5-2009

Extended Superposed Quantum State Initialization Using Disjoint Prime Implicants

David Rosenbaum
Portland State University

Marek Perkowski
Portland State University

Citation Details

Rosenbaum, David, and Marek Perkowski. "Extended superposed quantum-state initialization using disjoint prime implicants." Physical Review A 79, no. 5 (2009): 052310.

Extended superposed quantum-state initialization using disjoint prime implicants

David Rosenbaum*
Department of Computer Science, Portland State University, 1900 SW 4th Avenue, Portland, Oregon 97201, USA

Marek Perkowski†
Department of Electrical Engineering, Portland State University, 1900 SW 4th Avenue, Portland, Oregon 97201, USA

(Received 1 January 2009; published 12 May 2009)

Extended superposed quantum-state initialization using disjoint prime implicants is an algorithm for generating quantum arrays for the purpose of initializing a desired quantum superposition. The quantum arrays generated by this algorithm almost always use fewer gates than other algorithms and in the worst case use the same number of gates. These improvements are achieved by allowing certain parts of the quantum superposition that cannot be initialized directly by the algorithm to be initialized using special circuits. This allows more terms in the quantum superposition to be initialized at the same time which decreases the number of gates required by the generated quantum array.

DOI: 10.1103/PhysRevA.79.052310 PACS number(s): 03.67.Lx

I. INTRODUCTION

The problem of initializing a quantum superposition is important for Grover's algorithm [1], quantum neural networks [2,3], and other applications. The purpose of the algorithm presented here is to generate a quantum array that initializes a desired quantum superposition from a basis state. Ventura and Martinez [4] created an algorithm that generates a quantum array capable of initializing a quantum superposition of the form $|\psi\rangle=\Sigma_{i=0}^{2^n-1}\frac{t_i}{\sqrt{m}}|i\rangle$, where $t_i\in\{-1,0,1\}$, m is equal to the number of terms in the desired quantum superposition, and n is the number of qubits in the desired quantum superposition using $\Theta(mn)$ one and two qubit operations and $n+1$ ancilla qubits. Long and Sun [5] created another algorithm based on a different principle that solves a similar problem without using any ancilla bits but requires an exponential number of one and two qubit operations in the generated quantum array. An advantage of the Long-Sun algorithm [5] is that it uses a training operator that is based on sinusoidal functions in contrast to the Ventura-Martinez algorithm which uses a special training operator. The superposed quantum state initialization using disjoint prime implicants (SQUID) algorithm [6] is based on the Ventura-Martinez algorithm [4]. The SQUID algorithm [6] is an improvement over the Ventura-Martinez [4] and Long-Sun [5] algorithms and generates quantum arrays that almost always use fewer gates than those generated by the Ventura-Martinez algorithm [4] and in the worst case generates a quantum array that requires the same number of operations. The extended superposed quantum state initialization using disjoint prime implicants (ESQUID) algorithm presented in this paper improves on the SQUID algorithm [6] by generalizing the phase groups used in the SQUID algorithm [6] with the introduction of generalized phase groups. This is done by using special circuits to initialize parts of the quantum superposition that cannot be initialized directly by the algorithm in an efficient manner. This allows the desired quantum superposition to be represented using fewer groups which results in fewer gates in the quantum arrays generated by the ESQUID algorithm.

II. INITIALIZING THE STARTING STATE

The Ventura-Martinez [4], Long-Sun [5], and SQUID [6] algorithms require all qubits to be initialized to $|0\rangle$ before the generated quantum array is applied. The ESQUID algorithm presented in this paper also requires this starting state to be initialized before the quantum array it generates is applied. This requires another algorithm to be run first to initialize the state to $|0^n,00\rangle$ prior to applying the quantum array generated by the ESQUID algorithm. One method for initializing the starting state is the Schulman-Vazirani heat engine [7]. The rest of this paper will assume that the starting state has been initialized and will focus on initializing the desired superposition from the starting state.

III. DESIRED QUANTUM SUPERPOSITIONS

The ESQUID algorithm is capable of initializing the same class of quantum superpositions as the SQUID algorithm [6]. Thus, the desired quantum superposition must be of the form

$$|\psi\rangle=\sum_{i=0}^{2^n-1}\frac{t_i}{\sqrt{m}}|i\rangle,\quad t_i\in\{-1,0,1\},\tag{1}$$

where m is the number of nonzero amplitudes in $|\psi\rangle$. Note that this is also the number of terms in $|\psi\rangle$.

A. Phase maps

Phase maps [6] are a special type of Karnaugh map that were created for use with the SQUID algorithm. A phase map provides a way to visualize a quantum superposition. The idea is that each cell on the phase map contains the coefficient of the term in the quantum superposition that corresponds to the minterm of the cell. This can be obtained by concatenating the binary representations of the row and column of the cell in the same way as for a conventional Karnaugh map. If all amplitudes are equal as is the case in Eq.

*drosenba@cs.pdx.edu
†mperkows@ece.pdx.edu

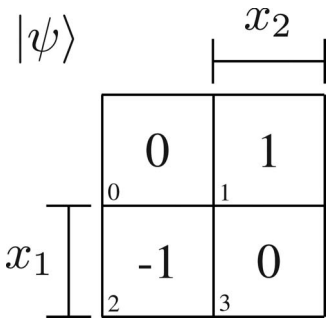

FIG. 1. The phase map for Eq. (2).

(1), the amplitudes can be omitted and each cell can contain only the phase. Thus, in the case of the SQUID and ESQUID algorithms, each cell must contain some $t_i \in \{-1,0,1\}$, where i is the index of the cell on the phase map. As an example, the quantum superposition

$$|\psi\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle \tag{2}$$

can be represented using the phase map shown in Fig. 1.

B. Phase groups

The phase maps used in the SQUID and ESQUID algorithms also use a type of group called a phase group to represent quantum superpositions. The idea is that each phase group contains a set of minterms that correspond to terms in the desired quantum superposition that will be initialized at the same time when the SQUID algorithm is run. A term in a quantum superposition is the product of a basis state that has a nonzero coefficient in the quantum superposition and its coefficient. Certain restrictions apply to which sets of minterms can form a phase group. It must be possible to express a phase group as a string of length n where each element of the string is 0 or 1 if the corresponding qubit is 0 or 1, respectively, in the phase group. If the qubit is not constant within the phase group then $*$ is used as the corresponding element in the string in order to indicate this. Furthermore, the quantum superposition that corresponds to the phase group must be of the form

$$|\alpha\rangle = \pm \bigotimes_{j=0}^{n} |\beta_j\rangle, \tag{3}$$

where

$$|\beta_j\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle), \quad |0\rangle \text{ or } |1\rangle. \tag{4}$$

The symbol $\otimes$ denotes a Kronecker product and the sign in Eq. (4) is determined by the corresponding phase group. This implies that each phase group either has the value -1 in every cell, the value 1 in every cell, or has an equal number of the values -1 and 1 in its cells. Thus, phase groups have the same shapes as literal product groups in a standard Karnaugh map but restrictions apply to the phases that can be present within each phase group. Two phase groups are disjoint if they do not overlap on the phase map. Because the SQUID [6] and ESQUID algorithms assume that all phase groups are disjoint and will not initialize the desired quantum superposition correctly if this is not the case, minterms with amplitudes of 0 may not be present in any phase group.

C. Generalized phase groups

While the SQUID algorithm [6] requires that the desired superposition be expressed as a disjoint set of phase groups, the ESQUID algorithm allows the use of a more general type of phase group called a generalized phase group. A generalized phase group is a set of 2^ℓ minterms on the phase map where there exists a $2^n \times 2^n$ permutative operator U such that

$$|\gamma\rangle = U|\alpha_\gamma\rangle, \tag{5}$$

where $|\gamma\rangle$ is the quantum superposition that corresponds to the generalized phase group and $|\alpha_\gamma\rangle$ is a phase group that contains 2^ℓ minterms. The quantum state $|\alpha_\gamma\rangle$ is called the intermediate phase group. Note that every phase group is also a generalized phase group because U can be taken to be the identity matrix. Because U is a permutative operator, a generalized phase group can have the value -1 in every cell, the value 1 in every cell, or an equal number of the values -1 and 1 in its cells as is the case for phase groups. In the ESQUID algorithm, the desired quantum superposition is represented as a sequence of generalized phase groups. In order for the ESQUID algorithm to initialize the correct quantum superposition, all generalized phase groups in the sequence must be disjoint and the intermediate phase group that each generalized phase group is created from must be disjoint from all previous generalized phase groups in the sequence of generalized phase groups. This is necessary because the generalized phase groups at the beginning of the sequence will be initialized first by the ESQUID algorithm. If an intermediate phase group or a generalized phase group later in the sequence overlapped with a generalized phase group earlier in the sequence, part of the quantum superposition that had already been initialized would be changed when the new intermediate phase group or generalized phase group was initialized. This would result in the ESQUID algorithm failing to initialize the correct quantum superposition.

D. Complexity of the U operations

Because a permutative U operation is used for each generalized phase group initialized by the ESQUID algorithm, the U operations have an important impact on the complexity of the ESQUID algorithm in terms of the number of one and two qubit gates required. Since U is an arbitrary permutative operation, implementing it using inverters controlled by up to $n-1$ qubits requires an exponential number of operations in the most general case. The number of operations is still exponential if the permutative U operations are implemented using controlled single qubit operations selected from any finite set of single qubit unitary matrices. However, it is possible to place restrictions on the U operations so that they use a polynomial number of operations. Let G be a finite set of

single qubit unitary matrices. Let a be the number of elements in the set G. To show that the number of operations required is exponential in general consider the number of possible permutative operators on n qubits compared with the number of possible quantum arrays that can be created using a polynomial number of controlled single qubit operations from the set G. Since each permutative operation on n qubits corresponds to an invertible function over the set of all integers between 0 and 2^n-1 inclusive, there are a total of $2^n!$ permutative operations on n qubits. If every permutative operation on n qubits can be implemented using a polynomial number of controlled single qubit operations from the set G then there exists a polynomial $p(n)$ such that $p(n)$ is an upper bound on the number of operations in every such quantum array. Since there are n qubits and a possible operations, there are $an\Sigma_{j=0}^{n-1}\binom{n-1}{j}+1=an2^{n-1}+1$ choices for each operation in each quantum array since there are $\binom{n-1}{j}$ ways to place each operation with j controls. Note that each of these operations could also be the identity operation since $p(n)$ is an upper bound so $an2^{n-1}+1$ is the number of choices for each operation rather than $an2^{n-1}$. This implies that the number of quantum arrays that can be constructed using up to $p(n)$ controlled inverters is bounded above by $(an2^{n-1}+1)^{p(n)}$. It can be shown that

$$\lim_{n\to\infty}\frac{(an2^{n-1}+1)^{p(n)}}{2^n!}=0. \tag{6}$$

It follows that $(an2^{n-1}+1)^{p(n)}<2^n!$ for large n. Therefore, there exist permutative operations that cannot be implemented using a polynomial number of controlled operations from a finite set of single qubit unitary matrices. It will now be shown how to place restrictions on the U operations in order to allow them to be implemented using a polynomial number of operations. Let the variables x_j, $j=1,\ldots,n$ denote the inputs to the permutative operator U. Let $j_1,\ldots,j_n$ be integers that denote the order in which each output is calculated in the quantum array such that the qubit with index j_k is the kth output calculated. Let the output on the (j_k)th qubit be $y_{j_k}=f_{j_k}(y_{j_1},\ldots,y_{j_{k-1}},x_{j_k},\ldots,x_{j_n})$. The idea here is that the outputs are calculated in the order given by $j_1,\ldots,j_n$. Since each output y_{j_k} replaces x_{j_k} as the value of the (j_k)th qubit, each output y_{j_k} is a function of the inputs $x_{j_\ell},\ell=k,\ldots,n$ to the permutative U operator as well as the outputs $y_{j_\ell},\ell=1,\ldots,k-1$. This is because when the kth output is computed, the first $k-1$ outputs have already been computed so each of the values $x_{j_\ell},\ell=1,\ldots,k-1$ has been replaced by the values $y_{j_\ell},\ell=1,\ldots,k-1$, respectively. Let $q(n)$ be a polynomial. The U operations can now be restricted to a polynomial number of controlled inverters by requiring each output $y_{j_k}=f_{j_k}(y_{j_1},\ldots,y_{j_{k-1}},x_{j_k},\ldots,x_{j_n})$ to be an exclusive sum of products (ESOP) with at most $q(n)$ products of any input or its negation (termed a literal). This means that each function can be constructed using controlled inverters since the exclusive sum corresponds to the exclusive or operation. The total number of controlled inverters required to implement a U operation is therefore bounded above by $nq(n)$ and is therefore polynomial. By using different degrees for the polynomial $q(n)$ more control can be gained over the complexity of the U operations.

IV. ESQUID ALGORITHM

The ESQUID algorithm operates on the state $|x_1,\ldots,x_n,c_1c_2\rangle$ where the desired quantum superposition is initialized on the qubits $|x_i\rangle,i=1,\ldots,n$ and $|c_1\rangle$ and $|c_2\rangle$ are ancilla qubits.

A. Codes in the ancilla qubits

The ancilla qubits $|c_1\rangle$ and $|c_2\rangle$ are called the code qubits and keep track of the following:

(i) which terms in the current superposition have been initialized,

(ii) which terms are currently being initialized, and

(iii) which terms will be used later to create more terms in the quantum superposition (this is called the generator state [4]).

The following codes are used:

(i) $|00\rangle$ on the $|c_1\rangle$ and $|c_2\rangle$ qubits is not used.

(ii) $|01\rangle$ on the $|c_1\rangle$ and $|c_2\rangle$ qubits is used to indicate that the corresponding terms in the quantum superposition have already been initialized to the proper values and should not be modified again by the algorithm.

(iii) $|10\rangle$ indicates that the corresponding terms in the quantum superposition are part of the generalized phase group that is currently being initialized.

(iv) $|11\rangle$ is used to indicate the generator state. Note that applying a SWAP gate to the code that indicates the current group transforms it into the code for an initialized term in the quantum superposition. Also, applying a SWAP gate to the code for the generator state will not change it. The generated quantum array will take advantage of both of these properties by using controlled SWAP gates to update the codes for the terms in each generalized phase group after it is initialized.

B. Initialization operators

The training operator used in the ESQUID algorithm is the same as the training operator used in the SQUID algorithm [6] which is based on the training operator from the Ventura-Martinez algorithm [4]. However, it relies on a different concept than the operator used in the Ventura-Martinez algorithm as it operates on phase map groups rather than on the individual minterms that the original operator in the Ventura-Martinez algorithm operates on. This allows many minterms to be initialized at the same time by creating a new term in the quantum superposition and then splitting it using controlled Hadamard gates. The operator is defined by Eq. (7),

$$S_{t,g,p}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&\sqrt{\frac{p-g}{p}}&t\sqrt{\frac{g}{p}}\\0&0&-t\sqrt{\frac{g}{p}}&\sqrt{\frac{p-g}{p}}\end{bmatrix}. \tag{7}$$

This operator works by splitting the terms corresponding to the groups off from the generator state during the algorithm

TABLE I. Algorithm I: High level pseudocode for the ESQUID algorithm.

(1)	Find a small set A of generalized phase groups using logic synthesis methods
(2)	Initialize all qubits $\|x_i\rangle$, $\|c_1\rangle$, and $\|c_2\rangle$ to $\|0\rangle$
(3)	Set the $\|c_1\rangle$ and $\|c_2\rangle$ qubits to the code for the generator state
(4)	for all $a \in A$ do
(5)	Split the term corresponding to the intermediate phase group from the generator state
(6)	Split the term into the intermediate phase group using controlled Hadamard gates
(7)	Transform the intermediate phase group into the generalized phase group
(8)	Change the codes for the terms in the generalized phase group to the codes for initialized terms
(9)	end for

where t is the phase that is multiplied by all the minterms in the group, g is the number of cells in the group on the phase map, and p is the number of minterms that still need to be added to the quantum superposition including those in the current group. Note that since this operator is always applied to the $|c_1\rangle$ and $|c_2\rangle$ qubits, a superposition containing a new generator state and a term that can be split into the current group will be created. Also, due to the nature of this algorithm, this operator will never be applied to a superposition containing codes for the current group. Thus, only the generator state will be modified and the terms in the quantum superposition that have already been initialized will not be changed.

C. High-level overview of the ESQUID algorithm

The pseudocode shown in Table I is intended only as a high level overview of the algorithm and ignores several important details. A complete and detailed description of the algorithm is given in Sec. VI.

V. SIMPLE EXAMPLE

This section illustrates the basic idea behind the ESQUID algorithm using a simple example. For this example, the state

$$|\psi\rangle = \tfrac{1}{2}|0101\rangle + \tfrac{1}{2}|0110\rangle - \tfrac{1}{2}|1001\rangle - \tfrac{1}{2}|1010\rangle \tag{8}$$

will be initialized. The phase map that corresponds to the state $|\psi\rangle$ is shown in Fig. 2. First observe that all of the minterms in Fig. 2 can be put into one generalized phase group which is denoted in Fig. 2 by the small circles connected with lines. This is because applying the quantum array in Fig. 3 transforms the intermediate phase group in Fig. 4 into the generalized phase group in Fig. 2. Note that the intermediate phase group that corresponds to the phase group in Fig. 4 can be denoted by the string $*1*1$ because the qubits $|x_1\rangle$ and $|x_3\rangle$ are both $|0\rangle$ and $|1\rangle$ within this phase group but the qubits $|x_2\rangle$ and $|x_4\rangle$ are constant within this

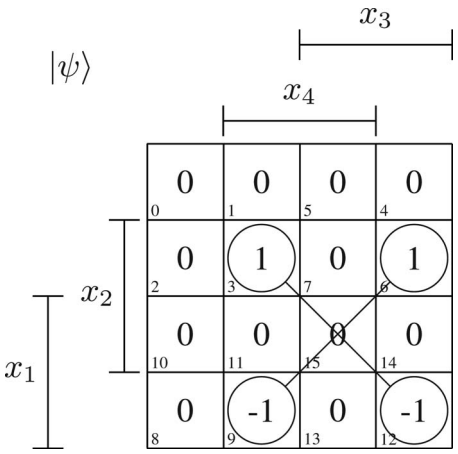

FIG. 2. The phase map for Eq. (8).

phase group. It is important to observe that this string does not completely describe the phase group because it does not account for phase. This phase group can also be written as a quantum superposition according to the form in Eq. (3) which results in

$$|\varphi\rangle = \tfrac{1}{2}|0101\rangle + \tfrac{1}{2}|0111\rangle - \tfrac{1}{2}|1101\rangle - \tfrac{1}{2}|1111\rangle \tag{9}$$

$$= \tfrac{1}{2}(|0\rangle - |1\rangle)|1\rangle(|0\rangle + |1\rangle)|1\rangle. \tag{10}$$

The quantum array for initializing the quantum superposition in Eq. (8) is therefore as shown in Fig. 5. The inverters denoted by G_1 and G_2 set the code to indicate that the only term in the quantum superposition is the generator state. Note that $|11\rangle$ is the code for the generator state from Sec. IV A. The state of the quantum array before any operators are applied is $|\psi_0\rangle = |0000,00\rangle$ so the state after G_1 and G_2 are applied is $|\psi_1\rangle = |0000,11\rangle$. Now consider the factored form in Eq. (10). The qubits $|x_2\rangle$ and $|x_4\rangle$ are both $|1\rangle$ in Eq. (10); these qubits must be set to $|1\rangle$ in the generator state so the controlled-NOT (CNOT) gates denoted by G_3 and G_4 are applied. Because the state of the qubit $|x_1\rangle$ is $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, it is necessary to set the state of the $|x_1\rangle$ qubit to $|1\rangle$ because a Hadamard gate will be applied later to this qubit to initialize the intermediate phase group in Eq. (10). To do this, the CNOT gate denoted by G_5 is applied. Note that inverters could have been used rather than these CNOT gates because at this point in the algorithm the generator state is the only term in the quantum superposition. However, this is not true during later stages of the algorithm. Applying these gates results in a new state $|\psi_2\rangle = |1101,11\rangle$. The gate $S_{1,4,4}$ denoted by G_6 is then applied to the c_1 and c_2 qubits. From Eq. (7),

$$S_{1,4,4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

so that the state becomes $|\psi_3\rangle = |1101,10\rangle$ after G_6 is applied. Note that applying $S_{1,4,4}$ has changed the code to $|10\rangle$ which is the code for the phase group that is currently being initialized as defined in Sec. IV A. The controlled Hadamard

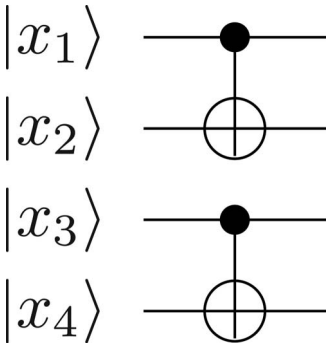

FIG. 3. The quantum array for transforming the intermediate phase group in Eq. (10) into the generalized phase group in Eq. (8).

gates denoted by G_7 and G_8 are applied to split the term that corresponds to the current phase group into the intermediate phase group from Eq. (10). This results in the state $|\psi_4\rangle=\frac{1}{2}(|0\rangle-|1\rangle)|1\rangle(|0\rangle+|1\rangle)|1,10\rangle$. It is now necessary to transform this intermediate phase group into the desired generalized phase group from Eq. (8). This can be done using the quantum array from Fig. 3 with controls added so that only the generalized phase group that is currently being initialized will be affected. Note that these controls are not necessary for the first generalized phase group. This is because all terms in the current quantum superposition are in the first generalized phase group since no other generalized phase groups are initialized before the first generalized phase group. However, these controls are required for generalized phase groups that are initialized after the first generalized phase group. First note that $|\psi_4\rangle$ can be rewritten as $|\psi_4\rangle=\frac{1}{2}|0101,10\rangle+\frac{1}{2}|0111,10\rangle-\frac{1}{2}|1101,10\rangle-\frac{1}{2}|1111,10\rangle$. The state after the gates G_9 and G_{10} are applied is therefore $|\psi_5\rangle=\frac{1}{2}|0101,10\rangle+\frac{1}{2}|0110,10\rangle-\frac{1}{2}|1001,10\rangle-\frac{1}{2}|1010,10\rangle$. The SWAP gate G_{11} is then applied to the terms in the quantum superposition that correspond to the generalized phase group that is currently being initialized. This changes the codes for these terms to the code for initialized terms. Because in this case all terms in the quantum superposition are in the group that is currently being initialized, it is not necessary to control the SWAP gate

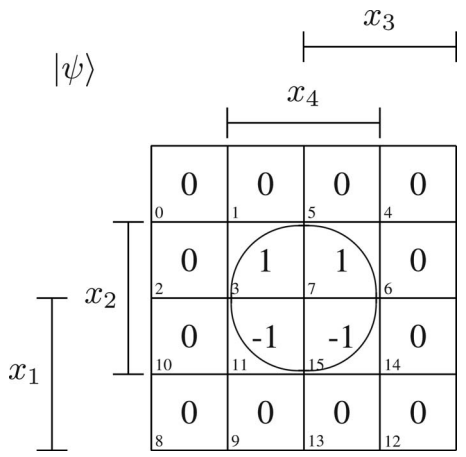

FIG. 4. The intermediate phase group for the generalized phase group in Eq. (8).

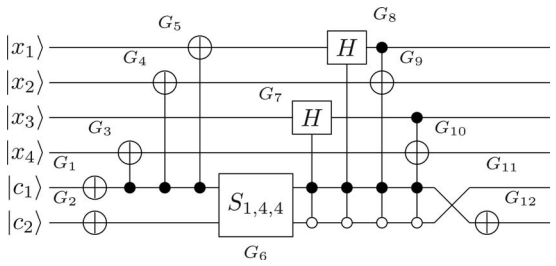

FIG. 5. The quantum array for initializing the quantum superposition from Eq. (8).

denoted by G_{11}. This results in the state $|\psi_6\rangle=\frac{1}{2}|0101,01\rangle+\frac{1}{2}|0110,01\rangle-\frac{1}{2}|1001,01\rangle-\frac{1}{2}|1010,01\rangle$. Finally, the inverter denoted by G_{12} is applied to restore states of the code qubits to $|0\rangle$. This results in the state $|\psi_7\rangle=\frac{1}{2}|0101,00\rangle+\frac{1}{2}|0110,00\rangle-\frac{1}{2}|1001,00\rangle-\frac{1}{2}|1010,00\rangle$ which is equal to the desired quantum superposition from Eq. (8) if the code qubits are ignored. Since the states of the code qubits have been restored to $|0\rangle$, the code qubits can be reused later for other tasks.

VI. DETAILED PSEUDOCODE

The algorithm will now be described using the detailed pseudocode in Table II. The qubits $|x_j\rangle$ are the qubits that the desired quantum superposition will be initialized on and the qubits $|c_1\rangle$ and $|c_2\rangle$ are used for storing the codes discussed in Sec IV A. The notation $s_a=s_{a,1},\ldots,s_{a,n}$ is used to denote the string that corresponds to the intermediate phase group for the generalized phase group a and U_a is the unitary operator from Eq. (5).

Theorem 1. The ESQUID algorithm initializes the desired quantum superposition.

Proof. Let $|\psi_i\rangle=|x_{i,1},\ldots,x_{i,n},c_{i,1}c_{i,2}\rangle$ denote the state of the quantum array after the ith iteration has been completed where $|\psi_0\rangle$ is defined to be the state immediately before the first iteration. Let m be the number of terms in the desired quantum superposition from Eq. (1) and let t_i be the value that is assigned to t at the ith iteration of the algorithm. Let g_i be the number of minterms that are initialized in the quantum superposition during the ith iteration of the algorithm and let p_i be the total number of minterms that need to be initialized in the quantum superposition by the ith iteration and all subsequent iterations of the algorithm. Let b be the total number of generalized phase groups that are initialized, let $|a_i\rangle$ be the quantum superposition that corresponds to the generalized phase group that is initialized at the ith iteration of the algorithm, and let $|s_{a_i}\rangle$ be the intermediate phase group that corresponds to a_i as defined in Eq. (5). Let q_i be the binary string of length n that corresponds to the states of the $|x_j\rangle$ qubits in the generator state before line 30. Let q_0 be the starting states of the qubits $|x_j\rangle$ in the generator state. It will be proven by induction on i that the state after the ith iteration is

$$|\psi_i\rangle=\sum_{k=1}^{i}\frac{1}{\sqrt{m}}|a_k,01\rangle+\sqrt{\frac{p_i-g_i}{m}}|q_i,11\rangle. \quad (11)$$

Consider the basis case where $i=0$. By definition, this is the quantum state immediately before the first iteration which is

TABLE II. Algorithm II: Detailed pseudocode for the ESQUID algorithm.

(1) Find a small set A of generalized phase groups using logic synthesis methods
(2) Initialize the state to $|0^n,00\rangle$
(3) Apply inverters to $|c_1\rangle$ and $|c_2\rangle$; this results in $|\psi_0\rangle = |0^n,11\rangle$
(4) for all $a \in A$ do
(5) Let g be the number of minterms in intermediate phase group that corresponds to s_a
(6) Let p be the number of terms in the quantum superposition that have not been initialized yet including the terms about to be initialized in the current group
(7) Find the state that corresponds to s_a using the form shown in Eq. (3)
(8) if the sign outside the product is positive then
(9) Let $t=1$
(10) else
(11) Let $t=-1$
(12) end if
(13) for all $j=1,\ldots,n$ do
(14) if $s_{a,j}=*$ then
(15) if the sign in Eq. (4) for the jth qubit in the intermediate phase group is positive then
(16) if the jth qubit is $|1\rangle$ in the generator state then
(17) Apply an inverter controlled by $|1\rangle$ on $|c_1\rangle$ to the jth qubit
(18) end if
(19) else
(20) if the jth qubit is $|0\rangle$ in the generator state then
(21) Apply an inverter controlled by $|1\rangle$ on $|c_1\rangle$ to the jth qubit
(22) end if
(23) end if
(24) else
(25) if the jth qubit in the generator state is not equal to $|s_{a,j}\rangle$ then
(26) Apply an inverter controlled by $|1\rangle$ on $|c_1\rangle$ to the jth qubit
(27) end if
(28) end if
(29) end for
(30) Apply $S_{t,g,p}$ to $|c_1c_2\rangle$
(31) for all $j=1,\ldots,n$ do
(32) if $s_{a,j}=*$ then
(33) Apply a Hadamard gate controlled by $|10\rangle$ on $|c_1c_2\rangle$ to the jth qubit
(34) end if
(35) end for
(36) Apply a U_a operation controlled by $|10\rangle$ on $|c_1c_2\rangle$ to $|x_1,\ldots,x_n\rangle$
(37) Apply $U_a^\dagger$ to $|x_1,\ldots,x_n\rangle$
(38) Apply a SWAP gate controlled by $s_{a,j}$ on each $|x_j\rangle$ where $s_{a,j}\neq *$ to $|c_1c_2\rangle$
(39) Apply U_a to $|x_1,\ldots,x_n\rangle$
(40) end for
(41) Apply an inverter to $|c_2\rangle$

the quantum state after the operations on line 3 of the algorithm are applied. Line 1 does not affect the state. After line 2, the quantum state is $|0^n,00\rangle$. Note that there is no 0th iteration of the algorithm so the number of minterms initialized in the 0th iteration is 0. Also, since no minterms have been initialized yet, the number of minterms that still need to be initialized is m. Thus, $g_0=0$ and $p_0=m$. Using these results, the state after line 3 is applied is

$$|\psi_0\rangle = |0^n,11\rangle \tag{12}$$

$$= \sum_{k=1}^{0} \frac{1}{\sqrt{m}}|a_k,01\rangle + \sqrt{\frac{p_0-g_0}{m}}|q_0,11\rangle \tag{13}$$

so the basis case holds. The inductive case will now be proven. Assume that Eq. (11) holds for the ith iteration. Lines 5–11 update the parameters g, p, and t to g_{i+1}, p_{i+1}, and t_{i+1}. Let the factored form of the intermediate phase group be $\pm\otimes_{j=0}^{n}|\beta_{i+1,j}\rangle$, where $|\beta_{i+1,j}\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. After lines 13–26 are run the following properties hold:

(i) If $s_{a_{i+1},j}=*$ and the sign in $|\beta_{i+1,j}\rangle$ is positive then the jth qubit in the generator state is $|0\rangle$. This is because the qubits in the generator state are always either $|0\rangle$ or $|1\rangle$ so if the jth qubit in the generator state is not $|1\rangle$, it must be $|0\rangle$. However, since the sign in $|\beta_{i+1,j}\rangle$ is positive, the jth qubit in the generator state cannot be $|1\rangle$ because if it was $|1\rangle$ it would have been set to $|0\rangle$ by line 17. Hence, $q_{i+1,j}=0$ in this case.

(ii) If $s_{a_{i+1},j}=*$ and the sign in $|\beta_{i+1,j}\rangle$ is negative then the jth qubit in the generator state is $|1\rangle$. This is because the qubits in the generator state are always either $|0\rangle$ or $|1\rangle$ so if the jth qubit in the generator state is not $|0\rangle$, it must be $|1\rangle$. However, since the sign in $|\beta_{i+1,j}\rangle$ is negative, the jth qubit in the generator state cannot be $|0\rangle$ because if it was $|0\rangle$ it would have been set to $|1\rangle$ by line 21. Hence, $q_{i+1,j}=1$ in this case.

(iii) If $s_{a_{i+1},j}\neq *$, the jth qubit in the generator state is set to $s_{a_{i+1},j}$ on line 26. Hence, $q_{i+1,j}=s_{a_{i+1},j}$ in this case.

These operations affect only the generator state because they are controlled by $|1\rangle$ on $|c_1\rangle$ and from Eq. (11), the qubit $|c_1\rangle$ is $|1\rangle$ only in the generator state. After the loop on line 13 finishes, the state is therefore

$$|\delta_{i+1}\rangle = \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + \sqrt{\frac{p_{i+1}}{m}}|q_{i+1},11\rangle. \tag{14}$$

Note that $p_{i+1}=p_i-g_i$ because all minterms that are not initialized at the ith iteration must be initialized by the $(i+1)$th or later iterations. Now consider line 30. By definition, t_{i+1}, g_{i+1}, and p_{i+1} are the values assigned to the parameters t, g, and p at the $(i+1)$th iteration of the algorithm. Hence, line 30 applies the operator $S_{t_{i+1},g_{i+1},p_{i+1}}$ to $|c_1c_2\rangle$. From Eq. (7), this operator does not affect the code $|01\rangle$ so the state is now

$$|\zeta_{i+1}\rangle = \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + \sqrt{\frac{p_{i+1}}{m}}|q_{i+1}\rangle \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{p_{i+1}-g_{i+1}}{p_{i+1}}} & t_{i+1}\sqrt{\frac{g_{i+1}}{p_{i+1}}} \\ 0 & 0 & -t_{i+1}\sqrt{\frac{g_{i+1}}{p_{i+1}}} & \sqrt{\frac{p_{i+1}-g_{i+1}}{p_{i+1}}} \end{bmatrix} |11\rangle \right) \tag{15}$$

$$= \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + \sqrt{\frac{p_{i+1}}{m}}|q_{i+1}\rangle \left(t_{i+1}\sqrt{\frac{g_{i+1}}{p_{i+1}}}|10\rangle + \sqrt{\frac{p_{i+1}-g_{i+1}}{p_{i+1}}}|11\rangle \right) \tag{16}$$

$$= \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + t_{i+1}\sqrt{\frac{g_{i+1}}{m}}|q_{i+1},10\rangle + \sqrt{\frac{p_{i+1}-g_{i+1}}{m}}|q_{i+1},11\rangle. \tag{17}$$

The loop on line 31 applies a Hadamard gate to each qubit if $s_{a_{i+1},j}=*$. Since $q_{i+1,j}=0$ if $s_{a_{i+1},j}=*$ and the sign in $|\beta_{i+1,j}\rangle$ is positive and $H|0\rangle=\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$, applying a Hadamard gate results in the correct phase in this case. Also, since $q_{i+1,j}=1$ if $s_{a_{i+1},j}=*$ and the sign in $|\beta_{i+1,j}\rangle$ is negative and $H|1\rangle=\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle$, applying a Hadamard gate also results in the correct phase in this case. If $s_{a_{i+1},j}\neq *$, then $s_{a_{i+1},j}=q_{i+1,j}$. Therefore, after the loop on line 31 runs, the state is

$$|\xi_{i+1}\rangle = \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + t_{i+1}\frac{1}{\sqrt{m}}\left(\bigotimes_{j=1}^{n}|\beta_{i+1,j}\rangle\right)|10\rangle + \sqrt{\frac{p_{i+1}-g_{i+1}}{m}}|q_{i+1},11\rangle \tag{18}$$

$$= \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + \frac{1}{\sqrt{m}}|s_{a_{i+1}},10\rangle + \sqrt{\frac{p_{i+1}-g_{i+1}}{m}}|q_{i+1},11\rangle. \tag{19}$$

Note that these Hadamard gates only affect the terms with the code $|10\rangle$ because of the controls on the Hadamard gates. Line 36 transforms the intermediate phase group into the generalized phase group. This results in the state

$$|\phi_{i+1}\rangle = \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + \frac{1}{\sqrt{m}}|a_{i+1},10\rangle + \sqrt{\frac{p_{i+1}-g_{i+1}}{m}}|q_{i+1},11\rangle. \tag{20}$$

Observe that since $U_{a_{i+1}}$ is a permutative matrix, it performs a one-to-one mapping from the set of all basis states in the intermediate phase group to the set of all basis states in the generalized phase group. Thus, the operator $U^{\dagger}_{a_{i+1}}$ maps a basis vector to basis vector in the intermediate phase group if and only if the basis vector is in the generalized phase group. Therefore, line 37 causes the SWAP operation on line 38 to be applied to the terms in the generalized phase group that was just initialized and line 39 restores the state of the $|x_j\rangle$ qubits. Note that since the generalized phase groups that were previously initialized are disjoint from this intermediate phase group, this SWAP gate will not be applied to these generalized phase groups. Thus, the state after lines 37–39 is

$$|\omega_{i+1}\rangle = \sum_{k=1}^{i} \frac{1}{\sqrt{m}}|a_k,01\rangle + \frac{1}{\sqrt{m}}|a_{i+1},01\rangle + \sqrt{\frac{p_{i+1}-g_{i+1}}{m}}|q_{i+1},11\rangle \tag{21}$$

$$= \sum_{k=1}^{i+1} \frac{1}{\sqrt{m}}|a_k,01\rangle + \sqrt{\frac{p_{i+1}-g_{i+1}}{m}}|q_{i+1},11\rangle \tag{22}$$

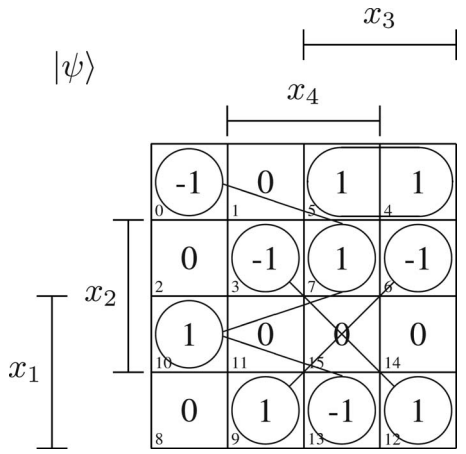

FIG. 6. The phase map for Eq. (26).

$$= |\psi_{i+1}\rangle. \tag{23}$$

Thus, the inductive case is proven. Therefore, by the principle of mathematical induction, the state after the ith iteration is as shown in Eq. (11). Applying Eq. (11) for the final bth iteration results in

$$|\psi_b\rangle = \sum_{k=1}^{b} \frac{1}{\sqrt{m}}|a_k,01\rangle + \sqrt{\frac{p_b - g_b}{m}}|q_b,11\rangle \tag{24}$$

$$= \sum_{k=1}^{b} \frac{1}{\sqrt{m}}|a_k,01\rangle. \tag{25}$$

Note that this is because $p_b = g_b$ since the bth iteration is the last iteration. Line 41 applies an inverter to the $|c_2\rangle$ qubit which results in the state $\Sigma_{k=1}^{b}\frac{1}{\sqrt{m}}|a_k,00\rangle$ which is equal to the desired state $|\psi\rangle$ from Eq. (1) with the addition of two ancilla qubits set to $|0\rangle$. Therefore, the ESQUID algorithm initializes the desired state. ■

VII. MORE COMPLEX EXAMPLE

This section will show how to use the ESQUID algorithm to initialize the quantum superposition,

$$\begin{aligned}|\psi\rangle = &-\frac{1}{\sqrt{10}}|0000\rangle + \frac{1}{\sqrt{10}}|0010\rangle + \frac{1}{\sqrt{10}}|0011\rangle - \frac{1}{\sqrt{10}}|0101\rangle \\ &- \frac{1}{\sqrt{10}}|0110\rangle + \frac{1}{\sqrt{10}}|0111\rangle + \frac{1}{\sqrt{10}}|1100\rangle + \frac{1}{\sqrt{10}}|1001\rangle \\ &+ \frac{1}{\sqrt{10}}|1010\rangle - \frac{1}{\sqrt{10}}|1011\rangle.\end{aligned} \tag{26}$$

This is much more complicated than for the example in Sec. V as two generalized phase groups and one phase group are required to initialize this quantum superposition. The phase map for Eq. (26) is shown in Fig. 6. In Fig. 6, two generalized phase groups and one phase group are used to represent the quantum superposition. Note that a phase group is also a generalized phase group so ESQUID can be used to initialize

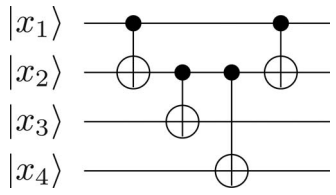

FIG. 7. The quantum array for transforming the intermediate phase group in Eq. (32) into the generalized phase group in Eq. (30).

this quantum superposition. The phase map itself does not specify the order in which the generalized phase groups are initialized although an order must be chosen so that the previously discussed constraints are satisfied. Since the order depends on the intermediate phase groups, the order will be chosen after the intermediate phase groups have been selected. The generalized phase group

$$|\alpha\rangle = -\tfrac{1}{2}|0101\rangle - \tfrac{1}{2}|0110\rangle + \tfrac{1}{2}|1001\rangle + \tfrac{1}{2}|1010\rangle \tag{27}$$

can be initialized from the intermediate phase group

$$|\beta\rangle = -\tfrac{1}{2}|0101\rangle - \tfrac{1}{2}|0111\rangle + \tfrac{1}{2}|1101\rangle + \tfrac{1}{2}|1111\rangle \tag{28}$$

$$= -\tfrac{1}{2}(|0\rangle - |1\rangle)|1\rangle(|0\rangle + |1\rangle)|1\rangle \tag{29}$$

using the quantum array in Fig. 3. The generalized phase group

$$|\gamma\rangle = -\tfrac{1}{2}|0000\rangle + \tfrac{1}{2}|0111\rangle + \tfrac{1}{2}|1100\rangle - \tfrac{1}{2}|1011\rangle \tag{30}$$

can be initialized from the intermediate phase group

$$|\delta\rangle = -\tfrac{1}{2}|0000\rangle + \tfrac{1}{2}|0100\rangle + \tfrac{1}{2}|1100\rangle - \tfrac{1}{2}|1000\rangle \tag{31}$$

$$= -\tfrac{1}{2}(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)|00\rangle \tag{32}$$

using the quantum array in Fig. 7. Now the intermediate phase group $|\beta\rangle$ overlaps with the generalized phase group $|\gamma\rangle$. As mentioned in Sec. III C, each intermediate phase group must be disjoint from all generalized phase groups that have already been initialized. This implies that the generalized phase group $|\alpha\rangle$ must be initialized before the generalized phase group $|\gamma\rangle$ is initialized. The phase group

$$|\zeta\rangle = \frac{1}{\sqrt{2}}|0010\rangle + \frac{1}{\sqrt{2}}|0011\rangle \tag{33}$$

$$= \frac{1}{\sqrt{2}}|001\rangle(|0\rangle + |1\rangle) \tag{34}$$

can be initialized at any point in the algorithm. If $|\alpha\rangle$ is initialized first, $|\gamma\rangle$ is initialized second and $|\zeta\rangle$ is initialized last, the ESQUID algorithm generates the quantum array in Fig. 8. Note that the two CNOT gates above the second controlled SWAP gate in Fig. 8 are unnecessary and can be removed from the quantum array. However, these gates have

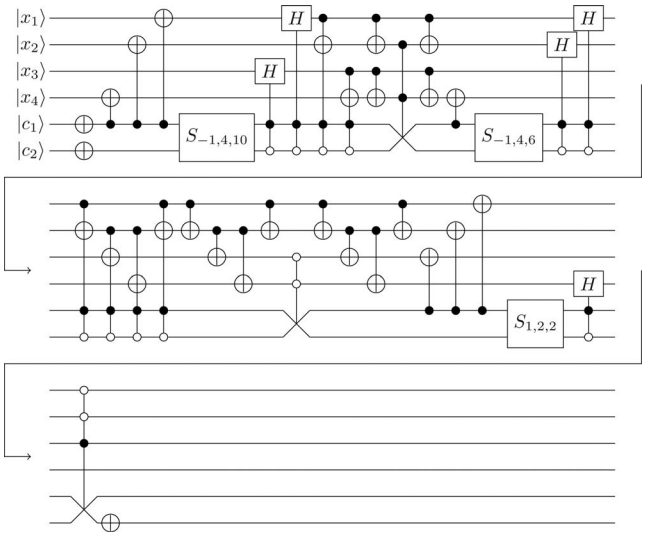

FIG. 8. The quantum array for initializing the quantum superposition in Eq. (26).

not been removed in order to demonstrate the operation of the algorithm. Calculating the quantum states that result from applying this quantum array to the starting state $|0^n,00\rangle$ confirms that the ESQUID algorithm works correctly.

VIII. COMPLEXITY OF THE ESQUID ALGORITHM

This section will analyze the complexity of the ESQUID algorithm in terms of the number of one and two qubit operations required. The number of iterations required by the ESQUID algorithm will be denoted by b. It is assumed that applying the b required initialization operators requires a total of $O(bn)$ one and two qubit operations. It is also assumed that applying the $2b$ U operators from Eq. (5) and the b required $U^\dagger$ operators requires a total of $O(bn)$ one and two qubit operations. The operations on line 3 require two inverters and hence two single qubit gates. Lines 5–11 do not require any operations. Lines 13–26 require $O(n)$ CNOT gates. Since this is repeated b times, this requires a total of $O(bn)$ two qubit gates. Line 30 requires a total of $O(bn)$ one and two qubit gates by assumption. Lines 31–33 use n Hadamard gates with two controls. These gates can be implemented using five two qubit gates [8] so since this is repeated b times, this requires $O(bn)$ two qubit gates. By assumption, lines 36–39 use $O(bn)$ one and two qubit gates to implement the U and $U^\dagger$ operators. Implementing the controlled SWAP operations requires $O(bn)$ two qubit gates if n ancilla qubits are used for this purpose since a controlled SWAP gate can be implemented using two CNOT gates and a Toffoli gate which can be implemented using the method in the Ventura-Martinez algorithm [4]. Since line 41 uses only one single qubit gate, the entire algorithm requires $O(bn)$ one and two qubit gates. If the Toffoli gates used to implement the controlled SWAP gates in the ESQUID algorithm are implemented using a quadratic number of two qubit gates [8], then the algorithm requires a total of $O(bn^2)$ one and two qubit gates. In this case, the b initialization operators used by the algorithm must use $O(bn^2)$ one and two qubit gates and the $2b$ U operators from Eq. (5) and the b required $U^\dagger$ operators must also use $O(bn^2)$ one and two qubit gates. This means that the initialization and U operators from Eq. (5) may use more gates when ancilla qubits are not used to implement the Toffoli gates required for constructing the controlled SWAP gates. This version of the ESQUID algorithm will be referred to as the modified ESQUID algorithm for the remainder of this paper.

IX. COMPARISON OF INITIALIZATION ALGORITHMS

The ESQUID algorithm is compared with the SQUID [6], Ventura-Martinez [4], and Long-Sun [5] algorithms in Table III where n is the number of qubits in the desired quantum superposition, m is the number of minterms in the quantum superposition, p is the number of phase groups required by the SQUID algorithm [6], and b is the number of generalized phase groups required by the ESQUID algorithm. In Table III, the modified ESQUID algorithm is as described in Sec. VIII; similarly, the modified SQUID algorithm [6] uses a quadratic number of one and two qubit gates [8] to implement the Toffoli gates required for constructing the controlled SWAP gates used in the SQUID algorithm [6]. The modified Ventura-Martinez algorithm [4] also uses a quadratic number of one and two qubit gates [8] to implement the Toffoli gates required for the algorithm. Because in the worst case, a phase group contains only a single minterm, $p \leq m$. Since any generalized phase group is also a phase

TABLE III. Comparison of the complexity of different quantum initialization algorithms where m, p, and b are the numbers of iterations required by the Ventura-Martinez, SQUID, and ESQUID algorithms, respectively, and n is the number of qubits in the desired quantum superposition.

Algorithm	Worst case	Best case	Total qubits
Long-sun algorithm	$\Theta(n^2 2^n)$	$\Theta(n^2 2^n)$	n
Ventura-Martinez algorithm	$\Theta(mn)$	$\Theta(mn)$	$2n+1$
Modified Ventura-Martinez algorithm	$\Theta(mn^2)$	$\Theta(mn^2)$	$n+2$
SQUID algorithm	$O(pn)$	$O(n)$	$2n+2$
Modified SQUID algorithm	$O(pn^2)$	$O(n)$	$n+2$
ESQUID algorithm	$O(bn)$	$O(n)$	$2n+2$
Modified ESQUID algorithm	$O(bn^2)$	$O(n)$	$n+2$

group, b can always be selected so that $b \leq p$ which implies that $b \leq p \leq m$. Thus, the quantum arrays generated by the ESQUID algorithm will never use more gates than quantum arrays generated by the SQUID algorithm [6] which will never use more gates than quantum arrays generated by the Ventura-Martinez [4] and Long-Sun [5] algorithms. Hence, quantum arrays generated by the ESQUID algorithm will never use more gates than any existing algorithm. For most quantum superpositions, the ESQUID algorithm will use far less gates than existing algorithms due to the increased flexibility of generalized phase groups over phase groups. The best case performance for quantum arrays generated by the ESQUID algorithm is the same as for quantum arrays generated by the SQUID algorithm [6] and is an exponential improvement over all other initialization algorithms. The best case for quantum arrays generated by the ESQUID algorithm will also occur more often than the best case performance for quantum arrays generated by the SQUID algorithm [6] since there are far more quantum superpositions that can be represented by a single generalized phase group than by a single phase group. This makes the ESQUID algorithm much more efficient than all other algorithms for initializing quantum superpositions including the SQUID algorithm [6].

X. CONCLUSION

The ESQUID algorithm generates quantum arrays that are much more efficient than the quantum arrays generated by other quantum initialization algorithms for almost all quantum superpositions. Furthermore, quantum arrays generated by the ESQUID algorithm never require more gates than quantum arrays generated by other algorithms assuming that the generalized phase groups are selected as described in Sec. IX. As with the SQUID algorithm [6], the ESQUID algorithm provides an exponential improvement in the number of gates required in the generated quantum arrays in the best case. Because the generalized phase groups introduced in this paper are more general than phase groups, the best case for the ESQUID algorithm occurs for more quantum superpositions than the best case for the SQUID algorithm. Due to these properties, the ESQUID algorithm is much better for initializing quantum superpositions efficiently than other initialization algorithms.

[1] L. K. Grover, *Proceedings of the Annual ACM Symposium on Theory of Computing* (ACM, New York, 1996), p. 212.
[2] D. Ventura and T. Martinez, Inf. Sci. (N.Y.) **124**, 273 (1999).
[3] A. A. Ezhov, A. V. Nifanova, and D. Ventura, Inf. Sci. (N.Y.) **128**, 271 (1999).
[4] D. Ventura and T. Martinez, Found. Phys. Lett. **12**, 547 (1999).
[5] G.-L. Long and Y. Sun, Phys. Rev. A **64**, 014303 (2001).
[6] D. J. Rosenbaum and M. A. Perkowski, *Proceedings of the 38th International Symposium on Multiple Valued Logic* (IEEE, Piscataway, NJ, 2008), p. 144.
[7] L. J. Schulman and U. V. Vazirani, *Conference Proceedings of the Annual ACM Symposium on Theory of Computing* (ACM, New York, 1999), p. 322.
[8] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995).