Superposed Quantum State Initialization Using Disjoint Prime Implicants (SQUID)

David Rosenbaum[1], Marek Perkowski[2]
[1]Department of Computer Science, Portland State University;
[2]Department of Electrical Engineering, Portland State University
(E-mail: drosenba@cs.pdx.edu, mperkows@ee.pdx.edu)

Abstract

The problem of initializing a quantum superposition is important for Grover's Algorithm [1], Quantum Neural Networks [2] and other applications. The purpose of the algorithm presented here is to generate a quantum array that initializes a desired quantum superposition on n qubits. The SQUID algorithm almost always creates quantum arrays that perform better than those created by existing algorithms such as the Ventura-Martinez [3] and Long-Sun [4] algorithms. The best case performance for the quantum arrays created by the SQUID algorithm is $O(n)$ when the superposition contains all possible states which is an exponential improvement over all existing algorithms. Also, the worst case performance of the quantum array created by the SQUID algorithm is never worse than the performance of existing algorithms. The SQUID algorithm represents a vast improvement over previous quantum superposition initialization algorithms and allows quantum superpositions to be initialized much more efficiently than with other algorithms.

1. Introduction

The problem of initializing a quantum superposition is important for quantum algorithms such as Grover's algorithm [1] and Quantum Neural Networks [2]. Ventura and Martinez [3] created an algorithm that generates a quantum array capable of initializing a quantum superposition of the form $|\Psi\rangle = \sum_{i=0}^{2^n-1} \frac{t_i}{\sqrt{m}} |i\rangle$ in the complexity class $O(mn)$ with $n+1$ ancilla bits. Long and Sun [4] created another algorithm that introduced a new method for solving a similar problem without using any ancilla bits but with an exponential complexity class for the generated quantum array. An advantage of the Long-Sun algorithm 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 goal of the SQUID algorithm is to improve the efficiency of the generated quantum array while adding very few ancilla bits.

2. Initializing the Starting State

The algorithm presented in this paper requires an initial state of $|0^n, 00\rangle$. This requires another algorithm to be run first to initialize the state to $|0^n, 00\rangle$. One method for initializing to this state is the Schulman-Vazirani heat engine [5]. The rest of this paper will assume that this starting state has been initialized and will focus on initializing the desired superposition from this state.

3. The Desired Quantum Superposition

The desired quantum superposition must be of the form $|\Psi\rangle = \sum_{i=0}^{2^n-1} \frac{t_i}{\sqrt{m}} |i\rangle$ (1) where $t_i \in \{-1, 0, 1\}$ and m is the number of terms with nonzero amplitudes in the desired quantum superposition.

3.1. The Phase Map

The algorithm uses a special Karnaugh Map which we propose to call a phase map to represent a desired quantum superposition of the form shown in (1). The indexes on the side of the map indicate the state in the desired superposition that the cell corresponds to. Each cell on the phase map contains the phase for its corresponding term so the number in a cell is $t_i \in \{-1,0,1\}$. For example, the state $\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$ (2) can be represented using a phase map as shown in Figure 1:

$x_1 \backslash x_2$	0	1
0	0	1
1	-1	0

Figure 1: The phase map for (2)

The phase map also uses a new type of product group which we propose to call a phase group to represent sets of minterms that will be initialized together in the algorithm. Some restrictions apply to the minterms that can form a phase group. It must be possible to write the phase group as a string s which contains the symbols 0 and 1 for the qubits that do not vary within the group and the symbol * for the qubits that do vary within the group. Note that this string does not contain any information about the phases of the minterms so it does not completely describe the group; however it is used later for defining the algorithm. These phase groups must be disjoint which means that different phase groups may not overlap. Also, it must be possible to write the superposition corresponding to a phase group as $|s\rangle = \pm \prod_{j=1}^{n} |r_j\rangle$ (3) where $|r_j\rangle = |s_j\rangle$ if $s_j \neq *$ and $|r_j\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$ (4) if $s_j = *$. The sign in $|r_j\rangle$ depends on the desired phases for terms in the desired quantum superposition. This implies that each phase group has an equal number of the values -1 and 1 in its cells. Thus, phase groups have the same shapes as groups in a standard Karnaugh Map. Also, minterms with amplitudes of 0 may not be present in any phase group since phase groups must be disjoint.

4. The SQUID Algorithm

The quantum array generated by the SQUID algorithm uses two groups of qubits. The first group qubits are denoted by $x_i, i=1 \ldots n$ and are the qubits over which the desired superposition is initialized. The second group of qubits are ancilla bits used for storing codes and are denoted by c_1 and c_2.

4.1. Codes in the Ancilla Bits

The qubits c_1 and c_2 (also called the code qubits) are used for keeping track of

- which terms in the superposition have been initialized,
- which are currently being initialized and
- which terms will be used later to create more terms in the superposition (this is called the generator state [3]).

The following codes are used:

- $|00\rangle$ on the c_1 and c_2 qubits is not used.
- $|01\rangle$ on the c_1 and c_2 qubits is used to indicate that the corresponding terms in the quantum superposition have been initialized to the proper values and should not be modified again by the algorithm.
- $|10\rangle$ indicates that the corresponding terms in the superposition are part of the group that is currently being initialized.
- $|11\rangle$ 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 facts.

4.2. The Initialization Operator

The operator used in this algorithm is based on the operator from the Ventura-Martinez algorithm [3]. 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 state in the superposition and splitting it using controlled Hadamard gates. The new operator is defined by equation (5)

$$S_{t,g,p}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \sqrt{\frac{p-g}{p}} & t\sqrt{\frac{g}{p}}\\ 0 & 0 & -t\sqrt{\frac{g}{p}} & \sqrt{\frac{p-g}{p}}\end{bmatrix} \quad (5)$$

This operator works by splitting state corresponding to the groups off from the generator state during the algorithm 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 number of minterms that still need to be added to the superposition including those in the current group. Note that since this operator is always applied to the c_1 and c_2 qubits, a superposition containing a new generator state and a state 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 superposition that have already been initialized will not be changed.

4.3. A High-Level Overview of the Algorithm

The following 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 later.

```
1 Find a small set of groups G using logic synthesis methods.
2 Initialize all $x_i$ and $c_j$ qubits to $|0\rangle$ .
3 Set the $c_1$ and $c_2$ qubits to the code for the generator state.
4 For each group in G:
    4.1 Split the term corresponding to the group from the generator state using the initialization operator.
    4.2 Split the group into its corresponding minterms.
    4.3 Change the codes for the terms in the current group to the code for an initialized term in the superposition.
```

4.4. Example 1

This example involves initializing the quantum state to $|\Psi\rangle=\frac{1}{2}|00\rangle-\frac{1}{2}|01\rangle+\frac{1}{2}|10\rangle-\frac{1}{2}|11\rangle$ (6). The first step is to write the superposition as a phase map as shown in figure 2 so that the optimal set of groups can be found.

x_1 \ x_2	0	1
0	1	-1
1	1	-1

Figure 2: The phase map for (6)

Note that this superposition can also be written in the factored form shown in (3) as $|\Psi\rangle=\frac{1}{2}(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)$ (7) so it satisfies the factoring requirements. The circuit generated by the algorithm for this superposition is shown in figure 3:

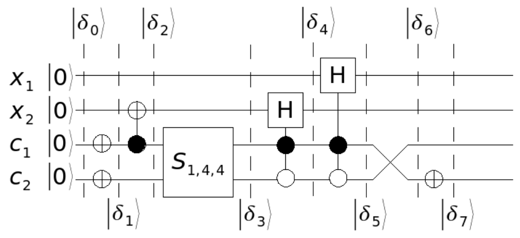

Figure 3: The Circuit for (6)

The initial state is $|\delta_0\rangle = |00,00\rangle$. Two inverters are applied to the c_1 and c_2 qubits of this state to set the code to the generator state. Conceptually, no terms have been initialized in the superposition so this term must be the generator state. This results in a new state of $|\delta_1\rangle = |00,11\rangle$. Notice that in the factored superposition for this group, the first qubit is $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and the second qubit is $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Observe that these states can be created by applying Hadamard gates to the states $|0\rangle$ and $|1\rangle$ respectively. However, the second qubit in the generator state is in state $|0\rangle$. Thus, an inverter is applied in order to set it to $|1\rangle$ in the generator state. The inverter is controlled by $|1\rangle$ on c_1 because at this stage of the algorithm the code qubits always contain a superposition of zero or more codes for initialized terms and exactly one term with the code qubits set to the code for the generator state. Since the c_1 is $|1\rangle$ in the code for the generator state but not in the code for an initialized term, controlling by $|1\rangle$ on c_1 affects only the generator state. This results in a new state of $|\delta_2\rangle = |01,11\rangle$. The initialization operator is now applied to split the current group from the superposition. For this initialization operator, $t=1$ because there is no minus sign outside the factored superposition that corresponds to the current group. Also, $g=4$ because the current group contains 4 minterms and $p=4$ because there are 4 minterms left to add to the superposition including those in the current group. Now

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

so $S_{1,4,4}|11\rangle = |10\rangle$. Thus the new state is $|\delta_3\rangle = |01,10\rangle$. Note that since all of the remaining amplitude will be used to initialize the minterms associated with the current group, the code has simply been changed to the code for the current group by the initialization operator. The next two gates split the current group into individual minterms. This results in $|\delta_4\rangle = |0\rangle\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)|10\rangle$ and

$$|\delta_5\rangle = |0\rangle\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)|10\rangle .$$

The next step involves applying a swap gate to update the codes for the current group. Since the current group contains all possible terms, no controls are used on the swap gate. The state becomes

$$|\delta_6\rangle = |0\rangle\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)|01\rangle .$$

At the end of the algorithm an inverter is applied to c_2 to restore the state of the c_2 qubit to $|0\rangle$. The final state is

$$|\delta_7\rangle = |0\rangle\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)|00\rangle$$
$$= \left(\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle\right)|00\rangle$$

which is the desired superposition with the addition of two ancilla bits that can be reused later. Note that in this case where all possible minterms can be put in one group, the SQUID algorithm uses $O(n)$ controlled single-qubit gates, which is an exponential improvement over the Ventura-Martinez and Long-Sun algorithms.

4.5. Pseudo Code for The Algorithm

The algorithm will now be described using detailed pseudo code. In this algorithm, $|\Psi_j\rangle = |x, c_1 c_2\rangle$ represents the state of the circuit at the jth iteration and $|x\rangle$ represents the state of the n qubits that will be initialized to the desired quantum superposition and c_1 and c_2 are ancilla bits. $|\Psi_0\rangle$ represents the initial state before the 1st iteration. Although a correctness proof exists for this algorithm, it is too long to show in this paper.

```
1 Find a small set of groups G where each $s=s_1s_2...s_n \in G$ is the binary string described in section 3.1 that represents each group.
2 Initialize the state to $|\Psi_0\rangle = |x, c_1c_2\rangle = |0^n, 00\rangle$
3 Apply inverters to $c_1$ and $c_2$. The state is now $|\Psi_0\rangle = |x, c_1c_2\rangle = |0^n, 11\rangle$.
4 For $s=s_1s_2...s_n \in G$
    4.1 Let $g$ denote the number of minterms in the group $s$.
    4.2 Let $p$ be the number of terms in the superposition that have not been initialized yet including the terms about to be initialized in the current group.
    4.3 Find the superposition that corresponds to $s$ as described in step 1. This results in (3).
    4.4 If the sign outside the product in (3) is +, let $t=1$. Otherwise, if the sign is -, let $t=-1$.
    4.5 For $j=1...n$
        4.5.1 If $s_j=*$
            4.5.1.1 If the sign in (4) is +
                4.5.1.1.1 If the jth qubit is set to $|1\rangle$ in the generator state, apply an inverter to the jth qubit controlling by $|1\rangle$ on $c_1$.
            4.5.1.2 If the sign in (4) is -
                4.5.1.2.1 If the jth qubit is set to $|0\rangle$ in the generator state, apply an inverter to the jth qubit controlling by $|1\rangle$ on $c_1$.
        4.5.2 Otherwise, if $s_j \neq *$ apply an inverter if the state of the jth qubit in the generator state is not equal to $s_j$ controlling by $|1\rangle$ on $c_1$.
    4.6 Apply $S_{t,g,p}$ to the $c_1$ and $c_2$ qubits. This splits a new state off the generator state that can be used to create the terms in the current group.
    4.7 For $j=1...n$
        4.7.1 If $s_j=*$ apply a Hadamard gate to the jth qubit controlling by $|10\rangle$ on $|c_1c_2\rangle$.
    4.8 Apply a swap gate to the $c_1$ and $c_2$ qubits controlling by $s_j$ on each $x_j$ where $s_j \neq *$. This changes the codes of the terms from the current group to the code for initialized terms. This freezes these terms so they will not be modified by the rest of the algorithm.
5 Apply an inverter to $c_2$.
```

5. Comparison with Other Algorithms

The SQUID algorithm is compared with the Ventura-Martinez and Long-Sun algorithms in Table 1 where m is the number of terms with nonzero amplitudes in the desired quantum superposition, n is the number of qubits in the superposition and p is the number of phase groups in the superposition. The complexity is measured in terms of two qubit gates.

Algorithm:	**Worst Case Performance:**	**Best Case Performance:**	**Total Number of Qubits**
Ventura-Martinez Algorithm	$O(mn)$	$O(mn)$	2n+1
Modified Ventura-Martinez Algorithm	$O(mn^2)$	$O(mn^2)$	$n+2$
Long-Sun Algorithm	$O(n^2 2^n)$	$O(n^2 2^n)$	n
SQUID Algorithm	$O(pn)$	$O(n)$	2n+2

m			
Modified SQUID Algorithm	$O(pn^2)$	$O(n)$	$n+2$

Table 1: Comparison of Quantum Superposition Initialization Algorithms

The modified versions of the Ventura-Martinez and SQUID algorithms refer to implementing the controlled gates using a quadratic blowup rather than by adding extra ancilla bits [6]. Because p is the number of phase groups, $p \leq m$ so the SQUID algorithm will never be worse than the Ventura-Martinez algorithm and the modified SQUID algorithm will never be worse than the modified Ventura-Martinez algorithm. In fact, the SQUID algorithm and modified SQUID algorithm will be faster than the Ventura-Martinez and modified Ventura-Martinez algorithms respectively whenever at least one phase group that covers more than one cell is found. This is true for most superpositions so the SQUID algorithm will be better for most superpositions. The best case performances of the SQUID and modified SQUID algorithms are much better than the best case performance of the Ventura-Martinez and modified Ventura-Martinez algorithms. The two versions of the SQUID algorithm also only use one additional ancilla bit to achieve this reduction in the number of gates required.

6. Conclusion

The SQUID algorithm discussed in this paper allows quantum superpositions to be initialized much more quickly than with existing algorithms for the vast majority of quantum superpositions. In fact, the quantum array generated by SQUID algorithm will be faster than the quantum array generated by any existing algorithm in all cases where at least group exists. The quantum array generated by the SQUID algorithm is also never slower than any existing algorithm and is an exponential improvement over all existing algorithms in the best case. This paper also introduces several new concepts such as the phase map, the new type of groups used in the phase map, the use of a new initialization operator to split groups off the generator state rather than single minterms as in previous papers and the use of controlled Hadamard gate for splitting groups into individual minterms in a linear amount of time.

7. References

1. Grover, A Fast Quantum Mechanical Algorithm for Database Search, *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ACM, 1996
2. Ventura and Martinez, Quantum Associative Memory,, *Information Sciences*, Elsevier, 1999
3. Ventura and Martinez, Initializing the Amplitude Distribution of a Quantum State, *Foundations of Physics Letters*, Plenum Publishing Corporation, 1999
4. Long and Sun, Efficient Scheme for Initializing a Quantum Register with an Arbitrary Superposed State, *Physical Review A*, The American Physical Society, 2001
5. Schulman and Vazirani, Molecular Scale Heat Engines and Scalable Quantum Computation, *STOC*, ACM, 1999
6. Barenco et. al., Elementary Gates for Quantum Computation, *Physical Review A*, The American Physical Society, 1995