Quantum Information and Computation, Vol. 23, No. 3&4 (2023) 0195–0222

QUANTUM ALGORITHM FOR EXACT MINIMAL ESOP MINIMIZATION OF INCOMPLETELY SPECIFIED BOOLEAN FUNCTIONS AND REVERSIBLE SYNTHESIS

HRITHIK KETINENI[a]
Department of Electrical and Computer Engineering, Portland State University
Portland, Oregon 97201, United States

MAREK PERKOWSKI[b]
Department of Electrical and Computer Engineering, Portland State University
Portland, Oregon 97201, United States

Received August 26, 2022
Revised January 6, 2023

The Exclusive-OR Sum-of-Product (ESOP) minimization problem has long been of interest to the research community because of its importance in classical logic design (including low-power design and design for test), reversible logic synthesis, and knowledge discovery, among other applications. However, no exact minimal minimization method has been presented for more than seven variables on arbitrary functions. This paper presents a novel quantum-classical hybrid algorithm for the exact minimal ESOP minimization of incompletely specified Boolean functions. This algorithm constructs oracles from sets of constraints and leverages the quantum speedup offered by Grover's algorithm to find solutions to these oracles, thereby improving over classical algorithms. Improved encoding of ESOP expressions results in substantially fewer decision variables compared to many existing algorithms for many classes of Boolean functions. This paper also extends the idea of exact minimal ESOP minimization to additionally minimize the cost of realizing an ESOP expression as a quantum circuit. To the extent of the authors' knowledge, such a method has never been published. This algorithm was tested on completely and incompletely specified Boolean functions via quantum simulation.

Keywords: ESOP, minimization, logic design, reversible synthesis, quantum, Grover's algorithm, Boolean functions, quantum cost

1 Introduction

A *reversible circuit* is a circuit with a time-reversible one-to-one mapping from the inputs to the outputs, meaning that the input states are always reconstructible from the output states. Reversible logic has applications in a wide variety of technologies, from optical, DNA, and quantum computing to nanotechnology and classical CMOS, but the prime importance of reversible logic resides in the inevitability that many future technologies will require reversible gates to reduce power [1].

This paper aims to develop a quantum algorithm for classical reversible synthesis based on the exact minimal minimization of general Exclusive-OR Sum-of-Product representations

[a]hrithik@pdx.edu
[b]h8mp@pdx.edu

of incompletely specified Boolean functions. The problem of searching for efficient designs of quantum circuits includes as its subproblem the synthesis of classical reversible gates [2]. That being the case, we pursue a secondary objective of reducing the quantum cost of reversible gates.

An *Exclusive-OR Sum-of-Product (ESOP)* expression is a representation of Boolean functions consisting of an Exclusive-OR (XOR) of cubes (products of literals). ESOP expressions are well-suited for many applications. They are fundamental in the design of quantum circuits because such circuits use XOR-based gates such as the Feynman and generalized Toffoli gates to realize an XOR-of-Products. Finding minimal ESOP expressions reduces the number of qubits and gates in a given quantum circuit [3]. These reductions are extremely important given the decoherence of current quantum computers. ESOP minimization methods also have broad applications in classical technology, including but not limited to machine learning, pattern theory [4], and the design of arithmetic, communication and testing circuits. ESOP is, along with SOP minimization, the most famous problem in classical logic synthesis. ESOP minimization is increasingly important because of its application in the design of quantum circuits and some nanotechnologies.

ESOP minimization translates to reversible synthesis [3] because an ESOP expression can be realized as a quantum circuit with negations and generalized Toffoli gates as shown in Figure 4. The cost of each generalized Toffoli gate depends on the number of inputs [5]. Finding ESOP representations of a smaller size naturally reduces the cost of realization, but the minimal total cost of the reversible circuit is dependent not only on the number of product terms of the ESOP expression but on other criteria as well such as the number of literals. This criterion differs from conventional logic optimization, which seeks only to minimize reversible circuits based on the cost of realization in classical technology.

There are two classes of ESOP minimization algorithms: *exact* and *heuristic*.

Many methods of heuristic minimization of ESOP expressions have been invented, most of which reduce ESOP expressions by repeatedly applying cube transformations, expanding, reshaping, and collapsing cubes [6, 7, 8]. Heuristic methods are fast, with cube transformations being particularly fast. Though fast, heuristic methods are not exact.

Most methods of finding exact minimal ESOP representations of Boolean functions are based on *Helliwell Decision functions*, both classically [9, 10] and in quantum [11]. However, the number of decision variables in these methods to describe an ESOP grows as $O(3^n)$ with the number of Boolean variables n, making such methods impractical for large problems on classical computers. Moreover, current quantum algorithms utilizing decision functions do not attempt to reduce the complexity of the number of variables [11, 12].

Riener et al. reduced the ESOP minimization problem to a satisfiability (SAT) problem, decreasing the number of decision variables [13]. Their method works by constructing SAT constraints that an ESOP representation of a specific size must satisfy to correctly represent a Boolean function, using a SAT-solver to solve the SAT problem, and iteratively decreasing the size and repeating until left with the solution of the smallest size. Unfortunately, SAT has been proven to be an NP-complete problem for which there is no polynomial time algorithm [14, 15]. Therefore, proving the unsatisfiability of a sufficiently large SAT problem in its present form at high values of n form is computationally expensive and cannot be done in reasonable time, making it impractical for high values of n as this is a crucial part of the

algorithm.

Numerous other classical methods have been proposed to minimize various kinds of AND-XOR expressions, none of which can find minimal ESOP representations of arbitrary functions for more than 7 variables [16, 17, 18, 19, 20, 21, 22]. Generally, in the worst case, all classical algorithms are, at best, only slightly better than an exhaustive algorithm.

A quantum algorithm called MOQMin for general ESOP minimization has been proposed by Sampson et al. [12] based on the three possible canonical expansions: *Shannon*, *positive Davio*, and *negative Davio*. The number of variables in the encoding that this algorithm uses to represent ESOPs grows as $O(2^n)$, and therefore it too is impractical for high values of n. This method is impractical for more than 7 variables.

This paper presents a quantum algorithm based on even-odd covering with a quadratic speedup over a hypothetical equivalent classical classical algorithm, which does not yet exist. The number of qubits required in our algorithm scales linearly due to the use of counters and an improved encoding [23]. Furthermore, it minimizes the entire quantum circuit (more than just the number of product terms of ESOP).

2 Exclusive-OR Sum-of-Products

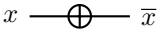
This section formalizes the basic concepts and notation regarding Exclusive-OR Sum-of-Products (ESOPs) used in this paper.

Definition 1: A Boolean function f is a mapping of every set of values of n Boolean variables to a Boolean value, taking the form $f : \{0,1\}^n \rightarrow \{0,1\}$. A Boolean function is *completely specified* if the corresponding Boolean value is known for every vector of values of all Boolean variables. It is *incompletely specified* if the corresponding value is not known for every vector of values of all Boolean variables. Every Boolean function of n Boolean variables can be represented by a propositional expression of n variables $x_1, x_2, ..., x_{n-1}, x_n$. Any two representations are logically equivalent if they represent the same Boolean function.

Definition 2: The NOT gate, also known as negation or complement, is a single-input Boolean function $f(x) = \overline{x}$ specified by the truth table shown in Figure 1a. Its quantum equivalent is the Pauli-X gate, depicted in Figure 1b.

x	$\overline{x}$
0	1
1	0

(a) Truth table of Boolean NOT.



(b) Pauli-X - the quantum equivalent of NOT.

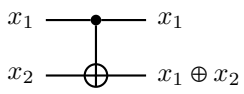
Fig. 1: The NOT gate and Pauli-X gate.

Definition 3: The XOR gate, or exclusive disjunction, is a Boolean function $f(x_1, x_2) = x_1 \oplus x_2$.

The XOR operation can be thought of as logical addition, excluding the case where both the operands are '1', as listed in Figure 2a, hence the name "Exclusive-OR." The XOR

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

(a) Truth table of Boolean XOR.



(b) The CNOT (Controlled Pauli-X), or Feynman gate - the reversible equivalent of XOR.

Fig. 2: The XOR gate and Controlled Pauli-X (CNOT) gate.

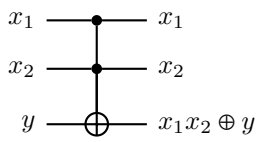
operation is denoted by the symbol $\oplus$, e.g., $x_1 \oplus x_2$. The XOR operation iteratively applied to a sequence of Boolean variables is denoted by $\bigoplus$, e.g., Eq. (1), in the same way that the uppercase letter sigma (Σ) is the operator for the summation of a sequence.

The CNOT gate, illustrated in Figure 2b, is the reversible equivalent of the classical AND gate. The black dot is known as the control, and the qubit it acts upon is known as the control qubit. The control qubit stays the same after the application of the CNOT gate, no matter what. The $\oplus$ symbol is known as the target, and the qubit it acts upon is known as the target qubit. If the control qubit is set to 1, the CNOT gate inverts the target qubit. If not, all qubits remain unchanged.

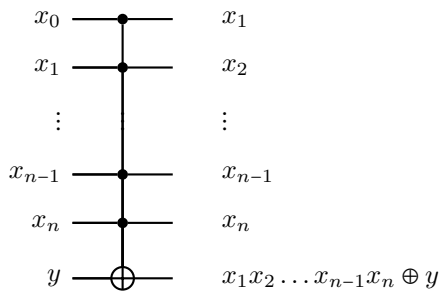
Definition 4: The AND gate, or conjunction, is a Boolean function $f(x_1, x_2) = x_1 x_2$.

x_1	x_2	$x_1 x_2$
0	0	0
0	1	0
1	0	0
1	1	1

(a) Truth table of Boolean AND.



(b) The Toffoli gate (Controlled-Controlled Pauli-X) - the reversible equivalent of AND.



(c) The generalized Toffoli gate - reversible equivalent of multi-input AND gate.

Fig. 3: The AND gate and Toffoli gate.

The AND operation is the Boolean equivalent of multiplication. As detailed by Figure 3a, if either of the inputs of the AND operations, x_1 and x_2, are equal to '0', the output is '0'. Only when both the inputs are '1' does the AND operations output a '1'. When the AND operator is explicitly shown, it is denoted by $\cdot$ or $\bigwedge$. More commonly, however, it is not shown but implied by the lack of space between two Boolean variables analogous to multiplication, e.g., $x_1 x_2$. The AND operation iteratively applied to a sequence of Boolean variables is denoted by $\bigwedge$, e.g., Eq. (1), in the same way that the uppercase letter pi (Π)

denotes the multiplication of a sequence.

The reversible equivalent of the AND gate is the *Toffoli* gate, shown in Figure 3b. The Toffoli gate has three inputs and three outputs. The black dots are known as controls, and the qubits they act upon are known as control qubits. The control qubits stay the same after the application of the Toffoli gate, no matter what. The $\oplus$ symbol is known as the target, and the qubit upon which it acts is known as the target qubit. If the control qubits are set to 1, the Toffoli gate inverts the target qubit. If not, all qubits are unchanged. When the value of the target qubit y is set to '0', its value after the application of the Toffoli gate is the conjunction of the values of the two control qubits. As Figure 3c indicates, a *generalized Toffoli* gate is an extension of the standard Toffoli gate with an arbitrary (more than two) number of inputs.

There are many ways to encode a product term (see Definition 7) in binary. One method uses two binary strings, one representing positive polarity and the other negative polarity[c][6]. For example, the product $x_2\overline{x_3}$ with a four variable support x_1, x_2, x_3, x_4 and its corresponding cube $-10-$ would be encoded in binary by $(0010, 0100)$ with the first string denoting the negative polarity and the second string denoting the positive polarity of each variable. The first bit of the cube **-**10− has neither a positive nor negative polarity, so the first bits of the positive and negative strings are both **0**. The second bit of the cube −**1**0− has a positive polarity, but not a negative polarity, so the positive string is 0**1** while the negative string is 0**0** up to the second bit. The third bit of the cube −1**0**− has a negative, not positive, polarity. Thus, the negative string is 00**1** while the positive string is 01**0** up to the third bit. The fourth and final bit of the cube −10**-** has neither a positive nor negative polarity, so the complete negative string is 0010 while the complete positive string is 0100.

Definition 5: An Exclusive-OR Sum-of-Product (ESOP) is an XOR of products (cubes). An ESOP of n Boolean variables $x_1, \ldots, x_n$ is expressed as

$$\bigoplus_{j=1}^{k} \left(\bigwedge_{i=1}^{n} x_i^{o_{i,j}} x_i^{p_{i,j}} \right), \tag{1}$$

where n is number of Boolean variables and k is the number of product terms. Each $o_{i,j} \in \mathbb{B}$, each $p_{i,j} \in \mathbb{B}$ and each expression

$$x_i^{o_{i,j}} = \begin{cases} 1, & \text{if } o_{i,j} = 0 \\ \overline{x_i}, & \text{if } o_{i,j} = 1 \end{cases} \tag{2}$$

$$x_i^{p_{i,j}} = \begin{cases} 1, & \text{if } p_{i,j} = 0 \\ x_i, & \text{if } p_{i,j} = 1 \end{cases} \tag{3}$$

for $1 \le i \le n$ and $1 \le j \le k$.

The value of $o_{i,j}$ denotes a bit of the negative polarity string of a cube within an ESOP expression. Correspondingly, the value $p_{i,j}$ denotes a bit of the positive polarity string. Note that a bit can never have both a negative and positive polarity, so $x_i^{o_{i,j}} = x_i^{p_{i,j}} \implies o_{i,j} = p_{i,j} = 0$. An ESOP expression can be represented using the set of all $o_{i,j}$ and $p_{i,j}$, numbering $2nk$ elements.[d]

[c] Variables of negative polarity are negated and variables of positive polarity are not.

[d] This notation has been adapted from [13] to define the method of encoding product terms described herein.

An ESOP can also be realized as a reversible circuit.

Example 1: Figure 4 shows a reversible circuit realizing the ESOP expression $x_1\overline{x_2}x_3 \oplus x_1x_2\overline{x_3} \oplus \overline{x_1}x_2$.

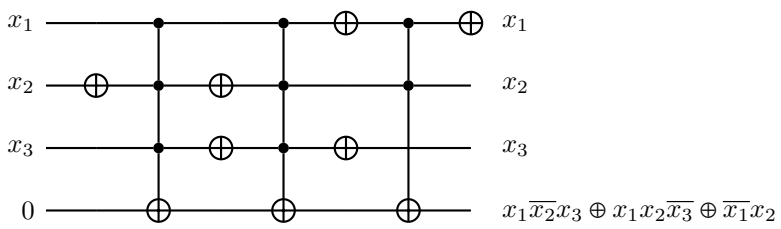


Fig. 4: Reversible circuit realization of ESOP expression.

Definition 6: A *literal* is a Boolean variable ($x_i^{o_{i,j}}$ or $x_i^{p_{i,j}}$) with a positive (x_i) or negative ($\overline{x_i}$) polarity.

Definition 7: A *cube* is a product term composed of literals multiplied together by the AND operator. A cube may also be called a product. For every j, the corresponding cube is given by the expression $\bigwedge_{i=1}^{n}\left(x_i^{o_{i,j}}x_i^{p_{i,j}}\right)$.

Definition 8: A *minterm* is a cube with a value of '0' or '1', in which each variable appears once. A minterm is called a *false minterm* if its value is '0' and a *true minterm* if its value is '1'. A minterm is said to be specified or a care minterm if it either true or false. Other minterms are said to be unspecified or a don't care. A Boolean function can be expressed as a list of true and false minterms, i.e., a truth table. A Boolean function can also be visualized by a Karnaugh map of minterms [24].

Example 2: If the Boolean function $f(x_1, x_2)$ can be represented by the ESOP expression $\overline{x_1}x_2 \oplus x_1\overline{x_2}$, the following equation is true for all specified minterms of f:

$$f(x_1, x_2) = \overline{x_1}x_2 \oplus x_1\overline{x_2} = \bigoplus_{j=1}^{2}\left(\bigwedge_{i=1}^{2} x_i^{o_{i,j}}x_i^{p_{i,j}}\right)$$

with $o_{1,1} = 1, o_{2,1} = 0, o_{1,2} = 0, o_{2,2} = 1$ and $p_{1,1} = 0, p_{2,1} = 1, p_{1,2} = 1, p_{2,2} = 0$.

Example 3: If the Boolean function $g(x_1, x_2, x_3)$ can be represented by the ESOP expression $\overline{x_1} \oplus x_2\overline{x_3}$, the following equation is true for all specified minterms of g:

$$g(x_1, x_2, x_3) = \overline{x_1} \oplus x_2\overline{x_3} = \bigoplus_{j=1}^{2}\left(\bigwedge_{i=1}^{3} x_i^{o_{i,j}}x_i^{p_{i,j}}\right)$$

with $o_{1,1} = 1, o_{2,1} = 0, o_{3,1} = 0, o_{1,2} = 0, o_{2,2} = 0, o_{3,2} = 1$ and $p_{1,1} = 0, p_{2,1} = 0, p_{3,1} = 0, p_{1,2} = 0, p_{2,2} = 1, p_{3,2} = 0$. The quantum circuit realizing this expression is shown in Figure 5.

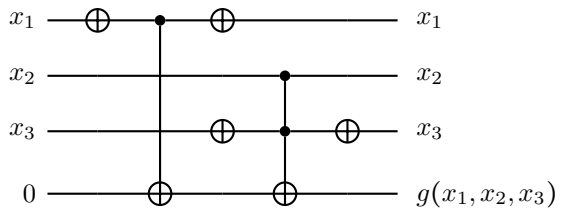


Fig. 5: Quantum circuit realization of $g(x_1, x_2, x_3)$.

Definition 9: A *Disjoint Sum-of-Product (DSOP)* expression is a specific form of an ESOP expression where all cubes are disjoint. Two cubes are said to be disjoint if they do not share a common minterm (the cubes do not overlap on the Karnaugh map), so their multiplication would result in a null value.

Definition 10: The ESOP minimization problem is to find the smallest set of cubes (products of literals) over the variables of a Boolean function such that each false minterm is covered by the cubes an even[e] number of times, and each true minterm of the Boolean function is covered an odd number of times. Unspecified minterms do not have such constraints and can be assigned an arbitrary value in the Karnaugh map of the solution expression.

Example 4: Taking the example of the function $f(x_1, x_2)$ shown in Figure 6a, let's find a Boolean expression for when an arbitrary product,

$$\bigwedge_{i=1}^{2} \left(x_i^{o_{i,1}} x_i^{p_{i,1}} \right),$$

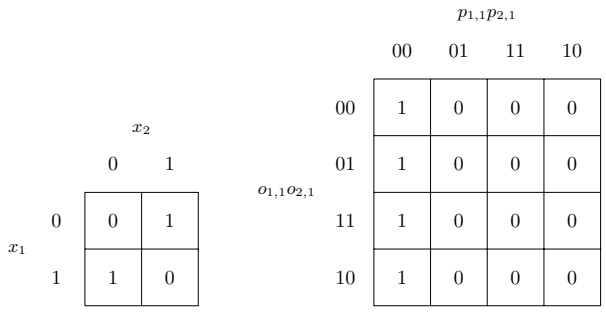
corresponding to the first product (j = 1) of an arbitrary ESOP representation covers the minterm 00. A Karnaugh map of this expression is shown in Figure 6b. The expression can be written as $\overline{p_{1,1}}\ \overline{p_{2,1}}$. The general methodology to create such expressions is to negate the "incorrect" polarity. For example, the minterm 10 is never covered by the cubes 0– or –1 or any of the cubes within them, i.e., 00, 01, or 11, so x_1 can never have a negative polarity ($o_{1,1} \neq 1$) and x_2 can never have a positive one ($p_{2,1} \neq 1$) as shown in Figure 7. Thus, the expression for the minterm 10 is $\overline{o_{1,1}}\ \overline{p_{2,1}}$.

From Definition 10, we know that each false minterm must be covered an even number of times, and each true minterm must be covered an odd number of times. The expression $\bigoplus_{j=1}^{k}(1)$ equals 0 if k is even and 1 if k is odd. We can use these general principles to construct constraints that a valid ESOP expression (given as a set of products) must satisfy for each minterm.

Theorem 1: For every true minterm $t_1 t_2 ... t_{n-1} t_n$ in the ON-set

$$\bigoplus_{j=1}^{k} \left(\left(\bigwedge_{i=1}^{n} \overline{t_i o_{i,j}} \right) \left(\bigwedge_{i=1}^{n} \overline{\overline{t_i} p_{i,j}} \right) \right) = 1. \tag{4}$$

[e] Zero is an even number!



(a) Karnaugh map of $f(x_1, x_2)$.

(b) Karnaugh map of expression for covering of the minterm 00.

Fig. 6: Karnaugh map for Boolean function $f(x_1, x_2)$ and a constraint.

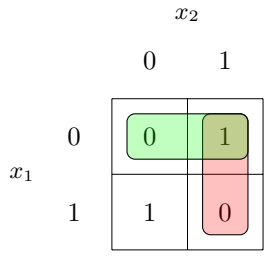


Fig. 7: Karnaugh map of $f(x_1, x_2)$ with the products not covering minterm 10.

Similarly, for every false minterm $f_1 f_2 ... f_{n-1} f_n$ in the OFF-set

$$\bigoplus_{j=1}^{k} \left(\left(\bigwedge_{i=1}^{n} \overline{f_i o_{i,j}} \right) \left(\bigwedge_{i=1}^{n} \overline{\overline{f_i} p_{i,j}} \right) \right) = 0. \tag{5}$$

Example 5: The completely specified Boolean function given in Figure 6a has four minterms: two false and two true. Let us construct constraints that an ESOP of size two[f] must satisfy to correctly represent this function. In total, we have four constraints, one for each minterm. If the function were incompletely specified, we would have one constraint per specified minterm. Starting with the false minterm 00, we have the constraint

$$\bigoplus_{j=1}^{2} \left(\left(\bigwedge_{i=1}^{2} \overline{0 o_{i,j}} \right) \left(\bigwedge_{i=1}^{2} \overline{\overline{0} p_{i,j}} \right) \right) = \bigoplus_{j=1}^{2} \left(\bigwedge_{i=1}^{2} \overline{p_{i,j}} \right) = \overline{p_{1,1}}\ \overline{p_{2,1}} \oplus \overline{p_{1,2}}\ \overline{p_{2,2}} = 0.$$

[f] Size is the number of cubes in an ESOP expression.

For the false minterm 11, we have the constraint

$$\bigoplus_{j=1}^{2}\left(\left(\bigwedge_{i=1}^{2}\overline{1 o_{i,j}}\right)\left(\bigwedge_{i=1}^{2}\overline{\overline{1} p_{i,j}}\right)\right)=\bigoplus_{j=1}^{2}\left(\bigwedge_{i=1}^{2}\overline{o_{i,j}}\right)=\overline{o_{1,1}}\ \overline{o_{2,1}}\oplus\overline{o_{1,2}}\ \overline{o_{2,2}}=0.$$

For the true minterm 01, we have the constraint

$$\bigoplus_{j=1}^{2}\left(\left(\bigwedge_{i=1}^{2}\overline{t_i o_{i,j}}\right)\left(\bigwedge_{i=1}^{2}\overline{\overline{t_i} p_{i,j}}\right)\right)=\bigoplus_{j=1}^{2}\left(\left(\overline{o_{2,j}}\right)\left(\overline{p_{1,j}}\right)\right)=\overline{o_{2,1}}\ \overline{p_{1,1}}\oplus\overline{o_{2,2}}\ \overline{p_{1,2}}=1.$$

For the true minterm 10, we have the constraint

$$\bigoplus_{j=1}^{2}\left(\left(\bigwedge_{i=1}^{2}\overline{t_i o_{i,j}}\right)\left(\bigwedge_{i=1}^{2}\overline{\overline{t_i} p_{i,j}}\right)\right)=\bigoplus_{j=1}^{2}\left(\left(\overline{o_{1,j}}\right)\left(\overline{p_{2,j}}\right)\right)=\overline{o_{1,1}}\ \overline{p_{2,1}}\oplus\overline{o_{1,2}}\ \overline{p_{2,2}}=1.$$

Finding an ESOP representation of this Boolean function is then reduced to finding the values of $o_{i,j}$ and $p_{i,j}$ such that the four constraints above are satisfied. This way, we reduce the problem of finding an ESOP solution of a given size to finding a solution to a set of constraints.

3 Grover's Algorithm

One method of solving a constraint satisfaction problem in classical computing is an *exhaustive search*. An exhaustive search is a simple algorithm that iterates through every state in a state space to solve a constraint satisfiability problem. There are often more efficient search methods, but sometimes an exhaustive search is the only applicable method.

Definition 11: A *constraint satisfaction problem (CSP)* is a problem characterized by a set of variables and a set of constraints upon different subsets of that set of variables. Each constraint describes which assignments of values to a subset of variables are allowed and which are not. All constraints must be satisfied.

Definition 12: A *state* is an assignment of values to all variables.

Definition 13: A *state space* is a set of all possible states.

Definition 14: A *solution* to a constraint satisfaction problem is a state that satisfies all given constraints. A CSP may have multiple such solutions.

Classical computing can perform exhaustive search in time complexity that grows linearly with the size of the state space, which itself usually grows exponentially with the size of the input. In quantum computing, however, there exists an algorithm that can perform such a search with square root the number of calls to the oracle. This algorithm is called *Grover's algorithm*. Grover's algorithm [25] is a well-known and versatile quantum search algorithm with a variety of applications, among them logic synthesis [26] [27].

Structure. Grover's algorithm is composed of a Hadamard transform on the input variables, several "Grover iterations," and measurement as shown in Figure 10. A *Grover iteration* is also known as an oracle-diffusion pair since it includes the oracle and diffusion as components.

A *quantum oracle* is a quantum circuit consisting of a reversible version of the constraints in the CSP. The oracle can be thought of as the reversible circuit realization of some classical function f that takes as input a state x and gives a single-bit output $f(x)$ that corresponds to whether or not the state satisfies the constraints of the problem [g] The qubit that acquires the value of $f(x)$ is known as the oracle's output or the target qubit of the oracle. The function $f(x)$ may equal 1 for more than one value of x. If the function f is not reversible, it can be converted to a reversible circuit with the addition of ancillary (ancilla) qubits, as seen in Figure 10.[h]

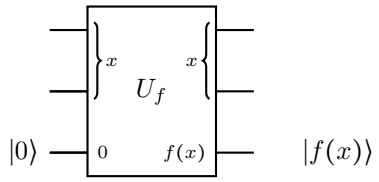


Fig. 8: Reversible realization of classical function f.

In the context of Grover's algorithm, the oracle U_f is responsible for flipping the phase of the solution states $|x\rangle$. This is achieved by initializing the output qubit to the $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ state. Phase kickback turns the classical, reversible circuit in Figure 8 into a Grover oracle shown in Figure 9. The result is that all solution states $|x\rangle$ are marked with a negative phase. Geometrically, this is equivalent to reflecting about an even superposition of non-solution states, which we call state $|s'\rangle$.

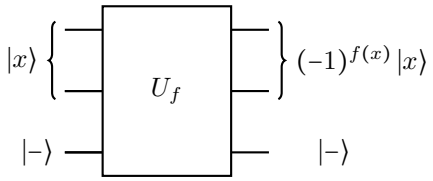


Fig. 9: The oracle in Grover's algorithm.

The *Grover diffusion* operator U_s is a quantum operation responsible for reflecting the solution states $|x\rangle$ about the superposition state $|s\rangle = H^{\otimes n}|0\rangle^n$, also called the "mean."

Grover iterations, or oracles and diffusions together, increase the probability amplitudes of the solution states while decreasing the amplitudes of all other states. This makes sense geometrically because starting at a uniform superposition of all states $|s\rangle$, the oracle reflects about the state $|s'\rangle$, and the diffusion reflects about the state $|s\rangle$, resulting in the transformation $U_s U_f$ rotating the initial state $|s\rangle$ closer to the solution states.

Grover iterations are repeated roughly $\sqrt{\frac{N}{M}}$ times where M is the number of correct solutions, and N is the number of all possible solutions in the general solution space[i] though

[g] The oracle has the same number of inputs as outputs given that is a reversible circuit, but in practice, after mirrors are applied, only one output is important.

[h] After the application of the oracle, the ancilla bits must return to their original values using mirrored circuits.

[i] $N = 2^n$ where n is the number of input variables in the oracle.

the optimal number of Grover iterations may not follow this trend for extreme values of N and M [28]; generally, the optimal number of iterations is $\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \rfloor$.

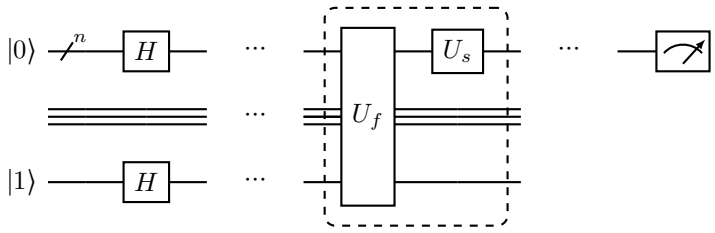


Fig. 10: Quantum circuit diagram for Grover's Algorithm.

Grover search is concluded by a measurement of the input qubits as shown in Figure 10. Grover's algorithm is probabilistic, and the probability of measuring a correct solution is usually high enough for this to be ignored for all practical purposes if all solutions are verified. For a given N and M, there exists an upper bound on the success probability of Grover's algorithm regardless of the number of iterations [28].

3.1 Components of the Oracle

A typical oracle of Grover's algorithm generally includes several different standardized components, which we call *quantum blocks.* Some common quantum blocks are the quantum counter, equality comparator, and less-than comparator.

Quantum Counter. The counter[j], shown in Figure 11, consists of a register of n qubits (called a *counting register*), which stores an integer in base two representation, together with an incrementer circuit that adds one to this stored integer when a control qubit is in the state $|1\rangle$ [23]. The counter allows us to represent an arbitrary number in a way that takes exponentially fewer qubits than having a single qubit for each possible number and to conditionally increment that number.

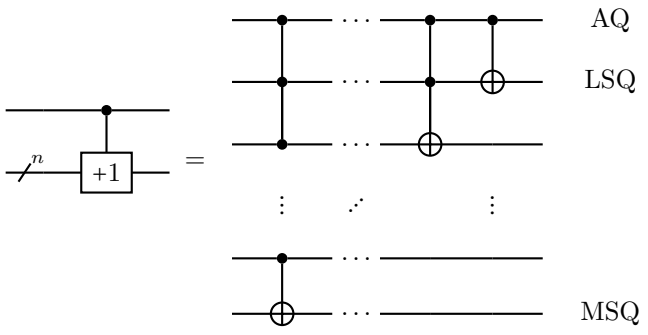


Fig. 11: Quantum circuit for a quantum counter.

The output of the counter is the stored integer in base two representation. The *most*

[j] Not to be confused with quantum counting as presented in [29].

significant qubit (MSQ) is at the bottom, and the *least significant qubit (LSQ)* is at the top of the register. We call the qubit above the least significant qubit, the control qubit, which hosts all the controls, the *activation qubit (AQ)*.

Equality Comparator. The equality comparator is essentially a series of Feynman gates connected from one string of bits to another string of bits [k]

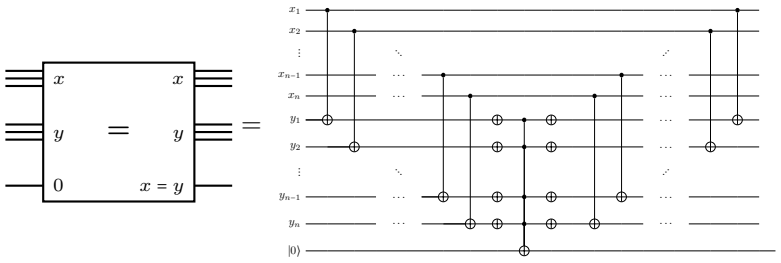


Fig. 12: Equality comparator as a quantum circuit with a mirror.

The Feynman gates produce a bit-by-bit XOR comparison, and the Toffoli gates with negation establish the condition that every two corresponding bits must be identical. Therefore, when x and y are the same, the output is 1.

Less-than Comparator. The name of the Less-than Comparator accurately describes its function. It takes two binary numbers as input and outputs a Boolean value for whether or not one number is less than the other.

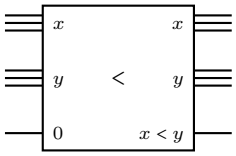


Fig. 13: Less-than comparator as a quantum circuit.

Using a standard iterative design with XOR logic, e.g., a standard heuristic minimizer, such as EXORCISM [6], a reversible circuit of the less-than comparator can be designed.

3.2 Grover's Algorithm for Optimization

A discrete optimization problem can be recast as a CSP by substituting the objective function for a constraint that sets the objective function equal to some high constant value. After finding a solution to this CSP, the value that the objective function must equal is iteratively decreased until the corresponding CSP can no longer be satisfied. Then, the solutions corresponding to the CSP with the smallest value of the objective function are said to minimize the objective function. Alternatively, starting at a low constant value, this value can be iteratively increased until the corresponding CSP is solvable. Then, the solutions to that solvable CSP minimize the objective function. The former method is called *downward search* while the latter is called *upward search*. This reformulation of an optimization problem as a series

[k]For brevity, we denote by x the binary string $x_1, x_2, ..., x_n$ and similarly by y the string $y_1, y_2, ..., y_n$.

of CSPs allows us to solve optimization problems using methods otherwise only pertinent to solve CSPs.

Definition 15: An *optimization problem (OP)* is characterized by a set of variables, a set of constraints, and an objective function. It is similar to a constraint satisfaction problem but differs in that it seeks not only to find a solution that satisfies constraints but to find the best of all possible solutions, one that minimizes the objective function. The *objective function* is a function that determines how good a solution is. For example, the objective function in the traveling salesman problem, represented by a graph with edges labeled with distance values, is the total cost of the edges in a solution.

Since Grover's algorithm can be used to solve CSPs, it follows that the repetition of Grover's algorithm can also be used to solve optimization problems. Using Grover's algorithm, an OP, or a brute-force sub-problem within an OP, can be solved with a quantum speedup.

4 Grover-based ESOP Minimization

ESOP minimization is an optimization problem. The set of variables is all variables $o_{i,j}$ and $p_{i,j}$ [l]. A specific assignment of values, i.e., a state, to these variables denotes a specific ESOP representation of a Boolean function. The state space is the set of all possible ESOP representations. The set of constraints is created by adopting the constraints detailed in Theorem 1 for each minterm. The solutions are the ESOP representations that satisfy these constraints. The objective function is the size, the number of product terms (cubes), of an ESOP representation.

Section 3.2 describes how an optimization problem can be converted to a series of constraint satisfiability problems. ESOP minimization is an optimization problem, and, as such, it can be converted to a series of CSPs. Each CSP, in this case, is to find an ESOP expression of a certain number, k, cubes, representing a Boolean function. Starting at some low value of k, k can be iteratively incremented until the corresponding CSP can be solved. The solution to that CSP is then the exact minimal ESOP solution.

4.1 Oracle Design for Exact Minimal ESOP Minimization

The space of all possible ESOP representations of size k for an n variable Boolean function is the set of every possible ordered set of k cubes. The *correct* ESOP representations of a Boolean function are those sets of cubes that satisfy Theorem 1. *Good* ESOP representations are sets of cubes of a certain acceptable size or lower size. *Exact minimal* representations are sets of cubes of the smallest possible size.

[l] Please do not confuse a *Boolean* variable, such as x_1 or $x_1^{p_{1,1}}$, of a Boolean function or its ESOP representations that we seek to find, with a variable of CSP or OP problem that we formulate, such as $o_{1,2}$, which corresponds to an input qubit of a Grover oracle. Also, note that many such terms bear different meanings in different contexts. This is because we attempt to formalize a quantum algorithm that itself minimizes quantum circuits, so when attempting to describe the quantum circuits, it may appear as if we are describing the quantum algorithm or vice versa. For example, we discuss quantum cost. This may refer to the gate cost of *our* algorithm, or it may refer to the gate cost of the quantum circuit (a quantum realization of an ESOP expression) we are trying to minimize.

Example 6: The space of all possible ESOP representations of size 2 for a 2 variable Boolean function is $\{\{1,1\},\{1,x_2\},\{1,\overline{x_2}\},\{1,x_1\},\{1,\overline{x_1}\},\{1,x_1x_2\},\{1,x_1\overline{x_2}\},\{1,\overline{x_1}\},\{1,\overline{x_1}x_2\},$ $\{1,\overline{x_1}\,\overline{x_2}\},\{x_2,1\},\{x_2,x_2\},\{x_1,\overline{x_2}\},\ldots\}$

As mentioned earlier, we can find the exact minimal ESOP representations of a Boolean function by constructing an algorithm that attempts to find all correct ESOP representations of a specific size[m] and then iteratively increasing this size until an ESOP representation of a given size exists. The algorithm would then yield these solutions. If a correct ESOP representation did not exist, the algorithm would not have produced correct solutions. This complete procedure is an upward search. Alternatively, we could use a downward search to achieve the same goal. We could find the exact minimal ESOP representations by constructing an algorithm to find correct ESOP representations of a large size and then iteratively decreasing this size until a smaller size cannot be found - essentially searching from the reverse direction. Both such algorithms can be implemented classically through a simple exhaustive search of every possible set of k cubes to find every combination of cubes that is a correct ESOP representation. Alternatively, an equivalent Grover-based method would yield a quadratic speedup over this hypothetical classical algorithm because Grover's algorithm could be used in place of the classical exhaustive search. The necessary oracles for Grover's algorithm would be made by translating the constraints from Theorem 1 to a quantum circuit.

Realization of constraints as a quantum circuit. The quantum circuit for the oracle of constraints can be constructed with Eq. (4) for every true minterm $t_1t_2...t_{n-1}t_n$ and Eq. (5) for every false minterm $f_1f_2...f_{n-1}f_n$. This can be done fairly simply because AND and XOR can be realized with the generalized Toffoli gate as shown in Figure 3c.

The circuit contains a qubit for all variables $o_{i,j}$ and $p_{i,j}$, qubits for two counting registers (one for false and one for true minterms), and an output qubit. For a given true minterm $t_1t_2...t_{n-1}t_n$, for every cube j, the expression

$$\left(\bigwedge_{i=1}^{n}\overline{t_i o_{i,j}}\right)\left(\bigwedge_{i=1}^{n}\overline{\overline{t_i}p_{i,j}}\right)$$

can be realized with a generalized Toffoli. Multiple generalized Toffoli gates together, with a target on the same qubit, can realize the expression

$$\bigoplus_{j=1}^{k}\left(\left(\bigwedge_{i=1}^{n}\overline{t_i o_{i,j}}\right)\left(\bigwedge_{i=1}^{n}\overline{\overline{t_i}p_{i,j}}\right)\right).$$

Example 7: Recall the two true constraints from Example 5. For the true minterm 01, we have the constraint

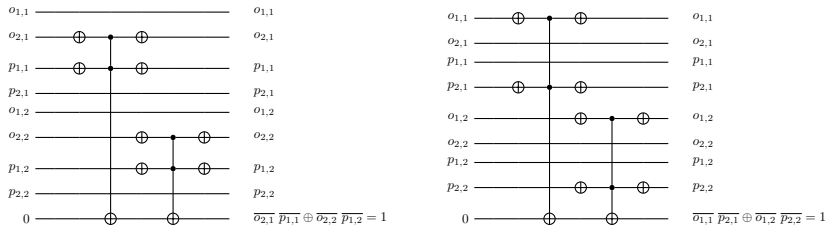
$$\bigoplus_{j=1}^{2}\left(\left(\overline{o_{2,j}}\right)\left(\overline{p_{1,j}}\right)\right) = \overline{o_{2,1}}\;\overline{p_{1,1}} \oplus \overline{o_{2,2}}\;\overline{p_{1,2}} = 1.$$

For the true minterm 10, we have the constraint

$$\bigoplus_{j=1}^{2}\left(\left(\overline{o_{1,j}}\right)\left(\overline{p_{2,j}}\right)\right) = \overline{o_{1,1}}\;\overline{p_{2,1}} \oplus \overline{o_{1,2}}\;\overline{p_{2,2}} = 1.$$

[m] A heuristic minimizer such as EXORCISM [6] can be used to find the initial starting size.

We can translate these constraints into a quantum circuit using a series of generalized Toffoli gates as shown in Figure 14.



(a) Constraint for true minterm 01. (b) Constraint for true minterm 10.

Fig. 14: Quantum circuit realizing an expression using generalized Toffoli gates.

Finding an ESOP representation is then reduced to finding the values of $o_{i,j}$ and $p_{i,j}$ such that the four constraints from Example 5 are satisfied.

These generalized Toffoli gates will have different combinations of controls as they correspond to different cubes, but their target will always be on the same qubit. Let us denote this set of Toffoli gates that correspond to a single true minterm by $\mathcal{T}$ as displayed in Figure 15.[n] The result is that the target qubit realizes Equation 4 of Theorem 1 for a single true minterm.

We can repeat this process for all true minterms, using counters to count every instance in which a true minterm satisfies Equation 4 and mirrors to reset all values besides the counter register before repeating. As shown in Figure 16, we do almost the same thing with false minterms. Instead, the expressions for individual generalized Toffoli gates are

$$\left(\bigwedge_{i=1}^{n} \overline{f_i o_{i,j}}\right)\left(\bigwedge_{i=1}^{n} \overline{\overline{f_i} p_{i,j}}\right).$$

The target qubit would then equal 1 when Equation 5 of Theorem 1 is *not* satisfied. To equate the target qubit to whether the equation is satisfied, we initially set the target qubit to the state $|1\rangle$. Again, we use counters to count every instance where a false minterm satisfies Equation 5, store this count in registers, and use mirrors to reset all values.

Suppose the *false register* contains r_f qubits, the *true register* contains r_t qubits, there are n_f false minterms, and there are n_t true minterms. We initialize the value of the false register qubits to $2^{r_f} - n_f$ and the true register qubits to $2^{r_t} - n_t$. This means that the final part of the oracle, a large generalized Toffoli gate with controls on all false register qubits and all true register qubits, is only activated if the counters on the false register are applied n_f times, and the counters on the true registers are applied n_t times.

[n]The circuit diagrams that follow do not include mirrors.

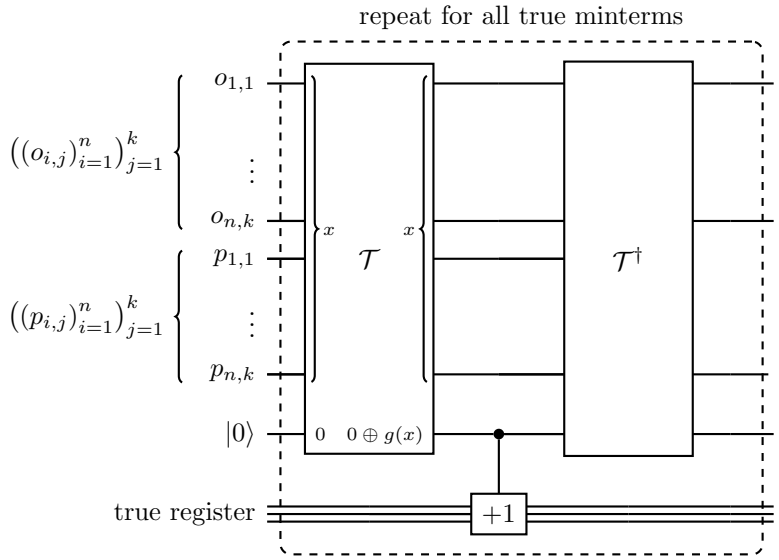


Fig. 15: Quantum circuit realizing constraints for true minterms.

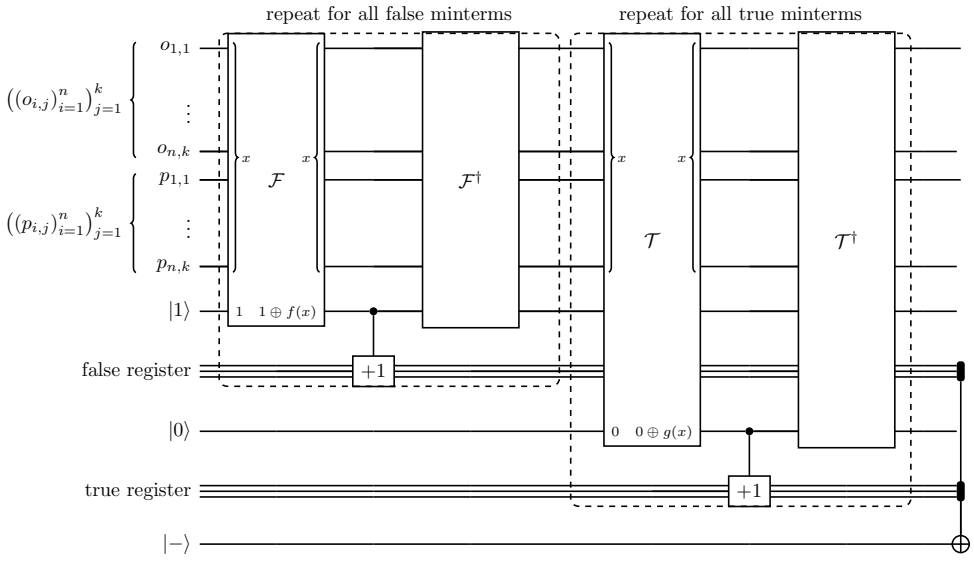


Fig. 16: General oracle for exact ESOP minimization problem specified by true and false minterms.

Example 8: The quantum circuit shown in Figure 18 is the oracle to find two product ESOP representations of the two-variable Boolean function specified by minterms in Karnaugh map from Figure 17. The input to this quantum circuit is the polarity-based encoding of product terms discussed in Section 2.

a \ b	0	1
0	0	1
1	1	-

Fig. 17: Karnaugh map of an incompletely specified two-variable Boolean function.

The blue area in Figure 18 realizes Equation 5 for the false minterm 00. Let us consider the term of the expression on left-hand side of the equation for when $j = 1$: $\left(\bigwedge_{i=1}^{n} \overline{f_i o_{i,1}}\right)\left(\bigwedge_{i=1}^{n} \overline{\overline{f_i} p_{i,1}}\right)$. Since $f_1 f_2$ is 00, we have $\left(\bigwedge_{i=1}^{n} \overline{f_i o_{i,1}}\right)\left(\bigwedge_{i=1}^{n} \overline{\overline{f_i} p_{i,1}}\right) = \overline{f_1 o_{1,1}} \cdot \overline{f_2 o_{2,1}} \cdot \overline{\overline{f_1} p_{1,1}} \cdot \overline{\overline{f_2} p_{2,1}} = \overline{0 o_{1,1}} \cdot \overline{0 o_{2,1}} \cdot \overline{\overline{0} p_{1,1}} \cdot \overline{\overline{0} p_{2,1}} = \overline{p_{1,1}} \cdot \overline{p_{2,1}}$. We realize this as a generalized Toffoli gate with negated controls on the qubits corresponding to $p_{1,1}$ and $p_{2,1}$ and a target on an ancilla bit. Similarly, $j = 2$ leads us to the expression $\overline{p_{1,2}} \cdot \overline{p_{2,2}}$ realized as a generalized Toffoli gate with negated controls on $p_{1,2}$ and $p_{2,2}$ and a target on the same ancilla bit. Applying the generalized Toffoli gates on the same target qubit effectively XORs the two expressions. We now have a single expression as a quantum circuit. The problem, however, is how to repeat this process in the most efficient way for all minterms. We have found that using the counters discussed in Section 2 is the most efficient method. The method works as follows: for every false minterm, we translate the Boolean constraints into a quantum form as we've just done, we add a counter, and mirror the constraints. We repeat this same process for true minterms though they have different expressions, and their counter registers and corresponding activations are on different qubits. This way, we translate the expression specifying all ESOP solutions into a quantum circuit.

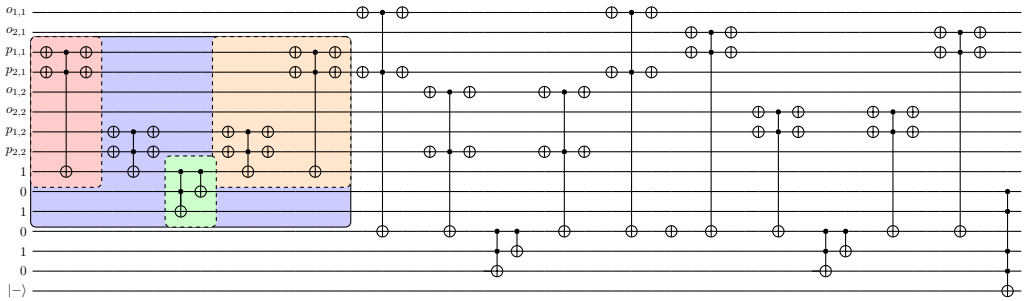


Fig. 18: Quantum oracle to find two product ESOP representations of the two-variable Boolean function from Figure 17.

4.2 Grover-based Exact ESOP Minimization procedure

The procedure for Grover-based exact ESOP minimization consists of iteratively constructing a quantum, reversible circuit, i.e., a Grover oracle that outputs true if and *only if* an

ESOP representation with k cubes exists. Grover's algorithm finds all possible solutions to this oracle. If a solution exists, Grover's algorithm will output an ESOP representation of k cubes. If no solutions exist, meaning that no ESOP representation of the Boolean function consisting of k product terms exists, k is increased, and this process is repeated. By systematically attempting to find solutions to the constraints realized by the reversible circuit for increasing values of k, an exact minimum ESOP representation (an ESOP of a minimal *size*) is guaranteed to be found.

Algorithm 1 Grover-based exact ESOP minimization (upward search).

Input: $f : \{0,1\}^n \to \{0,1,-\}, l \geq 1$
Output: $s = \left((o_{i,j})_{i=1}^{n}\right)_{j=1}^{k} \cup \left((p_{i,j})_{i=1}^{n}\right)_{j=1}^{k}$ for some value k

$s \leftarrow \varnothing$
function ORACLE(f, k) ▷ *Constructs oracle circuit*
    circuit $\leftarrow 2nk$ input qubits, a false register with r_f qubits, a true register with r_t qubits, 2 additional ancilla qubits, an output qubit
    for $m_1 m_2 ... m_{n-1} m_n \in \{0,1\}^n$ **do** ▷ *Iterates through all minterms $m_1 m_2 ... m_{n-1} m_n$*
        if minterm is true **then**
            realize true constraint $\bigoplus_{j=1}^{k} \left(\left(\bigwedge_{i=1}^{n} \overline{m_i o_{i,j}} \right) \left(\bigwedge_{i=1}^{n} \overline{\overline{m_i} p_{i,j}} \right) \right) = 1$ in circuit as $\mathcal{T}$
            realize counter with activation on output qubit of $\mathcal{T}$ to increment true register
            mirror $\mathcal{T}$
        else if minterm is false **then**
            realize false constraint $\bigoplus_{j=1}^{k} \left(\left(\bigwedge_{i=1}^{n} \overline{m_i o_{i,j}} \right) \left(\bigwedge_{i=1}^{n} \overline{\overline{m_i} p_{i,j}} \right) \right) = 0$ in circuit as $\mathcal{F}$
            realize counter with activation on output qubit of $\mathcal{F}$ to increment false register
            mirror $\mathcal{F}$
    return circuit
function GROVER(circuit) ▷ *Finds solutions to the oracle using Grover's algorithm*
    assemble full Grover's algorithm circuit
    run Grover's algorithm until all solutions are found
    if *correct* solutions exist **then**
        return solutions
    else
        return $\varnothing$
while $s = \varnothing$ **do**
    $s \leftarrow$ GROVER(ORACLE(f, l))
    $l \leftarrow l + 1$
output s

Once the minimal size is known, the oracle with additional constraints is used to optimize for the quantum cost. It is observed that for large, completely specified functions, the ESOP with the minimal quantum cost is also the one with the minimal size since smaller size necessitates larger ESOP terms (terms with a lower number of variables).

Algorithm 1 describes the exact minimization procedure using upward search. The inputs to it are a Boolean function f as a truth table of the true and false minterms along with a lower bound l for the minimal size of an ESOP representation. Methods exist to find this value [30]. The function ORACLE constructs a bit-flip oracle with $2nk$ input qubits, each for

every $o_{i,j}$ and $p_{i,j}$, that is satisfied when an ESOP representation of k products correctly represents f. The function GROVER finds and verifies all solutions to the CSP, specified as a quantum circuit, such that if the input qubits of the oracle are set to the solution state, the output qubit evaluates to 1. Grover's algorithm is used to compute these solutions. In order to decide unsatisfiability with sufficiently high probability, Grover's algorithm is repeated some number of times. If the solution produced by Grover is unsatisfiable, this process is repeated with the value k incremented.

4.3 Additional Constraints for Reversible Synthesis

The fundamental ESOP minimization procedure detailed in Section 4.2 finds the ESOP representation of a Boolean function with a minimal number of *cubes*. That in and of itself is significant progress in reducing the cost of gates in quantum circuits because generally less number of cubes leads to cubes with fewer literals which in turn decreases the quantum cost since literals correspond to controls of a generalized Toffoli gate. However, if we are to achieve a true exact reduction in the gate cost of quantum circuits, we must implement additional constraints to find ESOP solutions with the lowest quantum cost. This will not be strictly necessary for most ESOP representations to be realized using reversible logic. The need arises when there are a great many exact minimal ESOP representations of a Boolean function with a wide range of quantum costs, as in the case of highly incompletely specified functions, and we seek to find the ones least costly to realize as a reversible circuit.

The commonly accepted measure of quantum gate cost of generalized Toffoli gates is the one detailed by Maslov and Dueck in their seminal paper *Improved Quantum Cost for n-bit Toffoli Gates* [5]. The oracle can account for this cost by adding constraints for how many inputs the Toffoli gates that realize a given ESOP representation can have in total (the sum of the number of inputs of each *individual* Toffoli gate). These constraints can be implemented by having a series of counters on the qubits corresponding to the input variables of the oracle and a less-than comparator on the counter's register. Figure 19 illustrates the oracle with additional constraints[o]. Maslov and Dueck showed that the cost of a single generalized Toffoli gate increases linearly with the number of inputs if one garbage qubit is allocated per Toffoli gate. Thus, minimizing the sum of the total number of inputs of all generalized Toffoli gates minimizes the quantum cost of an ESOP representation, under the assumption that garbage qubits are used in the circuit, realizing the ESOP expression proportional to the size of the ESOP.

After finding the minimal size of an ESOP representation using Algorithm 1, the oracle with additional constraints can be used to find an ESOP representation with a corresponding quantum circuit that has a minimal total number of inputs, i.e., Toffoli controls. First, we construct an oracle with additional constraints that outputs true when an ESOP representation is of the minimal size and is lower than some maximum cost. Next, we use Grover's algorithm to find all solutions to the oracle. We then iteratively reduce this maximum cost and again use Grover's algorithm to find solutions until a solution with a lower cost cannot be found. We then repeat this reversible cost minimization process, but with oracles that take greater numbers of products as input. We stop at some high upper bound for the number of

[o]It is noteworthy that the way these additional constraints are presently realized in the oracle is relatively inefficient. It could be simplified much by using the method of reversible synthesis. This idea is interesting because this algorithm can potentially minimize itself and make itself more efficient.

products. There are several ways to compute this bound. Theorem 2 posits that the quantum cost of a minimal DSOP representation can be used as this upper bound. The ESOP solutions obtained via this process with the lowest total number of corresponding Toffoli controls are said to minimize the cost of a reversible circuit with a specific number of garbage bits representing an ESOP representation.

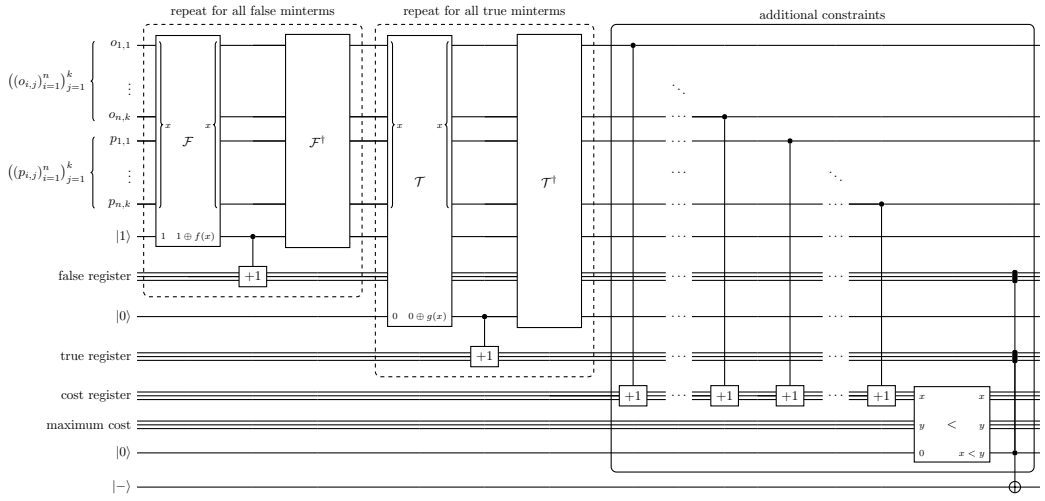


Fig. 19: Full oracle for finding exact minimal ESOP representations.

A slightly more sophisticated process is required to minimize the quantum cost of a Toffoli network[p], lacking garbage qubits, that realizes an ESOP expression representing a Boolean function under some reasonable assumptions.

Theorem 2: The quantum cost of the minimal DSOP representation of a Boolean function is greater than or equal to the quantum cost of the ESOP representation with the lowest quantum cost, regardless of whether garbage qubits are allowed. This is because DSOP expressions are ESOP expressions constrained by the fact that their products must be disjoint, which means that their products must be smaller and, thus, of a higher cost.

This process is the same as the one detailed above, that assumes garbage qubits, except that a classical computer must calculate the gate cost of the quantum circuit without garbage qubits for each ESOP expression found by Grover's algorithm and the starting value of the maximum cost must be higher. According to Maslov and Dueck, the quantum cost of a generalized Toffoli gate without garbage qubits grows exponentially with the number of inputs. Therefore, the ESOP expressions with the lowest calculated cost of realization without garbage qubits are the ones that minimize a reversible circuit representing an ESOP representation lacking garbage qubits.

[p] A Toffoli network is a quantum circuit composed of input qubits and a series of Toffoli gates, each one controlled by some or all of these inputs qubits with a target on the same output qubit as depicted by Figure 4.

4.4 Variants

There are several classes of AND-XOR expressions besides ESOP expressions. Variants of our presented algorithm can be applied to almost all of these different expressions.

Positive polarity Reed-Muller expressions. The algorithm would work particularly efficiently on Positive Polarity Reed-Muller expressions (PPRM). Every literal in a term of a PPRM expression is restricted to a single polarity, i.e., positive polarity, because $o_{i,j} = 0$ for all variables. Hence, a Boolean variable of a PPRM expression may either have positive or no polarity. In the current version of our algorithm, two qubits are required to describe the polarity or lack thereof of a literal (one for $o_{i,j}$ and another for $p_{i,j}$). In an equivalent algorithm for PPRM expressions, only a single qubit is required (for $p_{i,j}$). This would cut in half the number of input qubits needed in the oracle, exponentially reducing the gate cost since the cost of a generalized Toffoli gate grows exponentially without garbage bits [5].

Negative polarity Reed-Muller expressions. An equivalent increase in efficiency occurs for a variant of this algorithm to minimize Negative Polarity Reed-Muller (NPRM) expressions since the corresponding oracle circuit is almost identical to that of the PPRM, except the input qubits would instead represent negative polarity ($o_{i,j}$).

Fixed polarity Reed-Muller expressions. The variant of our algorithm for fixed polarity Reed-Muller expressions (FPRM) is greatly simplified to the same extent as the variants for PPRM and NPRM expressions. Since every term has a fixed polarity, each variable in the term is either of positive polarity or negative polarity. A single qubit can be used to represent each variable though an additional qubit is required for each term, which again reduces the number of variables in the oracle by a factor of 2 and brings about a similar decrease in the gate cost of the oracle.

5 Comparison with Existing Algorithms for Exact Minimal ESOP Minimization

Our algorithm is the most practical and scalable of the presently existing quantum algorithms for ESOP minimization of incompletely specified functions because its oracle circuit can be constructed without significant overhead for arbitrary functions, owing to its use of the minimum number of qubits and gates required to solve a specific ESOP minimization problem in contrast to other algorithms with largely fixed oracles. Further, variants of this algorithm are optimized for different classes of AND-XOR expressions.

5.1 Comparison with Earlier Quantum Algorithms

There are two approaches in the literature for minimizing ESOP using quantum methods. The first method is the use of a generating function [11]. The generating function method searches through 3^n generating functions. The second method is the MOQMin algorithm [12]. Given an incompletely specified Boolean function, MOQMin searches through all compatible[q] completely specified Boolean subfunctions of size $n - 1$. By contrast, our algorithm directly searches through the possible ESOP representations of a Boolean function. The efficiency

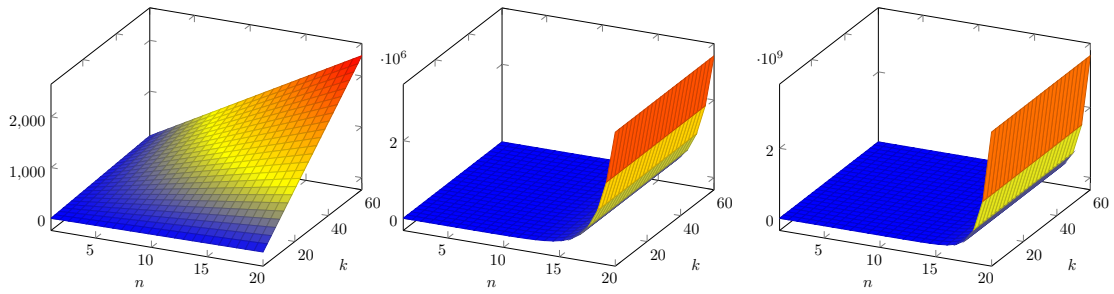
[q] Two functions are compatible if there is no conflict in the specified minterms.

gained from searching through this usually smaller search space translates to a lower number of qubits and gate cost.

The number of qubits. The generating function method requires 3^n decision variables to describe the selection of different combinations of generating functions. Our method uses exponentially fewer decision variables in the worst-case as n increases when $n > 5$. In addition to this, the generating function method also requires m ancilla bits, where m is the number of specified minterms, so such methods require roughly $3^n + m$ qubits in total. This means that the number of qubits required for the variables of the quantum algorithm based on generating functions scales exponentially with the number of variables n regardless of the proportion of minterms that are specified.

The MOQMin method uses a minterm representation of three subfunctions and a lookup table to calculate the weights of each subfunction. This requires $3 \cdot 2^{n-1}$ input qubits along with a similar number of ancilla qubits despite the fact that many minterms may not be specified. For each additional unspecified minterm, the search space of MOQMin is doubled; by contrast, the search space of our method only decreases because more greatly unspecified functions have smaller ESOP representations.

The proposed method in our paper provides a way to minimize Boolean functions with a substantially reduced number of qubits, allowing us to minimize considerably larger Boolean functions. Our method requires at most $2nk$ input qubits, where k is the number of products of the minimal ESOP representation, in addition to a small number of ancilla qubits that scale logarithmically with the number of specified minterms. For many types of Boolean functions, including but not limited to parity functions [31], adders [20], and highly incompletely specified functions, the number of products k has been shown to grow polynomially with n in the worst case, sometimes even linearly. Generally, as the number of unspecified minterms increases, the greater the benefit of our algorithm versus the alternatives. This means that the total number of qubits that our method requires grows polynomially in many practical cases versus the exponential growth of alternative quantum algorithms in all cases as shown in Figure 20.



(a) Our method. (b) MOQMin. (c) Generating Function.

Fig. 20: Number of input qubits required by different quantum methods for certain classes of Boolean functions.

Gate cost. An upper bound for the Maslov cost of our unmirrored oracle circuit, as a function M of m_f false minterms, m_t true minterms, n variables, and k terms, is presented in Equation 6.

$$
\begin{aligned}
M(m_f, m_t, k, n) = 2m_f\Bigg(&\underbrace{2k(2^{(n+1)}-3)+1}_{\text{false constraints}} + \underbrace{\sum_{c=2}^{\lceil \log_2 m_f \rceil}(2^{(c+1)}-3)}_{\text{counters of false constraints}}\Bigg) \\
+2m_t\Bigg(&\underbrace{2k(2^{(n+1)}-3)+1}_{\text{true constraints}} + \underbrace{\sum_{c=2}^{\lceil \log_2 m_t \rceil}(2^{(c+1)}-3)}_{\text{counters of true constraints}}\Bigg) \\
+&\underbrace{2^{(\lceil \log_2 m_f \rceil + \lceil \log_2 m_t \rceil + 1)} - 3}_{\text{big Toffoli gate}}
\end{aligned}
\tag{6}
$$

MOQMin utilizes lookup table circuits to assess the weight of each subfunction represented in its minterm representation. Since classical computers must calculate these lookup tables exhaustively, this method is impractical beyond 5 variables functions [7]. In the version of MOQMin for ESOP, each lookup table is constructed recursively from smaller lookup tables. Because the lookup table circuits must encode the weights of all possible subfunctions of $n-1$ variables, each represented by 2^{n-1} minterms, the MOQMin-Oracle has a higher gate cost than our method for all practical purposes.

The generating function method has a cost of approximately

$$
M(n) = \sum_{i=0}^{n}\left(2i\left(\binom{n}{i}2^i + 1\right)\right) + 2^n + \underbrace{2^{2^n+1}}_{\text{big Toffoli gate}} \tag{7}
$$

given n variables and assuming a completely specified function. The gate cost varies depending on which specific minterms are specified. This method has a final, big Toffoli gate that performs a Boolean AND of all constraints. In general, this Toffoli gate has $m = m_f + m_t$ inputs. For moderate and even small values of m with respect to n, this gate has an extremely high cost because each additional minterm doubles the cost of the Toffoli gate. For a completely specified function, the gate cost of the big Toffoli gate would be 2^{2^n+1}.

Given a Boolean function of 9 variables, 10 products, and 88% incompletely specified minterms, our algorithm requires 192 qubits in total whereas the generating function method requires 19744 qubits. MOQMin requires 768 input qubits and at least the same number of ancilla qubits. The circuit realizing MOQMin would need to encode lookup tables accounting for approximately 10^{77} combinations. This circuit would be so incredibly large that decoherence would be inevitable. At the time of this writing, IBM Osprey is the largest quantum computer in the world with 433 qubits. Our algorithm is therefore the only one able to minimize this Boolean function on a quantum computer.

5.2 Comparison with Classical Algorithms

The only comparable classical method for exact minimal ESOP minimization of large Boolean functions is software-based [13], whereas our proposed method is quantum-hardware-based. Hence, an exact comparison of the performance is not feasible.

This classical method relies on a SAT solver to solve a series of CSPs with km more decision variables. The Strong Exponential Time Hypothesis (SETH) [32] tells us that the worst case

time complexity of this process is roughly $O(2^{2nk+km})$. This means that the classical method has an exponential worst-case.

Grover's algorithm outperforms all classical algorithms for unstructured search and therefore has a better worst case complexity [25]. Though some NP-complete problems contain structure that can be exploited to create more efficient algorithms, for k-SAT, Grover's algorithm outperforms the best known classical algorithm [33]. Therefore, a quantum version of this classical method for ESOP minimization would be faster than the original classical method and would have a run time of $O(2^{nk+\frac{km}{2}})$. Our proposed algorithm is even faster than the direct quantum translation of the classical method described in [13] because it has km less decision variables, meaning less evaluations of the oracle. Specifically, our algorithm has a run time of $O(2^{nk})$. Thus, our algorithm is superior to the only comparable classical method.

Additionally, in order to guarantee an exact solution in an upward search, the classical method must repeatedly prove unsatisfiability. Proving the unsatisfiability of classical SAT is exponential in the worst case. Accordingly, if the classical method relies on upward search, it would be computationally similar to a repeated exhaustive search. The only alternative to an upward search is a downward search. Downward search does not necessitate repeatedly proving unsatisfiability and has the potential to be faster if the starting k is close to the minimum possible k. However, the downward search version of the classical algorithm [13] was even slower because, on average, it analyzed many more cases before reaching unsatisfiability than upward search.

Our proposed algorithm uses Grover's algorithm to produce a quadratic speedup and to identify unsatisfiability with near certainty. Moreover, it achieves the secondary benefit of being able to produce many solutions at a time when run multiple times.[r]

6 Experiments

Our algorithm was simulated in Qiskit [34], an open-source software development kit (SDK) for quantum computing, on many completely and incompletely functions involving a small number of variables n.

Example 9: The function that the Karnaugh Map displayed in Figure 6a represents has four minterms. Example 5 constructed four constraints, one for each minterm, that an ESOP of size two must satisfy to correctly represent this Boolean function.

$$(1)\ \overline{p_{1,1}}\ \overline{p_{2,1}} \oplus \overline{p_{1,2}}\ \overline{p_{2,2}} = 0 \qquad (2)\ \overline{o_{1,1}}\ \overline{o_{2,1}} \oplus \overline{o_{1,2}}\ \overline{o_{2,2}} = 0$$
$$(3)\ \overline{o_{2,1}}\ \overline{p_{1,1}} \oplus \overline{o_{2,2}}\ \overline{p_{1,2}} = 1 \qquad (4)\ \overline{o_{1,1}}\ \overline{p_{2,1}} \oplus \overline{o_{1,2}}\ \overline{p_{2,2}} = 1$$

These constraints are specified in the form of an oracle, shown in Figure 21.

Grover's algorithm is used to find solutions or assignment of values to the input qubits of this oracle such that the output bit is flipped. For this specific example, we use five Grover iterations, which comes from the formula $\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \rfloor$ given in Section 3. Qiskit allows us to get histogram data for an experiment, a set of bit-strings representing measurement values of a

[r] In fact, our algorithm is able to produce *all* correct solutions if run a sufficient number of times and if each time the oracle is modified to exclude all previous solutions.

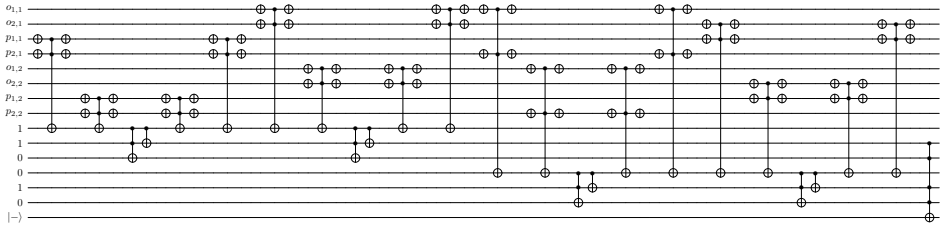


Fig. 21: Oracle specification of constraints of Example 5 (mirrors not shown).

set of qubits and their corresponding probabilities. This means that we can see all solutions to this oracle Grover's algorithm produces and the probability of each of them.

The function shown in Figure 6a has multiple ESOP representations of size two: $\overline{x_1}x_2 \oplus x_1\overline{x_2}, \overline{x_1} \oplus \overline{x_2}$, and $x_1 \oplus x_2$.

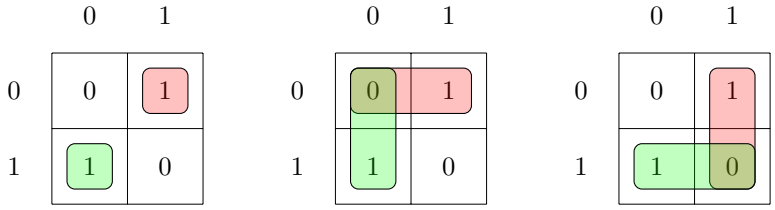


Fig. 22: Minimal ESOP representations of the function shown in example.

Each of these representations can be written in two different ways depending on the order of the products, e.g., $x_1 \oplus x_2$ vs $x_2 \oplus x_1$. Because the way in which we encode ESOP expressions means that order matters, there are always two different bit-strings representing the same ESOP expression! For example, the solution $x_1 \oplus x_2$ can be encoded $o_{1,1} = 0$, $p_{1,1} = 1$, $o_{2,1} = 0$, $p_{2,1} = 0$, $o_{1,2} = 0$, $p_{1,2} = 0$, $o_{2,2} = 0$, $p_{2,2} = 1$, meaning that x_1 is the first product and x_2 is second, or $o_{1,1} = 0$, $p_{1,1} = 0$, $o_{2,1} = 0$, $p_{2,1} = 1$, $o_{1,2} = 0$, $p_{1,2} = 1$, $o_{2,2} = 0$, $p_{2,2} = 0$ denoting the opposite order. The corresponding bit-strings are 00100001 and 00010010. Hence, since there are three unique ESOP solutions, we expect the Qiskit simulation of Grover's algorithm to produce six measurement values of equal probability, as shown in Figure 23.

In this way, we simulated and verified the results of Grover's algorithm with oracles representing many different Boolean functions and ESOP constraints. However, because of the limits of computer simulation and hardware constraints, we could only use Grover's algorithm to find solutions to oracles for the ESOP minimization of two to four variable Boolean functions of limited complexity.

7 Conclusion

ESOP minimization is one of the most challenging problems in logic synthesis. The pre-

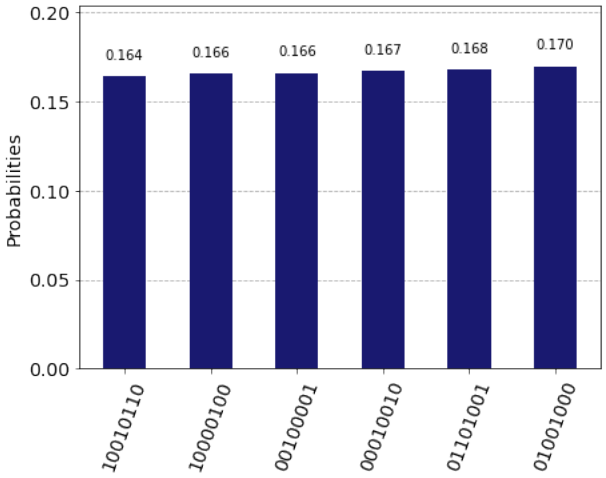


Fig. 23: Result of the algorithm on two-qubit function (order of measurement states are flipped).

sented work in this paper introduces a new hybrid quantum-classical algorithm for the exact minimal ESOP minimization of incompletely specified Boolean functions and, by extension, for reversible synthesis. By using a classical computer to construct an oracle for finding ESOP representations of a specific size, applying Grover's Algorithm to find all possible solutions, and then iteratively repeating this process with an adjusted size, an exact minimum ESOP representation of a given Boolean function is found.

Existing quantum algorithms for ESOP minimization, MOQMin, and the generating function method, use a substantially greater number of qubits on many classes of Boolean functions. The method presented in this paper reduces the number of input qubits by using an improved encoding of ESOP expressions. This makes our method highly scalable for sizeable reversible synthesis and machine learning classification tasks, among other applications, as quantum hardware advances. This algorithm was tested on small, completely, and incompletely specified Boolean functions via quantum simulation. It offers a quadratic speedup over a hypothetical equivalent classical algorithm and is by many metrics, especially by usability for large applications, superior to existing classical algorithms.

We also show how by using additional constraints, the Maslov cost of the quantum circuit realization of an ESOP expression can also be minimized. To the best of the authors' knowledge, this is the first method, whether classical or quantum in nature, that attempts to do this. Many variations of this algorithm were discussed for the exact minimization of different types of AND-XOR expressions.

References

1. Charles H Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973.
2. Robert Wille, Mathias Soeken, Christian Otterstedt, and Rolf Drechsler. Improving the mapping of reversible circuits to quantum circuits using multiple target lines. In *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 145–150. IEEE, 2013.
3. Giulia Meuli, Bruno Schmitt, Rüdiger Ehlers, Heinz Riener, and Giovanni De Micheli. Evaluating esop optimization methods in quantum compilation flows. In *International Conference on Reversible Computation*, pages 191–206. Springer, 2019.
4. Marek Perkowski, Tim Ross, Dave Gadd, Jeffrey A Goldman, and Ning Song. Application of esop minimization in machine learning and knowledge discovery. In *Proc. Reed Muller*, volume 95. Citeseer, 1995.
5. D. Maslov and G.W. Dueck. Improved quantum cost for n-bit toffoli gates. *Electronics Letters*, 39(25):1790, 2003.
6. Alan Mishchenko and Marek Perkowski. Fast heuristic minimization of exclusive-sums-of-products. *5th International Reed-Muller Workshop*, pages 242–250, 2001.
7. Norio Koda and Tsutomu Sasao. Lp characteristic vector for logic functions. In *Proc. of IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 99–108. Citeseer, 1993.
8. Martin Helliwell and Marek Perkowski. A fast algorithm to minimize multi-output mixed-polarity generalized reed-muller forms. In *Proceedings of the 25th ACM/IEEE Design automation conference*, pages 427–432, 1988.
9. Marek Perkowski and Malgorzata Chrzanowska-Jeske. An exact algorithm to minimize mixed-radix exclusive sums of products for incompletely specified boolean functions. In *1990 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1652–1655. IEEE, 1990.
10. Bernd Steinbach, Vladimir Yanchurkin, and Martin Lukac. On snf optimization: a functional comparison of methods. In *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technology (RM 2003)*, pages 11–18. Citeseer, 2003.
11. Peter Jin and Marek Perkowski. Designing variants of efficient satisfiability and max-satisfiability oracles for grover's algorithm with many practical applications. Internal report of Portland State University, Department of Electrical and Computer Engineering, 2021.
12. Marinos Sampson, Dimitrios Voudouris, and George K Papakonstantinou. A quantum algorithm for finding minimum exclusive-or expressions for multi-output incompletely specified boolean functions. In *CDES*, pages 105–111. Citeseer, 2008.
13. Heinz Riener, Rüdiger Ehlers, Bruno de O Schmitt, and Giovanni De Micheli. Exact synthesis of esop forms. In *Advanced Boolean Techniques*, pages 177–194. Springer, 2020.
14. Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.
15. L Levin. Universal search problems. *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.
16. A Gaidukov. Algorihm to derive minimum esop for 6-variable function. In *5th International Workshop on Boolean Problems, Sept. 2002*, 2002.
17. Stergios Stergiou and George Papakonstantinou. Exact minimization of esop expressions with less than eight product terms. *Journal of Circuits, Systems, and Computers*, 13(01):1–15, 2004.
18. George Papakonstantinou. Exclusive or sum of complex terms expressions minimization. *Integration*, 56:44–52, 2017.
19. Tsutomu Sasao. And-exor expressions and their optimization. In *Logic Synthesis and Optimization*, pages 287–312. Springer, 1993.
20. Tsutomu Sasao. Exmin2: a simplification algorithm for exclusive-or-sum-of-products expressions for multiple-valued-input two-valued-output functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(5):621–632, 1993.
21. Marinos Sampson, Marios Kalathas, Dimitrios Voudouris, and G Papakonstantinou. Exact esop

expressions for incompletely specified functions. *Integration*, 45(2):197–204, 2012.
22. Rolf Drechsler, Alexander Finder, and Robert Wille. Improving esop-based synthesis of reversible logic using evolutionary algorithms. In *European Conference on the Applications of Evolutionary Computation*, pages 151–161. Springer, 2011.
23. Edison Tsai and Marek Perkowski. A quantum algorithm for automata encoding. *Facta Universitatis, Series: Electronics and Energetics*, 33(2):169–215, 2020.
24. Maurice Karnaugh. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 72(5):593–599, 1953.
25. Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
26. Marek Perkowski. Inverse problems, constraint satisfaction, reversible logic, invertible logic and grover quantum oracles for practical problems. *Science of Computer Programming*, 218:102775, 2022.
27. Yiwei Li, Edison Tsai, Marek Perkowski, and Xiaoyu Song. Grover-based ashenhurst-curtis decomposition using quantum language quipper. *Quantum Information & Computation*, 19(1-2):35–66, 2019.
28. Simanraj Sadana. Grover's search algorithm for n qubits with optimal number of iterations. *arXiv preprint arXiv:2011.04051*, 2020.
29. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. *Lecture Notes in Computer Science*, page 820–831, 1998.
30. Norio Koda and Tsutomu Sasao. Minimization method for and-exor expressions using lower bound theorem. *Systems and Computers in Japan*, 24(13):16–27, January 1993.
31. Tsutomu Sasao and Philipp Besslich. On the complexity of mod-2l sum pla's. *IEEE Transactions on Computers*, 39(2):262–266, 1990.
32. Ryan Williams. On the strong exponential time hypothesis.
33. Solving satisfiability problems using grover's algorithm. In *Qiskit Textbook*. 2021.
34. Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, F Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, et al. Qiskit: An open-source framework for quantum computing.