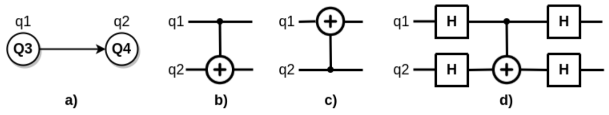


Fig. 2.: Overview of compatible vs. incompatible CNOT instructions. a) CNOT constraint from Q3 to Q4 and logical qubits q1 and q2 mapped respectively b) Compatible CNOT gate c) Incompatible CNOT gate d) Adding 4 Hadamard instructions to make c) compatible.

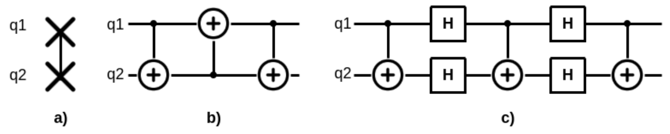


Fig. 3.: Decomposition of a SWAP instruction into elementary instructions. a) Symbol of a SWAP instruction having logical qubits q1 and q2 as operands b) SWAP instruction decomposed as 3 CNOT instructions (2nd CNOT instruction is incompatible) c) SWAP instruction decomposed as 7 elementary instructions having all compatible CNOT instructions.

III. Related Work

Many solutions have been proposed to address the qubit mapping problem. The efficacy of most of these solutions depends primarily on 1) the selection of an initial mapping, and 2) finding the minimal number of required swap operations. Some approaches [4, 6] focus on 5-qubit IBM quantum architectures whereas others [3, 5, 7] suggest generic solutions applicable to architectures possessing an arbitrary number of qubits. The initial mapping formulation suggested by Kole et al. [3] generates random initial mappings using a genetic algorithm which is efficient but does not guarantee an optimal initial mapping. The minimal cost mapping is found by visualizing the quantum program as a weighted directed graph called an incidence graph, then formulating the mapping cost based on the edge weights of the incidence graph and the shortest distances between the mapped physical qubits in the coupling map. Dueck et al. [6] focus on post-mapping optimization of quantum circuits to help further improve the fidelity of the circuit. Our DSE algorithm is inspired by the Deferred-Merge Embedding (DME) algorithm used for minimizing the clock skew in high-performance VLSI architectures [14]. DME finds the optimal wirelength while formulating a zero skew clock tree for a given clock distribution network connection topology. Similarly, we find the optimal set of swap sequences required for a given program, or instruction topology, while satisfying the CNOT constraints of the physical architecture.

IV. Proposed Approach

The proposed solution includes (1) generation of initial mappings through all permutations, (2) the deferred swap embedding (DSE) for finding a minimal cost mapping, and (3) optimization of the resulting circuit for eliminating redundant unary and binary quantum instructions. These steps are described in sections A, C, and E, respectively. Pre-processing is performed before the execution of DSE which includes removing all unary instructions from the input circuit, which will be re-inserted at their respective positions in the post-processing step. Details about other actions performed in pre- and post-processing are described in sections B and D. The overview of the proposed approach is illustrated in Figure 4.

A. Generating Initial Mappings

It is imperative that mappings of logical to physical qubits are done before running a quantum program which includes those logical qubits. This starting assignment, known as initial mapping, must be completed before the execution of the first instruction of the quantum program. An ideal initial mapping is one which results in 0 (zero) cost when DSE is executed. This is only possible if all CNOT instructions satisfy the CNOT constraints. As it is not possible for all quantum programs to have ideal initial mappings therefore we try to find a minimal cost initial mapping which may not necessarily be zero-cost. We generate all $n!$ permutations of mappings e.g. for the IBM QX4 architecture which has 5 qubits we generate $5! = 120$ initial mappings. Then for each initial mapping the DSE algorithm finds the cost of making the program compatible and identifies initial mappings which result in the minimum cost. There can be multiple initial mappings resulting in the same minimal cost therefore we select any one of those initial mappings for the respective quantum program.

B. Pre-processing

Finding the minimal number of additional instructions to ensure all instructions are physically executable is primarily dependent on binary instructions, specifically the CNOT instruction. Unary instructions do not affect the cost of finding additional instructions to ensure CNOT constraints are satisfied. Therefore to simplify the implementation we sift out all unary instructions from our quantum program and apply DSE on the remainder program. Similarly, an integral part of our solution is based on selecting shortest paths from the source to target qubits. Therefore, we pre-calculate the shortest distances using Floyd-Warshall algorithm [12] which finds all-pair shortest distances between all physical qubits by using the coupling map as the input graph. The coupling map directions are not considered and all edges are assigned a weight value equal to 1 - the cost for additional gates, due to the fact that different directions are counted separately when finding the total swap cost of each shortest path.

C. Deferred Swap Embedding

We propose a new technique called *Deferred Swap Embedding* which explores for each CNOT instruction all possible swap sequences that satisfy the CNOT constraints. A swap sequence is a special permutation of the shortest path found between the source and target qubits in which the source and the target qubits must be adjacent. Hence, the decision about which swap sequences to consider is *deferred* till the total cost for all instructions

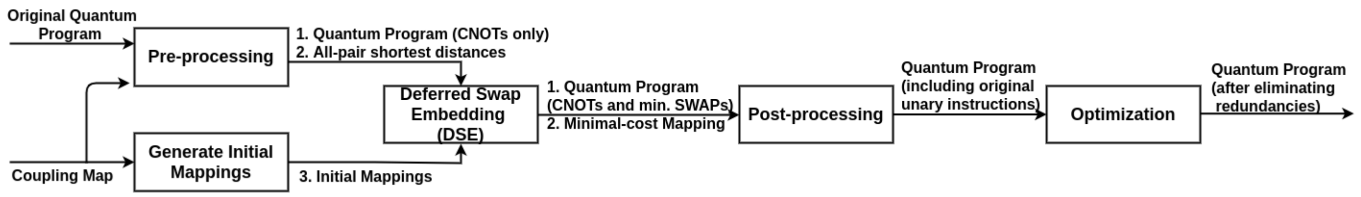
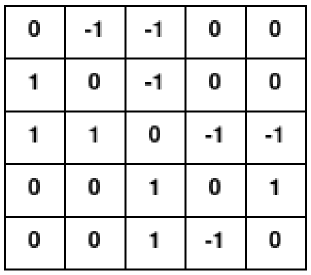

Original Quantum Program
Pre-processing
1. Quantum Program (CNOTs only)
2. All-pair shortest distances
Coupling Map
Generate Initial Mappings
3. Initial Mappings
Deferred Swap Embedding (DSE)
1. Quantum Program (CNOTs and min. SWAPs)
2. Minimal-cost Mapping
Post-processing
Quantum Program (including original unary instructions)
Optimization
Quantum Program (after eliminating redundancies)

Fig. 4.: Overview of the proposed approach showing inputs and outputs of each module.



0	-1	-1	0	0
1	0	-1	0	0
1	1	0	-1	-1
0	0	1	0	1
0	0	1	-1	0

Fig. 5.: Adjacency matrix for the coupling map graph for IBM QX4 architecture

is calculated. Inputs to the DSE algorithm are (1) initial mappings, m_{init}, (2) quantum program instructions $program_{init}$ which include only CNOT instructions after pre-processing is done, and (3) coupling map graph *c_map* represented as an adjacency matrix as seen in Figure 5. The DSE algorithm will find as output an initial mapping which results in the minimal total cost for making all CNOT instructions satisfy CNOT constraints. This is achieved by insertion of additional elementary instructions i.e. the total minimal additional number of elementary gates required while satisfying the CNOT constraints for each instruction.

The DSE algorithm comprises three important steps which are iteratively executed for each initial mapping and for each instruction of the program. These include 1) finding all shortest paths from source qubit to target qubit, 2) finding all swap sequences for each shortest path, and 3) generating minimum cost mappings from these swap sequences.

C.1 Finding all shortest paths from source to target qubits

For a particular mapping all shortest paths between the physical qubits p_s and p_t are found for the current CNOT instruction in $program_{init}$. Note that p_s is the physical qubit on which the logical source qubit l_s is mapped i.e. $m_{init}[p_s] = l_s$. Similarly, p_t is the physical qubit on which the logical target qubit l_t is mapped. We have already pre-calculated the shortest distances in the pre-processing module using the all-pair-shortest-path algorithm, Floyd-Warshall [12]. Now we find all shortest paths between the source and target qubits of the current instruction. Note that the shortest paths are only found if shortest distance $d >= 2$. The value of $d = 1$ implies that the source and target qubits are adjacent in the coupling map and hence no swaps are required. In general, the number of swap instructions required to make the current instruction compatible will be $d - 1$.

C.2 Generating all swap sequences from each shortest path

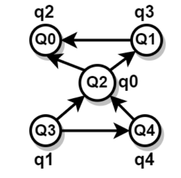
For all shortest paths having $d >= 2$ we generate all possible ways that can make the instruction compatible. For instance, consider a trivial case of a shortest path $p_s \rightarrow p_1 \rightarrow p_t$ where $d = 2$. In this case there are 2 ways of making p_s and p_t adjacent: 1) swap p_s and p_1 to get the sequence $p_1 \rightarrow p_s \rightarrow p_t$, and 2) swap p_t and p_1 to get the sequence $p_s \rightarrow p_t \rightarrow p_1$. We call these *swap sequences* and each swap sequence is a result of $d - 1$ swap instructions that need to be inserted before the current incompatible CNOT instruction. Therefore, for a particular input mapping of an incompatible instruction we may have multiple swap sequences which result in changes to the input mapping in multiple ways.

C.3 Generating minimum cost mappings from swap sequences

During the execution of the algorithm there can be multiple mappings generated as a result of each swap sequence. Hence the generated mapping which has the minimal cost thus far for the particular current instruction is selected as a candidate mapping for the next instruction.

Example: Consider the instance of inputs shown in Figure 6. The given initial mapping is one of the 5! permutation mappings. DSE is executed to identify the minimal mapping and the respective cost of adding extra required swap instructions. The input program has 4 CNOT instructions since the pre-processing step may have removed any unary instructions from the original program. The coupling map, *c_map*, is illustrated as an adjacency matrix having 1 in row i and column j if there is a directed connection from physical qubits i and j in the IBM QX4 architecture. Similarly, 0 refers to absence of a direct physical connection whereas -1 denotes a directed connection from j to i which requires the inclusion of four additional Hadamard instructions to make the

CNOT compatible as shown in Figure 2 d). For instruction 1, DSE evaluates this instruction as being compatible according to the current mapping as the corresponding physical qubits have a 1 in *c_map* resulting in cost = 0. The next instruction has the corresponding physical qubits connected but with a reverse direction resulting in a cost of 4 elementary gates. Therefore, thus far the cost is 4. The logical qubits of instruction 3 are mapped onto non-adjacent physical qubits for which DSE finds all possible swap sequences for the shortest path between Q3 and Q1. The 2 swap sequences and their resulting mappings are depicted in Figure 7. Since there was 1 swap required, the cost increases by 7 as discussed in Figure 3 c). Since there were multiple swap sequences, now DSE will *defer* its decision and check which of the resulting mappings results in the minimal cost for the next instruction. Mapping *m1* results in additional cost of 7 as it requires 1 swap to bring the logical qubits 0 and 2 adjacent, whereas *m2* results in a 0 additional cost. Therefore, *m2* is selected as the next mapping. This ensures a total minimal cost of 11 for this example instance.



Initial Mapping
m_init = [2,3,0,1,4]

```
cx q[1], q[4]
cx q[2], q[3]
cx q[1], q[3]
cx q[0], q[2]
```

Input Program (CNOT only) program_init

	Q0	Q1	Q2	Q3	Q4
Q0	0	-1	-1	0	0
Q1	1	0	-1	0	0
Q2	1	1	0	-1	-1
Q3	0	0	1	0	1
Q4	0	0	1	-1	0

Coupling Map for IBM QX4 c_map

Fig. 6.: Sample inputs for execution of DSE algorithm on the 5-qubit IBM QX4 architecture.

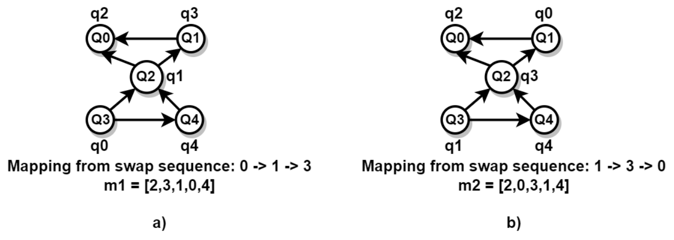

q2
q3
Q0
Q1
Q2
q1
Q3
Q4
q0
q4
Mapping from swap sequence: 0 -> 1 -> 3
m1 = [2,3,1,0,4]
a)
q2
q0
Q0
Q1
Q2
q3
Q3
Q4
q1
q4
Mapping from swap sequence: 1 -> 3 -> 0
m2 = [2,0,3,1,4]
b)

Fig. 7.: Candidate input mappings based on possible swap sequences for execution of instruction 3 of sample input program shown in Figure 6.

D. Post-processing

Before the program is optimized, it must include all unary instructions which were removed as part of the pre-processing step. Therefore, each unary instruction that was removed is inserted in its respective position. If additional swap instructions are included due to constraint satisfaction for a CNOT instruction, the unary gate that preceded that instruction in the original program is inserted before all such swap instructions. Furthermore, the optimization is performed using elementary instructions which only requires each of these swap instructions to be replaced by 7 elementary instructions as shown in Figure 3. This is also done in the post-processing step.

E. Optimization

The optimization module is performed as described by [3] where we perform optimizations on unary gates and binary gates separately.

V. Experimental Evaluation

This section provides an evaluation of the effectiveness of DSE using a set of benchmarks for the IBM QX4 architecture. We compare our proposed method against the baseline solutions by Kole et al. [3] and Wille et al. [4].

A. Hardware Architecture

We have analyzed the benchmark results on the 5-qubit IBM QX4 architecture with directed connections for CNOT gates from the source physical qubit to the target physical qubit as shown in Figure 1.

B. Programming and Execution Environment

The implementation of the proposed method has been done using C++. The experiments were run on a system with Intel Core i7 2.1 GHz CPU, 16 GB main memory, and Ubuntu 18.04 operating system.

C. Benchmarks

We evaluated all benchmarks in [3] and [4] with up to 5 logical qubits.

D. Results

The comparison of results between our proposed method, the DSE algorithm, and the other baseline approaches [4] and [3] is shown in Tables I and II, respectively. The first three columns are the benchmark name, number of logical qubits, and the total quantum gates in the original program. In order to make sure each gate satisfies the CNOT constraints additional swap gates are inserted. Our proposed approach clearly performs better than both baseline results as shown in the tables.

VI. Conclusion

This paper addresses the qubit mapping problem: in order to satisfy the architectural constraints for a given quantum program, for each instruction the aim is to introduce minimal additional swap instructions. We introduce

Table I: Comparison of total gates of Wille et al. [4] vs. proposed total gates

Benchmark	n	g	g'	g"	Δg	Improvement %
3_17_13	3	36	59	44	8	25.4
4gt11_82	5	27	62	51	24	17.7
4gt11_83	5	23	49	43	20	12.2
4gt11_84	4	18	34	26	8	23.5
4gt13_92	5	66	109	82	16	24.8
4mod5-v0_19	5	35	64	56	21	12.5
4mod5-v0_20	5	20	35	32	12	8.6
4mod5-v1_22	5	21	40	35	14	12.5
4mod5-v1_24	5	36	63	48	12	23.8
alu-v0_27	5	36	63	51	15	19.0
alu-v1_28	5	37	64	54	17	15.6
alu-v1_29	5	37	64	51	14	20.3
alu-v2_33	5	37	64	52	15	18.8
alu-v3_34	5	52	90	69	17	23.3
alu-v3_35	5	37	64	52	15	18.8
alu-v4_37	5	37	64	52	15	18.8
ex-1_166	3	19	31	23	4	25.8
ham3_102	3	20	36	26	6	27.8
miller_11	3	50	82	59	9	28.0
mod5d1_63	5	22	48	40	18	16.7
mod5mils_65	5	35	64	55	20	14.1
rd32-v0_66	4	34	63	54	20	14.3
rd32-v1_68	4	36	65	54	18	16.9

n: number of logical qubits; **g**: total gates in original program; **g'**: total gates by [4]; **g"**: total gates by proposed method; **Δg**: g" - g; **Improvement** %: 100 * (g'-g")/g';

Table II: Comparison of total gates of Kole et al. [3] vs. proposed total gates

Benchmark	n	g	g'	g"	Improvement %
01	5	51	64	59	7.8
17	4	43	79	78	1.3
07	5	60	88	87	1.1
4mod5-v0_18	5	69	138	126	8.7

n: number of logical qubits; **g**: total gates in original program; **g'**: total gates by [4]; **g"**: total gates by proposed method; **Improvement** %: 100 * (g'-g")/g';

the Deferred Swap Embedding (DSE) algorithm which calculates all possible swap sequences and defers the decision to select the best sequence until the next instruction. We evaluated our approach on the IBM QX4 architecture and demonstrate that DSE outperforms the two baseline approaches. Future work includes exploring the effectiveness of DSE on larger architectures of IBM systems as well as other architectures such as Google and Rigetti.

References

[1] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge: Cambridge University Press, 2010.

[2] "Eight trends accelerating the age of commercial-ready quantum computing — TechCrunch." https://techcrunch.com/2020/08/06/eight-trends-accelerating-the-age-of-commercial-ready-quantum-computing/ (accessed May 27, 2021).

[3] A. Kole, S. Hillmich, K. Datta, R. Wille, and I. Sengupta, "Improved Mapping of Quantum Circuits to IBM QX Architectures," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 39, no. 10, pp. 2375–2383, Oct. 2020.

[4] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations," in Proceedings of the 56th Annual Design Automation Conference 2019. Las Vegas, NV, USA: Association for Computing Machinery.

[5] A. Zulehner, A. Paler, and R. Wille, "An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 7, pp. 1226–1236, Jul. 2019, doi: 10.1109/TCAD.2018.2846658.

[6] G. W. Dueck, A. Pathak, M. M. Rahman, A. Shukla, and A. Banerjee, "Optimization of Circuits for IBM's five-qubit Quantum Computers," 2018 21st Euromicro Conference on Digital System Design (DSD), pp. 680–684, Aug. 2018.

[7] G. Li, Y. Ding, and Y. Xie, "Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA,April 13-17, 2019.

[8] "IBM Quantum." https://quantum-computing.ibm.com/ (accessed May 27, 2021).

[9] “Google AI Blog: A Preview of Bristlecone, Google’s New Quantum Processor.” https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html (accessed May 27, 2021).

[10] “Rigetti Computing — Rigetti.” https://www.rigetti.com/ (accessed May 27, 2021).

[11] 5-qubit backend: IBM Q team, ”IBM Q 5 Tenerife backend specification V1.4.0,” (2019). Retrieved from https://quantum-computing.ibm.com

[12] R. W. Floyd, “Algorithm 97: Shortest path,” Commun. ACM, vol. 5, no. 6, p. 345, Jun. 1962, doi: 10.1145/367766.368168.

[13] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, “A new universal and fault-tolerant quantum basis,” Information Processing Letters, vol. 75, no. 3, pp. 101–107, Aug. 2000, doi: 10.1016/S0020-0190(00)00084-3.

[14] K. D. Boese and A. B. Kahng, ”Zero-skew clock routing trees with minimum wirelength,” Proceedings of the Fifth Annual IEEE International ASIC Conference and Exhibit, 1992, pp. 17-21, doi: 10.1109/ASIC.1992.270316.