applied sciences

IMPACT FACTOR 2.5

CITESCORE 5.3

Article

Qutrit Control for Bucket Brigade RAM Using Transmon Systems

Lazaros Spyridopoulos, Dimitris Ntalaperas and Nikos Konofaos

Special Issue

Novel Approaches for Quantum Computing and Quantum Information Processing

Edited by

Prof. Dr. Nikos Konofaos

MDPI

https://doi.org/10.3390/app15073950

applied sciences

MDPI

Article

Qutrit Control for Bucket Brigade RAM Using Transmon Systems

Lazaros Spyridopoulos, Dimitris Ntalaperas * and Nikos Konofaos

Department of Informatics, Aristotle University of Thessaloniki, Biology Building, Main University Campus, 54124 Thessaloniki, Greece; lspyrido@csd.auth.gr (L.S.); nkonofao@csd.auth.gr (N.K.)
* Correspondence: ntalaperas@csd.auth.gr

Abstract: Qudits allow the encoding and manipulation of additional quantum information compared to that stored to a two-level qubit system. Although manipulations of qudit states are generally more complex and can introduce extra sources of noise, qudits can still be used in a number of applications when this error can be kept sufficiently low. One such application is the case of the Bucket Brigade Algorithm for realizing a Quantum RAM (QRAM), which inherently uses qutrits for encoding the state of address switches. In this paper, we study a methodology for qutrit manipulation that leverages efficient encoding techniques and pulse calibration methods for the case of transmon systems. The methodology employs an encoding scheme that allows the execution of controlled operations, using the subspace spanned by the two lowest levels of the transmon; we show how this scheme can be used for generating one- and two-qutrit gates by leveraging the Qiskit and Boulder Opal frameworks to compute the parameters of pulses that implement the quantum gates that are used by the BBA. For this type of gate, simulations show that the pulses perform the required operations with a low infidelity when errors introduced by the qutrit Hamiltonian dynamics are considered.

Keywords: quantum control; qutrit; transmon qubits; QRAM

Academic Editor: Nuno Silva

Received: 10 February 2025
Revised: 17 March 2025
Accepted: 26 March 2025
Published: 3 April 2025

Citation: Spyridopoulos, L.; Ntalaperas, D.; Konofaos, N. Qutrit Control for Bucket Brigade RAM Using Transmon Systems. *Appl. Sci.* **2025**, *15*, 3950. https://doi.org/10.3390/app15073950

1. Introduction

Although qubits logically represent two-level quantum systems, physical implementations are typically realized by systems that possess extra physical states. Good candidate architectures for implementing qubits are those that can efficiently isolate a two-level subsystem so that information does not leak to other states; fluxonium [1] and transmon [2] qubits are typical examples of such architectures. However, these extra states can be manipulated in many cases either to facilitate the computation in the two-level scheme or to expand the computational schema to include additional logical states. Tsukanov [3], for example, has demonstrated efficient pulses for single-qubit operations in a double donor structure; the pulses make use of the excited delocalized orbitals for achieving state transfer between the localized states used for encoding the logical qubit.

Going beyond the two-level logical qubit, the concept of qudit is used to denote any quantum information system that contains an n-level logical state. Examples of these systems are qutrit (three-level system), ququart (four-level system), and so on. Similarly to qubit operations, it can be demonstrated that universal gate sets for qudits can also be derived. As shown by Brylinski and Brylinski [4], any two-qudit gate V, combined with the collection of all 1-qudit gates A, constitutes a universal gate set, provided that V is *imprimitive*. An imprimitive 2-qudit gate is one that when acting on a decomposable

state, its output is a state that, in general, cannot be decomposed. For example, if two qudits with states $|x\rangle_0$ and $|x\rangle_1$ are acted upon by the operator V, then the resulting state $V(|x\rangle_0|x\rangle_1)$ cannot be decomposed into a product of the form $|y\rangle_0|y\rangle_1$, with $|y\rangle_i$ being single-qudit states. This situation is similar and can, in fact, be considered a generalization to universality requirements for qubit gates, where at least one entangling gate is required for a gate set to be universal. Wang et al. [5] also studied and demonstrated the existence of universal gate sets for qudits, while also reviewing the extent to which alternative quantum computational models based on qubits, such as measurement-based computing, adiabatic quantum computing, and topological quantum computing, can also be adopted for qudit-based computation. One particularly interesting use case is the possibility of using hybrid qudit computation, in which qudits of different dimensions are used; such hybrid approaches may offer more efficient algorithms when the problem to be solved has some natural mappings that are better served by nonbinary mappings (e.g., simulation of massive spin 1 systems). An algorithm for factorization using a hybrid approach has already been presented in [6], where one spin 3/2 nuclei coupled with a spin 1 nuclei was used to encode qudits of dimensions four and three, respectively.

One particular domain of the application of qudits that is also relevant to the scope of the present work is its usage to compress quantum information in a manner that minimizes the amount of operations and/or qubits needed to perform calculations. In this direction, Baker et al. [7] have shown that by using qudits, it is possible to completely remove ancilla qubits when their number is $O(n)$ and produce an equivalent quantum circuit based on qudits with the same order of depth. On the other hand, Gokhale et al. [8] constructed a logarithmic depth decomposition scheme of the Toffoli gate based on qutrits, which uses no ancilla qubits or qutrits. Two of the authors of the present work have moreover shown that by using higher-energy states, quantum entanglement can be performed at the logical level by only using the spectrum of a single transmon [9].

Since each quantum hardware can support natively different types of pulses and two-qudit entanglement, the choice of single-qudit and two-qudit gates will naturally be different for each architecture. For the regime of the transmon-based architecture, for example, Fisher et al. [10] have shown that qudit control can be efficiently implemented and have obtained excellent experimental results for the case of a cross-resonance gate in a two-ququart system. On the other hand, Chi et al. [11] demonstrated an implementation scheme of a qudit processor that uses silicon-photonic integrated circuits.

One of the main challenges of quantum computation is the design of efficient quantum circuits and finding optimal transpilations for decomposing the logical quantum circuits into a set of instructions that are supported by the native quantum architecture. The ADAPT-VQE and the qubit-ADAPT-VQE algorithms [12,13], for example, provide techniques for constructing hardware efficient ansätze for the Variational Quantum Eigensolver problem; these techniques take into account the constraints of current NISQ processors, such as the limited sized universal gate set and the also limited qubit couplings. This situation is similar to the qudit-based computation; both the techniques employed and their efficiency depend upon the characteristics of the problem at hand and the underlying architecture.

The present work investigates the process of designing optimal pulses for the transmon-based system and for the case of qutrits used in the implementation of the Bucket Brigade Algorithm (BBA), which is one of the candidate schemas for realizing a Quantum RAM (QRAM). Similarly to classical RAM, a QRAM is composed of a set of cells that contain quantum information (typically in the form of a quantum register). Each cell has an address, and the QRAM allows the selection of cells based on the given addresses. Various possible methodologies have been proposed to implement QRAM [14,15].

Briefly, some of the most promising frameworks for designing a QRAM can be summarized as follows:

- The Bucket Brigade QRAM was the first architecture proposed for quantum memory [16] and is the architecture considered in the present work. The Bucket Brigade QRAM uses qutrits for switches and requires the activation of $O(n)$ switches to access an address.
- The Fan-Out QRAM is a quantum extension of the classical Fan-Out RAM [14]. Although Fan-Out QRAM does not need qutrits to implement switches to address cells, it requires the activation of $O(2^n)$ switches to access and address a cell.
- The Flip-Flop QRAM [17] uses a three-stage process to store data; while it achieves a linear circuit width in terms of the number of addresses, the depth of the circuit is exponential.
- Parametric Quantum Circuit (PQC) QRAM is implemented by using a parametrized quantum circuit (a parametrized quantum circuit is one that contains gates that depend upon a set of parameters θ_i, which are typically computed by using an optimization process). This novel approach uses the principles of Machine Learning to train the parametrized circuit and produce a circuit that is efficient for addressing datasets that have similar statistical characteristics with those used for the training set. The Entangling Quantum Generative Adversarial Network (EQGAN) QRAM [18] is one such approach that has shown promise in efficiently storing quantum information, especially when this is used for tasks on the domain of Quantum Machine Learning (QML).

As mentioned, our work is targeted towards the BBA QRAM model. Without going into much detail, the BBA uses three-level systems to build the tree that addresses the registers. Two states are used to denote the path (left–right) and one state is the "wait" state; the path to the target register is obtained by following nodes that have a state different from the "wait" state. Figure 1 depicts an example of an 8-qubit QRAM and how the switches are activated to route to a specific address (see also [19] for a more detailed analysis and how such trees can be simulated in quantum circuits).

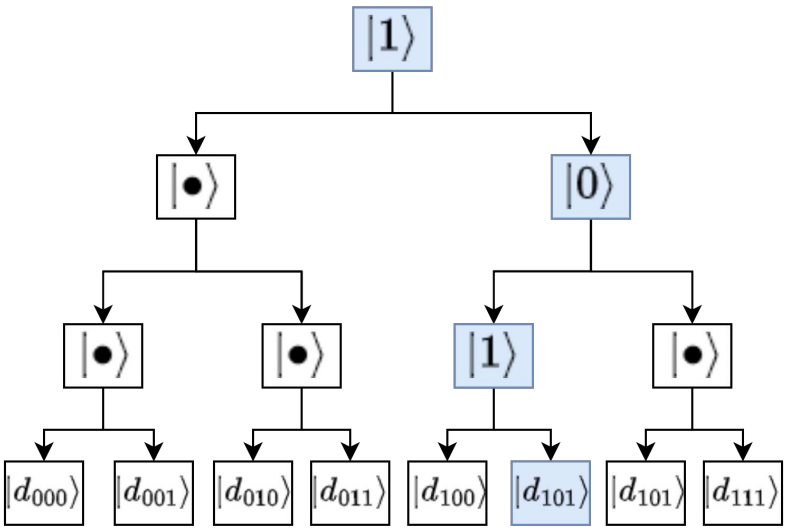


Figure 1. Example of addressing single-qubit data registers using the Bucket Brigade Algorithm. At each level of the tree, a value of 0(1) means a routing to the left (right) sub-tree. • states denote the wait state. d_i represents the data register at address i. In this example, the data of the register with address 101 are requested. Blue color indicates traversed nodes.

In such a scheme, the address register is coupled to the routing qutrits and changes the state from "wait" to the value of the ith qubit, where i denotes the index of the qubit in the index register. For the example of Figure 1, the address register is $|101\rangle$, which sets the states of the corresponding qutrits accordingly. Although a single address is depicted in the example, in general, the QRAM should be able to query a superposition of data registers, according to

$$\sum_{j=0}^{2^n} \alpha_j |j\rangle|0\rangle \rightarrow \sum_{j=0}^{2^n} \alpha_j |j\rangle|d_j\rangle \quad (1)$$

where j denotes the addresses of the data registers we want to obtain, α_j is the amplitude of the superposition of addresses, and d_j are the contents of the data register indexed at j. For the 8-qubit RAM considered in the above example, and assuming 1-qubit data registers for simplicity, if we wanted to obtain an equal superposition of the registers located at positions indexed at 2 and 7, the QRAM would reply with the superposition:

$$\frac{1}{\sqrt{2}}(|2\rangle\rangle d_2 + |7\rangle|d_7\rangle) \quad (2)$$

where Equation (1) is instantiated with $j = 2,7$ and with $\alpha_j = \frac{1}{\sqrt{2}}$ for $j = 2,7$, and $\alpha_j = 0$ otherwise. As the operations that are performed on qutrits in the BBA are of the form of single-qutrit rotations to obtain the desired states and controlled operations to change the state based on the incoming state (i.e., to change a qutrit value from waiting to zero or one), the following architecture can be considered a good candidate architecture for implementing a BBA circuit:

- It provides three-level systems with each state having a sufficiently high relaxation $T1$ and dephasing time $T2$ relative to the execution time of a quantum gate;
- It inherently supports gates for the arbitrary initialization of a qutrit state.
- It allows the execution of single- and two-qutrit gates as the ones required by the BBA algorithm.

Giovanetti et al. presented a possible implementation of BBA using photons propagating along a network of coupled cavities that contain trapped atoms [14]. In the regime of transmon systems, Peterer et al. [20] demonstrated that transmons can be excited with sequential pulses π up to the fourth excited state, with relaxation times that allow the possibility of using these extra energy states for encoding quantum information. The main aim of the work presented in this paper is to demonstrate that the necessary operations needed for implementing a QRAM based on the BBA can be realized in transmon systems and provide a framework for dynamically finding pulse parameters that realize the necessary transitions.

2. Materials and Methods

The present study leverages two methods for designing the desired pulses:

- A simple qutrit encoding schema that is used to map qutrit states to transmon energy levels. The purpose of this schema is to reduce the number of controlled operations that involve higher-energy states of the transmon.
- A simulation framework that leverages the Qiskit Dynamics [21] and the Boulder Opal [22] libraries to perform an ML-assisted pulse optimization problem, that targets the required logical transitions.

The two methods are presented in the following subsections.

2.1. Qutrit Encoding

Transitions of single-qutrit states can be described by 3×3 unitary operators. In the case of qutrits used for BBA, the most common operations are state preparations and state-based transitions. We identify two kinds of operators:

- The translation operators $T_{i,j}$ that perform a flip between the $|i\rangle$ and $|j\rangle$ basis states of the qutrit; for example, the operator $T_{0,2}$ alters the state $|0\rangle(|2\rangle)$ to $|2\rangle(|0\rangle)$, while leaving the state $|1\rangle$ invariant.
- The permutation operator P, which permutes the states according to $|i\rangle \rightarrow |(i+2) \mod 3\rangle$.

In matrix form, the operators can be given by

$$T_{0,1} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, T_{0,2} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, T_{1,2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3}$$

for the translation operators, and

$$P_3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{4}$$

for the permutation operator. Similarly to qubits, controlled operations involve the change of the state of a target qutrit based on the condition of the control qutrit. In direct extrapolation, we can identify the control operation $CT_{i,j}$ that performs the operation $T_{i,j}$ on the target qutrit if the control qutrit is in state $|2\rangle$. The implementation of pulses that perform the $T_{i,j}$ transitions can be performed by sending signals that are resonant with the $i \leftrightarrow j$ transition frequency. The permutation operators can be more difficult to implement directly; however, they can be deconstructed by a set of $T_{i,j}$ pulses by using the identity

$$P_3 = T_{0,1}T_{0,2} \tag{5}$$

Equation (5) can be validated by direct matrix multiplication; however, it can be instructive to see how it works to cycle qutrit states using qutrit translations. Reading from right to left (to simulate how, as a quantum operator, it would act on a qutrit state) and taking as an example the state $|2\rangle$, the state is first acted on by $T_{0,2}$. Since we consider the state with index equal to 2 (we use a zero-indexed scheme, so the column indexed at position 2 is actually the third column), the relevant column of the $T_{0,2}$ operator is $T^2_{0,2} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T$ which results in the transition to state $|0\rangle$. We then take operator $T_{0,1}$ and specifically its zero indexed column $T^0_{0,1} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T$ (since the state is now $|0\rangle$) and observe that it changes the state to $|1\rangle$. This is exactly the same as the operation performed by P_3 alone as can be seen by taking Equation (4) and isolating the second-indexed column. For a proof using explicit matrix multiplication, the reader can refer to Appendix A.

Considering the $CT_{i,j}$ gates, controlling an operation conditional on the control qutrit being in the first hyper-excited state may involve the design of very complex pulses. The operation itself may be prone to additional error since, generally, coupling operations involve pulses that have a longer duration, and the first hyper-excited state has a lower relaxation time than the two lower ones. Various approaches to surpass this problem can be followed, such as redefining the control gates to involve other control states. For our

purposes, however, we follow the simple approach of re-mapping the physical and logical states of the system. For a single transmon, we adopt the following mapping:

$$\begin{aligned} |0\rangle_T &\rightarrow |0\rangle_L \\ |1\rangle_T &\rightarrow |2\rangle_L \\ |2\rangle_T &\rightarrow |1\rangle_L \end{aligned} \tag{6}$$

where $|\rangle_T$ denotes the physical state of the transmon and $|\rangle_L$ the logical ones. Consider now a pulse sequence that performs the usual two-qubit CNOT pulse. Such pulses are implemented in modern hardware in such a manner as to ensure that there is minimum leakage to higher excited states; indeed, the fact that the ground and excited states can be separated by the higher ones is one of the key factors that constitute a good quantum processor architecture, such as the one based on transmon systems.

Ignoring the specifics of the pulse and assuming that it operates in the space spanned by the ground and first excited states, the transitions in the CNOT case for the transmon and logical states can be summarized in Table 1.

Table 1. Correspondence between the physical and logical state evolution under a two-qubit CNOT.

Initial Physical State	Final Physical State	Initial Logical State	Final Logical State
$\|00\rangle_T$	$\|00\rangle_T$	$\|00\rangle_L$	$\|00\rangle_L$
$\|01\rangle_T$	$\|01\rangle_T$	$\|02\rangle_L$	$\|02\rangle_L$
$\|02\rangle_T$	$\|02\rangle_T$	$\|01\rangle_L$	$\|01\rangle_L$
$\|10\rangle_T$	$\|11\rangle_T$	$\|20\rangle_L$	$\|22\rangle_L$
$\|11\rangle_T$	$\|10\rangle_T$	$\|22\rangle_L$	$\|20\rangle_L$
$\|12\rangle_T$	$\|12\rangle_T$	$\|21\rangle_L$	$\|21\rangle_L$
$\|20\rangle_T$	$\|20\rangle_T$	$\|10\rangle_L$	$\|10\rangle_L$
$\|21\rangle_T$	$\|21\rangle_T$	$\|12\rangle_L$	$\|12\rangle_L$
$\|22\rangle_T$	$\|22\rangle_T$	$\|11\rangle_L$	$\|11\rangle_L$

As can be seen in Table 1, the logical state of the target qubit makes the transition $|0\rangle_L \leftrightarrow |2\rangle_L$ when the logical state of the control qubit is $|2\rangle$. This corresponds to the operation of $CT_{0,2}$, which in explicit matrix notation can be described as

$$CT_{0,2} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{7}$$

The other controlled operations can be obtained by using the identities

$$\begin{aligned} CT_{0,1} &= (I_3 \otimes P_3)^2 CT_{0,2} (I_3 \otimes P_3) \\ CT_{1,2} &= (I_3 \otimes P_3) CT_{0,2} (I_3 \otimes P_3)^2 \end{aligned} \tag{8}$$

where I_3 is the three-dimensional identity matrix and the symbol $\otimes$ denotes tensor products. Similarly to the interpretation given for Equation (5), the identities can be logically deducted by observing that the $(I_3 \otimes P_3)$ operators act on the second qutrit subspace to alter its state

accordingly, before and after the operation of the $CT_{0,2}$ operator. As an example, consider the application of the $CT_{1,2}$ operator to a target qutrit in state $|2\rangle$. If the control qutrit is not in state $|2\rangle$, then the $CT_{0,2}$ operator will have no effect and, according to the second equation of Equation (8), the permutation operator will be applied three times to the state, performing the transitions $|2\rangle \rightarrow |1\rangle \rightarrow |0\rangle \rightarrow |2\rangle$, effectively leaving the state invariant. If, however, the control qutrit is in state $|2\rangle$, then after the second permutation, which sends the state of the target qutrit to $|0\rangle$, the $CT_{0,2}$ operator will alter its state to $|2\rangle$. The final permutation will then send state $|2\rangle$ to state $|1\rangle$, thus performing the equivalent of a $CT_{1,2}$ operation. We shall not exhaust all the other cases; the reader can refer to Appendix A for the computation that derives all the $CT_{i,j}$ matrices using the identities of Equation (8).

Under all of the assumptions above, we can see that the desired logical operations can be obtained by pulses that involve only single-qutrit $T_{i,j}$ transitions and only one two-qutrit gate that can be implemented by applying existing approaches that realize the two-qubit CNOT gate.

2.2. Pulse Optimization

The methodology for designing efficient pulses for state preparations and controlled state transitions relies on leveraging existing techniques for accessing the higher-energy states of transmon systems, while also avoiding performing controlled operations that involve higher excited states both as controls and as targets to reduce the computational error. For estimating pulse parameters, we adopt the following Hamiltonian (see, for example, the overview of Kratz et al. [23] for a complete description of how to describe the static and interacting terms of transmon Hamiltonians):

$$\begin{aligned} H = & \sum_{i=\{0,1\}} v_i a_i^\dagger a_i \\ & + \frac{1}{2} \sum_{i=\{0,1\}} \alpha_i a_i^\dagger a_i (a_i^\dagger a_i - I) \\ & + \sum_{i=\{0,1\}} d_i(t)(a_i + a_i^\dagger) \\ & + J \prod_{i=\{0,1\}} (a_i + a_i^\dagger) \end{aligned} \tag{9}$$

where the first two rows refer to the free Hamiltonians of the two transmons with frequencies v_0 and v_1 and with anharmonicities equal to α_0 and α_1. $a_i^\dagger(a_i)$ are the creation(annihilation) operator for qutrit i. $d_i(t)$ refer to the drive signals that can be applied to the transmons to achieve control; these can be further decomposed to the Rabi strengths and the time signals themselves, but we keep these two terms together for simplicity. Factors of 2π are omitted for simplicity. Finally, the coupling of the systems with coupling strength J is included in the final row. While a perfect transmon could be described by the Pauli operators, we introduce the more generic annihilation and creation operators both to access the higher-energy states and to ensure that unwanted leakage to other states is modeled appropriately. To this end, we adopt a 5-level oscillator to accommodate both cases.

For the one-qutrit pulses, the following assumptions are made:

- Relaxation and dephasing times have been ignored as a first approximation. Our main focus is to establish errors due to the leakage to unwanted states, which are typically introduced due to imperfect calibrations or crosstalk terms. These are modeled in our Hamlitonian.
- A maximum power of 1.0 is imposed on our simulations. Although driving a transmon from its ground state directly to the second excited state is, in theory, possible, it may involve the realization of strong pulses so that current architectures may not possess

(the IBM Quantum Platform for example, has an upper limit of a pulse amplitude equal to one; the value can be overridden but only for simulations).

Under those assumptions, the process for constructing the $T_{i,j}$ pulse is the following (for this section, the indices of the T pulses refer to physical states and not logical ones):

1. Perform a Rabi experiment to estimate the amplitude of the pulse.
2. Drive the qutrit and obtain the fidelity.
3. For the $T_{0,2}$ pulse especially, use the two calibrated $T_{0,1}$ and $T_{1,2}$ pulses in sequence (here, and for all single-qutrit operations described in the present section, we refer to the *physical* operations that alter the $|\rangle_T$ states of Equation (6)).

Using a frequency of 4.86 GHz for the first transmon and an anharmonicity of -320 MHz (these, and all the other system parameters, are taken from the existing Almaden backend from IBM Quantum Platform to make the experiments more realistic), the system is simulated using the Qiskit Dynamics library. A five-level oscillator is used to model the possibility of leakage to unwanted higher-energy states. For simplicity, constant pulses are used. For modulating the pulses to signals, we use the $v_0 + \alpha_0$ frequency as a carrier frequency for the transition $T_{0,2}$. Figure 2 shows the mixer output of the two combined signals that performs a $T_{0,2}$ transition. A Rabi strength of 0.21 is used. (Here, we use the normalized dimensionless units that Qiskit uses. The exact strength and units of the signal that a qubit would 'feel', is dependent on the physical realization of the qubit and of any transformations that the signal converter would do. For a flux qubit, for example, the signal would have to be converted to currents. Qiskit hides these details by exposing a dimensionless amplitude parameters and performing the underlying signal conversions.) The dimensionless coupling parameter is taken to be equal to $J = 2 \times 10^{-2}$.

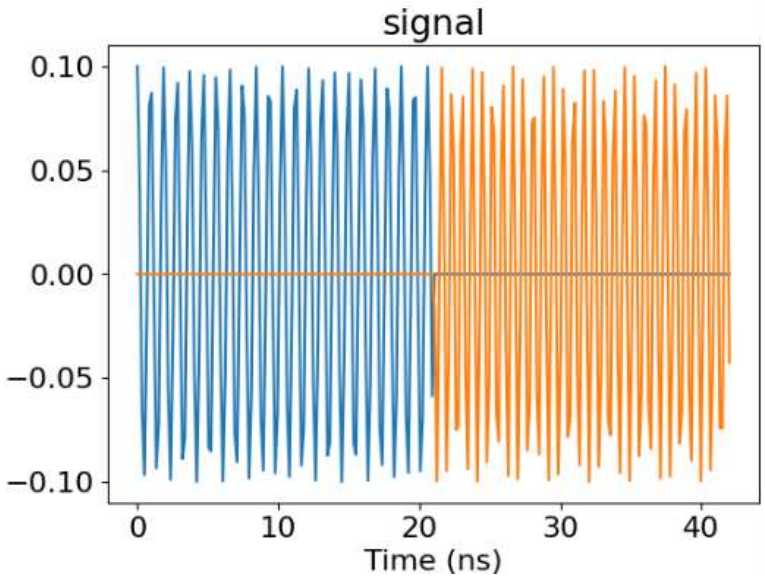


Figure 2. Mixer output for the $T_{0,2}$ signal which consist of a $T_{0,1}$ and $T_{1,2}$ signal in sequence. The first part (blue) is the modulated signal of the first pulse $T_{0,1}$ and the second one (orange) of the second pulse $T_{1,2}$. The two pulses are constant and are modulated with frequencies v and $v + anharm$, respectively.

In Figure 3, some indicative examples of how states are evolved are depicted for the case of the first qubit. As shown, $T_{0,1}$ swaps the populations of $|0\rangle$ and $|1\rangle$ states, while leaving state $|2\rangle$ invariant. The $T_{0,2}$ pulse, on the other hand, performs the $|0\rangle \rightarrow |2\rangle$ transition as expected.

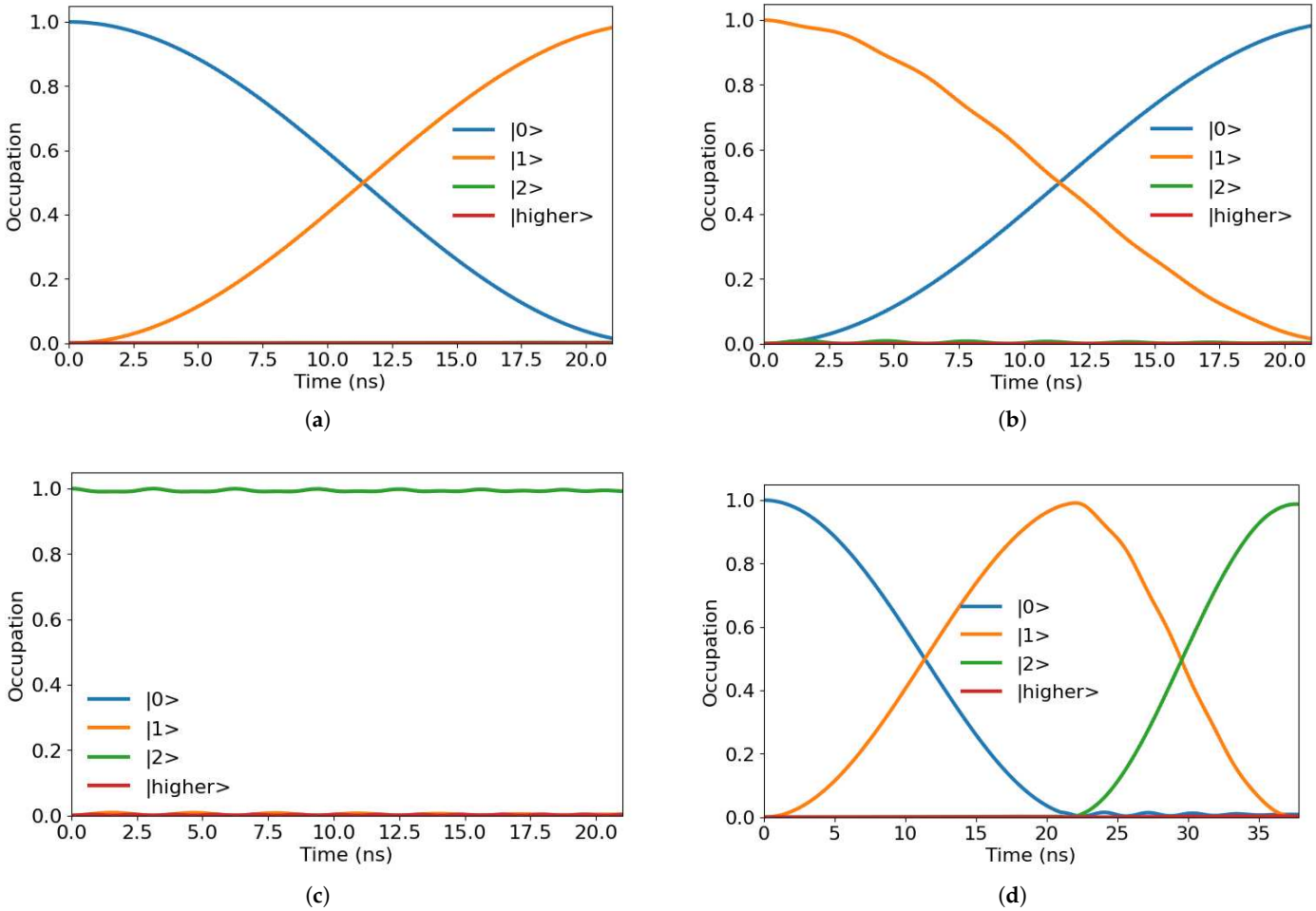


Figure 3. Indicative results for a combination of simulated pulses and starting states. (**a**–**c**) show how the states evolve when a $T_{0,1}$ pulse is performed, with starting states equal to $|0\rangle$ (**a**), $|1\rangle$ (**b**) and $|2\rangle$ (**c**). As expected, populations in states $|0\rangle$ and $|1\rangle$ are exchanged, whereas there is minimum leakage to other states. When the system starts in state $|2\rangle$, all populations remain unaffected to a large degree. (**d**) shows the full evolution of state $|0\rangle$ under a $T_{0,2}$ pulse. The system firstly goes to the excited state $|1\rangle$; a resonant pulse with frequency equal to $v_0 + \alpha_0$ then drives the $|1\rangle \rightarrow |2\rangle$ transition. As the $T_{0,2}$ pulse consists of two sequential pulses, its duration is the sum of the duration of the two constituent pulses.

Regarding the pulses' accuracies, these are estimated by calculating the corresponding infidelities, which are obtained by comparing the relative populations between the target and the computed states. More specifically, the infidelities $inf_{T_{i,j}}$ are computed using the type $inf_{T_{i,j}} = 1 - f_{T_{i,j}}$, where $f_{T_{i,j}}$ are the Hellinger fidelities of the count distributions. Removing the operator subscripts for clarity, for each operator, the Hellinger fidelities are computed by

$$f = (\sum_{i=0}^{5} \sqrt{q_i p_i})^2 \tag{10}$$

where q_i and p_i denote the final populations of the simulated and ideal states, respectively, for all possible outcomes. The possible outcomes are $i = \{0, 1, 2\}$ for the possible qutrit states and $i = \{4, 5\}$ to include the possibility of unwanted leakage to even higher-energy states. (It can be seen that, when comparing to ideal distribution, the sum degenerates to a single term since $p_i = 0$ for all populations not belonging to the target state. However, the Hellinger fidelity thus defined is used since it can be directly computed by using Qiskit's

library and can be easily extended to more general cases.)For a description of the basic setup and simulation parameters of the experiment, the reader can refer to Appendix B.

Table 2 shows the calculated infidelities for the three pulses. Note that the operations are the *physical* ones that correspond to the energy levels of the transmon.

Table 2. Pulse infidelities for the $T_{i,j}$ pulses.

Pulse	Infidelity
$T_{0,1}$	9.81×10^{-7}
$T_{1,2}$	9.86×10^{-7}
$T_{0,2}$	1.90×10^{-6}

It is to be noted that even when the second qutrit is taken into account, noises due to crosstalk can be minimal when the frequencies, anharmonicities, and couplings are appropriately defined. Figure 4 for example, shows the evolution of the first qutrit under a $T_{0,2}$ gate, where the second qutrit involved has a frequency equal to $v_1 = 4.97$ GHz, and an anharmonicity of $\alpha_1 = -320$ MHz, in the presence of crosstalk due to the coupling.

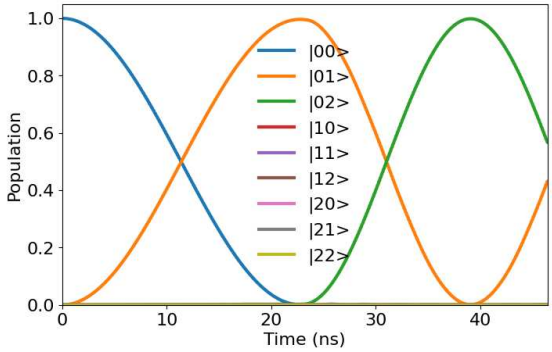


Figure 4. State population of the 2-qutrit system under a $T_{0,2}$ gate on the first qutrit.

2.3. Two-Qutrit Gates

As we have seen in Section 2.1, controlled qutrit operations can be reduced to a series of $T_{i,j}$ and 2-qubit *CNOT* operations with the appropriate encodings. Designing a pulse for performing a CNOT in transmon can then be seen as a trivial problem since we can calibrate existing pulses that modern quantum architectures offer. Figure 5, for example, shows a schedule that performs a CNOT gate on the Valencia processor of the IBM Quantum Platform.

The main issues of obtaining such pulses out of the box, calibrating them, if needed, and applying them for qutrit manipulation according to the schematics presented are the following:

- CNOT gates are rarely truly native gates in transmon architectures. For transmons with fixed frequencies, they are usually based on the cross-resonance gates, which are implemented by driving the control qubit with the target's qubit frequency. Calibrating such a CNOT may involve processes that are not directly applicable to processors that expose different gate-sets (e.g., an ECR gate) and additional calibration processes may need to be applied.
- As is usual with many transmon architectures, arbitrary rotations around the Z axis by and angle θ (realized $Rz(\theta)$ gates) are implemented with virtual pulses (i.e., changes of reference frames) that must be applied across channels in the case of a CNOT gate. When higher-frequency states are involved, these rotations may need to be recalculated to accommodate for the additional wiggle factors of the expanded basis set.

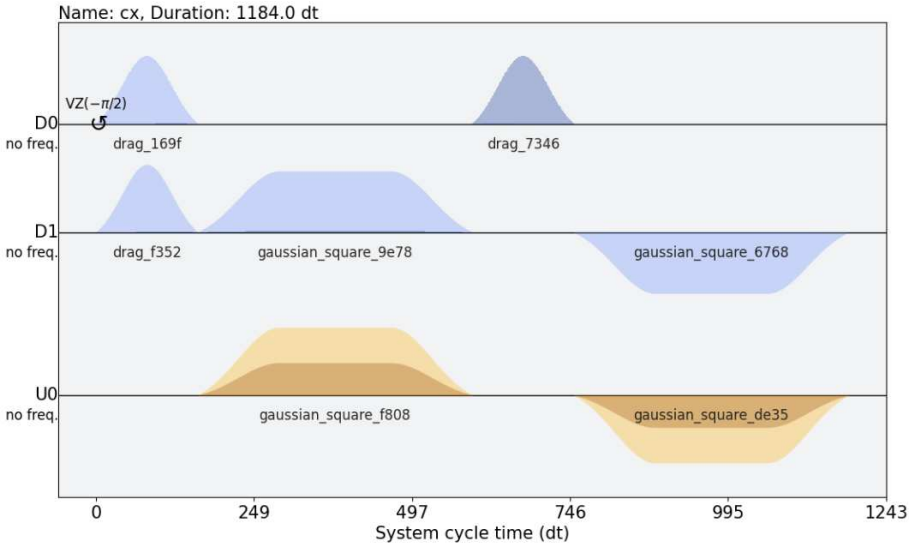


Figure 5. Pulse schedule for the Valencia backend of the IBM Quantum Platform for the CNOT gate. The target qubit is controlled by signals sent in the control's qubit control channel that is driven by the target qubit's frequency. The figure shows the unmodulated Gaussian pulses used for realizing the gate. D_i channels are used to drive qubit i. The U control channel is used to perform controlled operations in the cross-resonance scheme employed by the Valencia processor; pulses in the U channel use the frequency of the target qubit to realize controlled rotations. Virtual Z rotations are depicted as cyclic arrows with the notation $VZ(\phi)$, where ϕ is the angle of the virtual rotation.

To this end, we opt to adopt a more general approach in which a cross-resonance gate can be defined using the characteristics of the underlying Hamiltonian. If such a schedule is found, then a CNOT gate can be obtained by

$$CNOT = (I \otimes Z)^{-1/2}(X \otimes Z)^{-1/2}(X \otimes I)^{-1/2} \tag{11}$$

where $(X \otimes Z)^{-1/2}$ is the maximally entangling cross-resonance gate obtained from the general $R_{xz}(\theta) = \exp(-i\frac{\theta}{2}X \otimes Z)$ gate.

The right term corresponds to a one-qubit $\sqrt{X}$; in the gate set used in our model, this corresponds to a $\sqrt{T_{0,1}}$ gate, which can be performed by the same pulse used for the $T_{0,1}$ if it is applied for half a duration of the pulse.

The leftmost term corresponds to a one-qubit $\sqrt{Z}$ gate. R_z rotations, of which $\sqrt{Z}$ is a special case, can be performed by different methods in transmon systems. If an external pulse $V(t)$ of the form

$$V(t) = \Omega \cos(\omega_d t - \phi) \tag{12}$$

is applied to a transmon qubit, then the total Hamiltonian of the qubit can be written as follows (we use the Pauli matrices for simplicity since we focus for now to two-level systems; in addition, we remove all $\hbar$ factors for convenience):

$$H = -\frac{\omega_0}{2}\sigma_z + V(t)\sigma_x \tag{13}$$

It can be shown that by changing to the frame that rotates with the qubit's frequency and under the rotating wave approximation, the rotating Hamiltonian can be expressed as

$$H_R = \frac{\Omega}{2}\begin{pmatrix} 0 & e^{i\Delta t-\phi} \\ e^{-i\Delta t-\phi} & 0 \end{pmatrix} \tag{14}$$

where Δ, assuming that we are studying the 0-indexed qubit, is equal to $\Delta = \omega_d - \omega_0$. For $\Delta = 0$, we have resonant pulses that perform rotations around the X axis; all single-gate pulses that we have studied thus far fall under this category, with the main difference being that they are performed on systems that have more than two states.

An R_z rotation can be performed via the following:

- Use a slightly off-resonant pulse to introduce a phase difference between the qubit's state vector and the rotating frame.
- Use the ϕ angle in Equation (14) to place the rotation axis parallel to the z axis.

Many platforms, such as the IBM Quantum Platform, use the second approach (please also refer to https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.RZGate (accessed on 6 February 2025)), as this phase change can be performed by simply adjusting the phase of the pulses; such frame changes, also referred to as virtual gates, can be performed classically via the signal generators by introducing the appropriate delays. At the cost of having to track the delays and changes across the channels via the program that implements the pulse schedule of the whole quantum program, this approach implements R_z pulses with effectively zero duration and error (for a complete treatise of how to implement virtual R_z gates and how update the phases to take into account subsequent multi-qubit interactions, see [24]).

In our implementation, arbitrary R_z rotations are seldom needed for qutrit control that is typically encountered in BBA circuits, while, due to the extra computational states used, the extra wiggle factors of these states may increase the complexity of bookkeeping the phase delays. We are therefore using a simple calibration process for identifying optimum pulse parameters for implementing the

$$\sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{15}$$

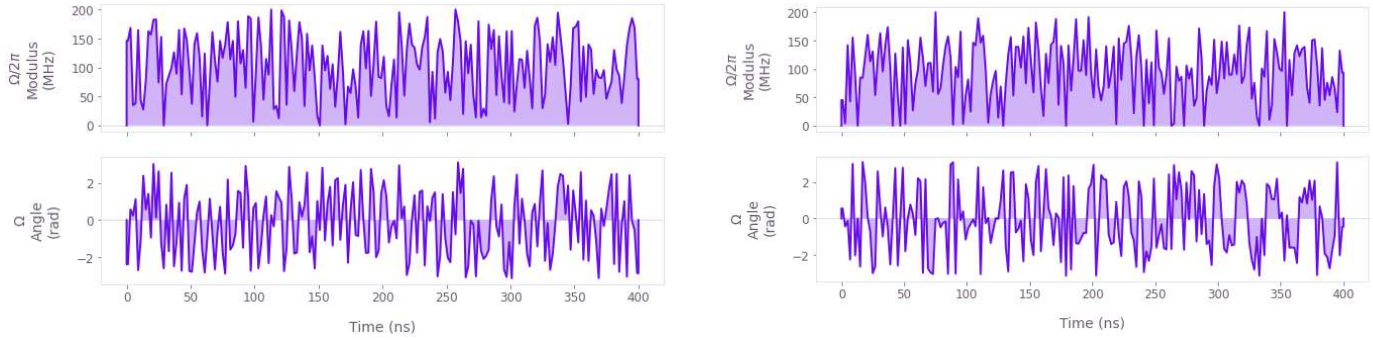
gate by using the Boulder Opal framework. The Boulder Opal framework offers a vast array of functionalities for computing optimal parameters for quantum control and for simulating systems, including, among others, superconducting qubits [25].

In similar fashion to the one-qutrit gates and the simulations based on Qiskit Dynamics, we define a two fixed-frequency transmon system and use the same frequency, anharmonicity, and coupling strengths. We then define the target operation and run the Boulder Opal optimizer to generate a pulse that is being composed by piecewise-constant segments that minimize the gate infidelity. Boulder Opal calculates the operational infidelity of a gate by using the formula:

$$Inf = 1 - \left| \frac{Tr(U_T^\dagger U)}{Tr(U_T^\dagger U_T)} \right|^2 \tag{16}$$

where U_T is the target operator (in our case, these operators are the $(XZ)^{-1/2}$ two-qutrit operator operating in the subspace of the two lowest energy states and $\sqrt{Z}$ operating in the subspace spanned by the two lowest energy states of the first qutrit) and the U is the operator that corresponds to the pulses computed by Boulder Opal using gradient optimization (the reader can refer to Appendix C for a brief description of how Boulder Opal can be used to instantiate problems and compute infidelities). The optimized pulse achieves an infidelity equal to 9.58×10^{-7} for the $\sqrt{Z}$ gate, while the one computed for the $(XZ)^{-1/2}$ following the same methodology achieves an infidelity equal to 1.23×10^{6}. The same methodology can be used for the $(XZ)^{-1/2}$ gate. See Figure 6 for a plot of the computed pulses. For each of the two pulses, the calculated Rabi amplitudes Ω are plotted together with the corresponding phase ϕ (in contrast to the Qiskit experiment, Boulder

Opal computes directly the relevant pulse amplitudes based on the provided Hamiltonian, and as such, these are depicted with explicit units of MHz, which are typical units for quantum systems coupled to microwave pulses).



(**a**) (**b**)

Figure 6. Pulse parameters (frequency and phase) for the $\sqrt{Z}$ (**a**) and $(XZ)^{-1/2}$ (**b**) pulses as computed by using Boulder's Opal optimization.

3. Results

This paper presented a methodology for designing efficient pulses on transmon systems that realize gates that are encountered in the Bucket Brigade Algorithm for QRAM. It focused on transitions between basis states and controlled operations for conditionally altering the state of one qutrit based on the other. In order to avoid the need of designing pulses that control the first hyperexcited state ($|2\rangle$), proper encodings were used, together with additional pulses of one qutrit.

Ignoring the relaxation and dephasing times, and focusing only on errors introduced by the Hamiltonian dynamics, such as crosstalk terms, the infidelities theoretically computed by the simulation are depicted in Table 3, where the indices of the gates correspond now to the logical states introduced in Table 1. Each gate is described by its relevant pulse. For economy, composite gates that are performed by a series of pulses are listed by their gate names in subsequent pulse schedules after they are first introduced (for example, the $T_{0,1}$ gate is described as a series of $T_{1,2}T_{0,2}$ pulses in the third row, but is subsequently listed as $T_{0,1}$ in pulse schedules). Furthermore, the gates $\sqrt{X}$, $\sqrt{Z}$ and the maximally entangling cross-resonance gate that act on the qubit subspace are listed as gates though, strictly speaking, they are not part of the model's gate set. This happens to list their infidelity separately and for better visualizing the pulse schedules.

Table 3. Correspondence between physical and logical state evolution under a two-qubit CNOT.

Gate	Pulse Realization	Infidelity
$T_{0,2}$	$T_{0,2}$	9.81×10^{-7}
$T_{1,2}$	$T_{1,2}$	9.87×10^{-7}
$T_{0,1}$	$T_{1,2}T_{0,2}$	1.90×10^{-6}
P_3	$T_{0,1}T_{0,2}$	3.70×10^{-6}
$Z^{-1/2}$	$Z_{0,1}^{-1/2}$	9.58×10^{-7}
$X^{-1/2}$	$T_{0,2}^{-1/2}$	9.81×10^{-7}
$XZ^{-1/2}$	$XZ^{-1/2}$	1.30×10^{-6}
$CT_{0,2}$	$(IZ)^{-1/2}(XZ)^{-1/2}(XI)^{-1/2}$	2.81×10^{-6}
$CT_{0,1}$	$(I_3P_3)CT_{0,2}(I_3P_3)^2$	3.23×10^{-5}
$CT_{1,2}$	$(I_3P_3)^2CT_{0,2}(I_3P_3)$	3.63×10^{-5}

All infidelities, except the last three, were obtained by performing the simulations described in Section 2. The last three were obtained by applying the pulse parameters in Boulder Opal and taking the average of infidelities of an ensemble of input states. The results agree with what was expected on the basis of the constituent pulse infidelities.

4. Discussion

The results suggest that under the appropriate logical mappings, the qutrit control operations required to operate addressing switches in QRAM based on the BBA architecture can be realized by using the same entangling gates as the one used for quantum computation based on qubits. Although this was achieved at the extra cost of some additional single-qutrit gates, we demonstrated that single-qutrit transitions can be easily defined and calibrated for the case of qutrits that are realized using the energy levels of transmon systems. A methodology for using existing tools to optimize the pulses needed was also presented; this pipeline can be parametrized both in terms of the transmon parameters (e.g., resonance frequencies and anharmonicities) and in terms of the definition of the native gates used, which can be set as target operators that the parametrized signals must approximate. The results suggest that using transmon systems to address quantum registers under the Bucket Brigade scheme may be a promising technique for implementing a QRAM based on BBA in transmons. However, there are various considerations and further steps that need to be taken before the model can be adopted experimentally.

4.1. Noise Suppression and Mitigation

While the infidelities computed seem promising for realizing the described qubit control, these were computed by only minimizing errors introduced by the dynamics of the Hamiltonian, such as leakage to unwanted states or crosstalk terms that need to be canceled for the cross-resonance gate to be effective. Couplings to the external environment that introduce relaxation and dephasing were not considered; these, however, are an important source of error in current NISQ architectures. Future work will focus on applying error mitigation techniques and validating the results on real quantum hardware. Among others, we will take advantage of the Fire Opal [26] package that has demonstrated good results in applying error suppression techniques and that provides a good number of integrations with hardware providers such as those of the IBM Quantum Platform.

4.2. Transmon Connectivity

Transpiling a logical quantum circuit into the native instruction set of a quantum processor may drastically increase the depth of the resulting circuit, also called the Instruction Set Architecture or ISA circuit. The efficiency of transpiling greatly depends on the characteristics of the logical quantum circuit, the native instruction set, and the qubit connectivity of the quantum processor; an efficient algorithm that produces small native circuits for a specific class of problems may perform poorly for other classes even in transpiled in the same quantum processor.

In the case of our model, the gates are implemented at the pulse level; as such, translation to the native gate set is straightforward as long as the quantum hardware supports access at the pulse level. Mapping the logical qutrits to the layout of the backend, however (the so-called routing stage of transpilation), is an entirely different matter, as this often involves quantum state transfers which are typically implemented via $SWAP$ gates for qubits. Designing such pulses with sufficiently low infidelity is one of the main goals of our future work. This work will be supplemented with the investigation of routing techniques for qutrit circuits, with the aim of possibly reducing the required amount of qutrit $SWAP$ operations.

Author Contributions: Conceptualization, L.S. and D.N.; methodology, L.S., D.N. and N.K.; software, D.N.; validation, L.S. and N.K.; formal analysis, L.S. and N.K.; investigation, L.S.; resources, D.N.; data curation, L.S. and D.N.; writing—original draft preparation, L.S. and D.N.; writing—review and editing, N.K.; visualization, D.N.; supervision, N.K.; project administration, N.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BBA	Bucket Brigade Algorithm
EQGAN	Entangling Quantum Generative Adversarial Network
ISA	Instruction Set Architecture
NISQ	Noise Intermediate Scale Quantum era
PQC	Parametric Quantum Circuit
Pwc	Piecewise continuous
QML	Quantum Machine Learning
QRAM	Quantum Random Access Memory
VQE	Variational Quantum Eigensolver

Appendix A

In this appendix, we provide explicit proofs for some of the operator identities presented in the paper. Here, we provide an explicit proof of Equation (6). Written in matrix form, the right-hand side of Equation (6) reads

$$\begin{aligned} T_{0,1}T_{0,2} &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \\ &= P_3 \end{aligned} \tag{A1}$$

For Equation (8), we first observe that

$$I_3 \otimes P_3 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{A2}$$

and

$$(I_3 \otimes P_3)^2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{A3}$$

so that

$$\begin{aligned} CT_{0,2} \otimes (I_3 \otimes P_3) &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned} \tag{A4}$$

and

$$\begin{aligned} CT_{0,2} \otimes (I_3 \otimes P_3)^2 &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned} \tag{A5}$$

Multiplying on the left, we obtain

$$
\begin{aligned}
(I_3 \otimes P_3)^2 CT_{0,2} \otimes (I_3 \otimes P_3) &= \begin{pmatrix} 0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \end{pmatrix} \begin{pmatrix} 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&1&0 \end{pmatrix} \\
&= \begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1 \end{pmatrix} \\
&= CT_{0,1}
\end{aligned}
\tag{A6}
$$

and

$$
\begin{aligned}
(I_3 \otimes P_3) CT_{0,2} \otimes (I_3 \otimes P_3)^2 &= \begin{pmatrix} 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0&0 \end{pmatrix} \begin{pmatrix} 0&1&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \end{pmatrix} \\
&= \begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&1&0 \end{pmatrix} \\
&= CT_{1,2}
\end{aligned}
\tag{A7}
$$

Appendix B

In this appendix, we give some sample code to demonstrate how the simulation and population dynamics can be retrieved using Qiskit's dynamics library for the case of single-qutrit operations. Listing A1 shows how to construct a simple 5-level Duffing oscillator. The *static_term* variable describes the Hamiltonian of the system when no interactions

are present, while the *drive_term* corresponds to the Hamiltonian term that describes the interaction with the pulse. For simplicity, the coupling to the second qutrit is omitted; this is obtained by forming the appropriate Kronecker products of the operators and summing two static terms and the coupling term which equals to $2 * np.pi * J * ((a0 + a0\_dag)@(a1 + a1\_dag))$. The coupling will introduce a shift to the energy levels of the transmons; while this will produce different values for the pulse parameters, the calibration process will be the same.

Listing A1. Single-qutrit model.

```
# Transmon parameters
v_0 = 4.86
r_0 = 0.22
dt = 1 / 22.
#dimensionality
dim = 5
#Anharmonicity of the transmon
anharm_0 = -0.32
I = np.eye(dim, dtype=complex)
a = np.diag(np.sqrt(np.arange(1, dim)), 1)
a_dag = np.diag(np.sqrt(np.arange(1, dim)), -1)
N_op = np.diag(np.arange(dim))
static_term = 2 * np.pi * v_0 * N_op + np.pi * anharm_0 * N_op * (N_op - I)
drive_term = 2 * np.pi * r_0 * (a + a_dag)
```

Pulses are built using the Pulse module of Qiskit. Listing A2 showcases the construction of a pulse that drives the qutrit from the ground to the $|2\rangle$ state. This pulse consists of two pulses performed in sequence, one that drives the qutrit to state $|1\rangle$ and a second one that waits for the first one to complete and then drives the qutrit to state $|2\rangle$.

Listing A2. The construction of constant pulses for the $0 \rightarrow 2$ transition.

```
with pulse.build(name="$X_{01}$ schedule") as sX_01:
    pulse.play(pulse.Constant(duration_01, amp_01), pulse.DriveChannel(0))

with pulse.build(name="$X_{12}$ schedule") as sX_12:
    pulse.delay(duration_01, pulse.DriveChannel(0))
    pulse.play(pulse.Constant(duration_02, amp_02), pulse.DriveChannel(0))
```

The two pulses are then sent to the mixer so that they can be modulated using the carrier frequency (Listing A3). Note how the second uses a carrier frequency modified by the anharmonicity of the transmon to construct the second signal.

Listing A3. Pulse to signal conversion.

```
mixer_01 = InstructionToSignals(dt, carriers={"d0": v_0})
mixer_12 = InstructionToSignals(dt, carriers={"d0": v_0+anharm})

signal_01 = mixer_01.get_signals(sX_01)
signal_12 = mixer_12.get_signals(sX_12)

signals = signal_01 + signal_12
```

The solver parameters are depicted in Listing A4. While the channel carrier frequency is the one of the driving channel, the second signal will have a modulation frequency that is offset by the anharmonicty; that will drive the $1 \rightarrow 2$ transition, instead of having the qutrit return to state $|0\rangle$ as would be the case of a typical Rabi experiment with a constant resonant frequency.

Listing A4. Solver parameters.

```
solver = Solver(
    static_hamiltonian=static_term,
    hamiltonian_operators=[drive_term],
    rotating_frame=static_term,
    rwa_cutoff_freq=2 * 5.0,
    hamiltonian_channels=['d0'],
    channel_carrier_freqs={'d0': v_0},
    dt=dt
)
```

Appendix C

Boulder Opal uses a graph structure to represent a computation, with all important parameters and optimizable signals being parts of the graph. To compute the pulse parameters that implemented our target operators, the *run_optimization* method of Boulder Opal was used on the constructed graph. This method performs a gradient-based optimization fit to find the best parameters that minimize the cost function. In our scenario, the parameters were the amplitude and phase of the pulse, which was represented as a piecewise constant (Pwc) function. Pwc functions are functions that are 'glued' from constant segments, with each segment having different values for the optimizable parameters. An optimization process fits these parameters so that the overall function minimizes the cost function. The cost was the operational fidelity defined in Equation (16).

The snippet in Listing A5, taken from the official Boulder Opal documentation, depicts how a graph can be created to describe the problem of computing an infidelity. The target operation is the identity, and this is compared to one signal that performs a *sigma_z* for a duration of 0.1 s followed by a $-$*sigma_z* for another 0.1 s. When no noise is considered, the two operations are the same and, thus, the infidelity is equal to zero.

Listing A5. Example of an infidelity calculation.

```
>>> sigma_z = np.array([[1, 0], [0, -1]])
>>> hamiltonian = graph.pwc(
...     durations=np.array([0.1, 0.1]), values=np.array([sigma_z, -sigma_z])
... )
>>> target = graph.target(np.eye(2))
>>> infidelity = graph.infidelity_pwc(
...     hamiltonian=hamiltonian, target=target, name="infidelity"
... )
>>> result = bo.execute_graph(graph=graph, output_node_names="infidelity")
>>> result["output"]["infidelity"]["value"]
0.0
```

Defining the cost to be minimized using Boulder Opal can be easily performed by using code as the one depicted in Listing A6. In this example, *z_sqr* is the matrix of the ideal gate $\sqrt{Z}$ acting on the subspace of the first qutrit, while the Hamiltonian variable is the parametrized pulse to be optimized.

Listing A6. Example of an infidelity calculation.

```
cost = graph.infidelity_pwc(
    hamiltonian=hamiltonian,
    target=graph.target(operator=z_sqr),
)
```

References

1. Nguyen, L.B.; Koolstra, G.; Kim, Y.; Morvan, A.; Chistolini, T.; Singh, S.; Nesterov, K.N.; Jünger, C.; Chen, L.; Pedramrazi, Z.; et al. Blueprint for a high-performance fluxonium quantum processor. *PRX Quantum* **2022**, *3*, 037001. [CrossRef]
2. Koch, J.; Yu, T.M.; Gambetta, J.; Houck, A.A.; Schuster, D.I.; Majer, J.; Blais, A.; Devoret, M.H.; Girvin, S.M.; Schoelkopf, R.J. Charge-insensitive qubit design derived from the Cooper pair box. *Phys. Rev. A—At. Mol. Opt. Phys.* **2007**, *76*, 042319. [CrossRef]
3. Tsukanov, A.V. Single-qubit operations in the double-donor structure driven by optical and voltage pulses. *Phys. Rev. B—Condens. Matter Mater. Phys.* **2007**, *76*, 035328. [CrossRef]
4. Brylinski, J.L.; Brylinski, R. Universal quantum gates. In *Mathematics of Quantum Computation*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2002; pp. 117–134.
5. Wang, Y.; Hu, Z.; Sanders, B.C.; Kais, S. Qudits and high-dimensional quantum computing. *Front. Phys.* **2020**, *8*, 589504. [CrossRef]
6. Zobov, V.; Ermilov, A. Implementation of a quantum adiabatic algorithm for factorization on two qudits. *J. Exp. Theor. Phys.* **2012**, *114*, 923–932. [CrossRef]
7. Baker, J.; Duckering, C.; Chong, F. Efficient Quantum Circuit Decompositions via Intermediate Qudits. *arXiv* **2020**, arXiv:2002.10592.
8. Gokhale, P.; Baker, J.M.; Duckering, C.; Brown, N.C.; Brown, K.R.; Chong, F.T. Asymptotic improvements to quantum circuits via qutrits. In Proceedings of the 46th International Symposium on Computer Architecture, Phoenix, AZ, USA, 22–26 June 2019; pp. 554–566.
9. Ntalaperas, D.; Konofaos, N. Encoding two-qubit logical states and quantum operations using the energy states of a physical system. *Technologies* **2021**, *10*, 1. [CrossRef]
10. Fischer, L.E.; Chiesa, A.; Tacchino, F.; Egger, D.J.; Carretta, S.; Tavernelli, I. Universal qudit gate synthesis for transmons. *PRX Quantum* **2023**, *4*, 030327. [CrossRef]
11. Chi, Y.; Huang, J.; Zhang, Z.; Mao, J.; Zhou, Z.; Chen, X.; Zhai, C.; Bao, J.; Dai, T.; Yuan, H.; et al. A programmable qudit-based quantum processor. *Nat. Commun.* **2022**, *13*, 1166. [CrossRef]
12. Tang, H.L.; Shkolnikov, V.; Barron, G.S.; Grimsley, H.R.; Mayhall, N.J.; Barnes, E.; Economou, S.E. qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor. *PRX Quantum* **2021**, *2*, 020310. [CrossRef]

13. Grimsley, H.R.; Economou, S.E.; Barnes, E.; Mayhall, N.J. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nat. Commun.* **2019**, *10*, 3007. [PubMed]
14. Giovannetti, V.; Lloyd, S.; Maccone, L. Architectures for a quantum random access memory. *Phys. Rev. A—At. Mol. Opt. Phys.* **2008**, *78*, 052310.
15. Phalak, K.; Chatterjee, A.; Ghosh, S. Quantum random access memory for dummies. *Sensors* **2023**, *23*, 7462. [CrossRef]
16. Giovannetti, V.; Lloyd, S.; Maccone, L. Quantum random access memory. *Phys. Rev. Lett.* **2008**, *100*, 160501.
17. Park, D.K.; Petruccione, F.; Rhee, J.K.K. Circuit-based quantum random access memory for classical data. *Sci. Rep.* **2019**, *9*, 3949.
18. Niu, M.Y.; Zlokapa, A.; Broughton, M.; Boixo, S.; Mohseni, M.; Smelyanskyi, V.; Neven, H. Entangling quantum generative adversarial networks. *Phys. Rev. Lett.* **2022**, *128*, 220505. [CrossRef] [PubMed]
19. Arunachalam, S.; Gheorghiu, V.; Jochym-O'Connor, T.; Mosca, M.; Srinivasan, P.V. On the robustness of bucket brigade quantum RAM. *New J. Phys.* **2015**, *17*, 123010. [CrossRef]
20. Peterer, M.J.; Bader, S.J.; Jin, X.; Yan, F.; Kamal, A.; Gudmundsen, T.J.; Leek, P.J.; Orlando, T.P.; Oliver, W.D.; Gustavsson, S. Coherence and decay of higher energy levels of a superconducting transmon qubit. *Phys. Rev. Lett.* **2015**, *114*, 010501.
21. Puzzuoli, D.; Wood, C.J.; Egger, D.J.; Rosand, B.; Ueda, K. Qiskit Dynamics: A Python package for simulating the time dynamics of quantum systems. *J. Open Source Softw.* **2023**, *8*, 5853. [CrossRef]
22. Q-CTRL. Boulder Opal. 2025. Available online: https://q-ctrl.com/boulder-opal (accessed on 2 February 2025).
23. Krantz, P.; Kjaergaard, M.; Yan, F.; Orlando, T.P.; Gustavsson, S.; Oliver, W.D. A quantum engineer's guide to superconducting qubits. *Appl. Phys. Rev.* **2019**, *6*, 021318. [CrossRef]
24. McKay, D.C.; Wood, C.J.; Sheldon, S.; Chow, J.M.; Gambetta, J.M. Efficient Z gates for quantum computing. *Phys. Rev. A* **2017**, *96*, 022330. [CrossRef]
25. Q-CTRL. Perform Model-Based Robust Optimization for the Cross-Resonance Gate. Available online: https://docs.q-ctrl.com/boulder-opal/apply/superconducting-systems/perform-model-based-robust-optimization-for-the-cross-resonance-gate (accessed on 23 January 2025).
26. Q-CTRL. Fire Opal. 2025. Available online: https://q-ctrl.com/fire-opal (accessed on 2 February 2025).