Boolean Matching Reversible Circuits: Algorithm and Complexity

Tian-Fu Chen[1,4,5] and Jie-Hong R. Jiang[1,2,3,4,5]
[1]*Graduate School of Advanced Technology, National Taiwan University*, Taipei, Taiwan
[2]*Graduate Institute of Electronics Engineering, National Taiwan University*, Taipei, Taiwan
[3]*Department of Electrical Engineering, National Taiwan University*, Taipei, Taiwan
[4]*Center for Quantum Science and Engineering, National Taiwan University*, Taipei, Taiwan
[5]*Physics Division, National Center for Theoretical Sciences*, Taipei, Taiwan
{d11k42001, jhjiang}@ntu.edu.tw

ABSTRACT

Boolean matching is an important problem in logic synthesis and verification. Despite being well-studied for conventional Boolean circuits, its treatment for reversible logic circuits remains largely, if not completely, missing. This work provides the first such study. Given two (black-box) reversible logic circuits that are promised to be matchable, we check their equivalences under various input/output negation and permutation conditions subject to the availability/unavailability of their inverse circuits. Notably, among other results, we show that the equivalence up to input negation and permutation is solvable in quantum polynomial time, while its classical complexity is exponential. This result is arguably the first demonstration of quantum exponential speedup in solving design automation problems. Also, as a negative result, we show that the equivalence up to both input and output negations is not solvable in quantum polynomial time unless UNIQUE-SAT is, which is unlikely. This work paves the theoretical foundation of Boolean matching reversible circuits for potential applications, e.g., in quantum circuit synthesis.

KEYWORDS

reversible circuits, Boolean matching, swap test, quantum algorithms

1 INTRODUCTION

Reversible logic circuits, simply called *reversible circuits*, are a class of Boolean circuits with n inputs and n outputs whose functions $\mathbb{B}^n \to \mathbb{B}^n$ form a bijection (one-to-one and onto) mapping, i.e., a permutation. Fundamentally, logical reversibility prevents information erasure and is a necessary condition for computation with extremely low energy dissipation [2, 9]. Remarkably, quantum computing [11] is intrinsically reversible. Every quantum gate, i.e., unitary operator, is reversible.[1] Reversible circuit synthesis has been extensively studied, e.g., [10, 12, 14–16], largely motivated by quantum computation due to the fact that several important quantum algorithms, such as Grover's algorithm [5], Simon's algorithm [13], etc., have oracle circuits, which are reversible circuits, as a key building block. In addition, reversible circuits also have applications in signal processing, cryptography, computer graphics, and nanotechnologies [12]. Constructing optimal reversible circuits is of practical importance. In this work, we are concerned with *Boolean matching* of reversible circuits.

[1]Although every quantum circuit is reversible, not every quantum circuit is referred to as a reversible circuit unless its unitary operator is a permutation matrix.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3657312

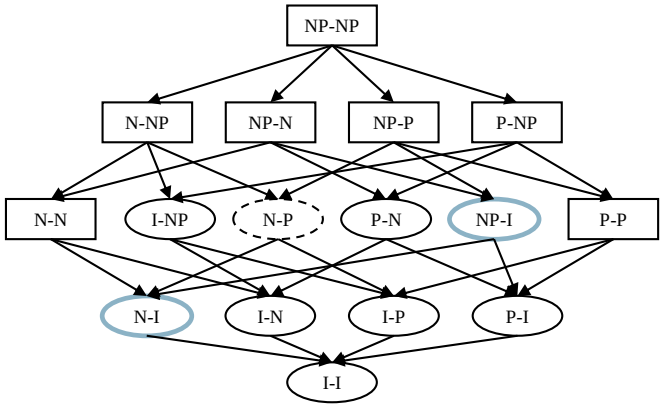

Figure 1: Domination relation among and computational complexities of various Boolean matching equivalences.

Conventionally, Boolean matching checks whether two (irreversible) logic circuits are equivalent up to the negation and/or permutation of inputs and/or outputs [6, 8]. It plays a vital role in logic synthesis and verification tasks such as technology mapping [1], engineering change order (ECO) [7], etc. Although well-studied for conventional Boolean circuits, Boolean matching on reversible circuits is largely, if not completely, missing. This work provides the first comprehensive study of Boolean matching reversible circuits and paves the theoretical foundation for potential applications. For example, template-based reversible logic synthesis [10] will greatly benefit from Boolean matching (extending from structural to functional matching), and so will oracle circuit synthesis and verification for quantum computation.

Given two (black-box) reversible logic circuits, also referred to as *oracles*, that are promised to be matchable, we explore the algorithm and complexity for Boolean matching on reversible circuits under different equivalences up to negation and/or permutation on their inputs and/or outputs. We define "X-Y equivalence" for X, Y ∈ {I, N, P, NP}, where X and Y denote equivalence conditions on the input and output sides, respectively, of the circuits, and I, N, P, and NP stand for identity, negation equivalence, permutation equivalence, and negation plus permutation equivalence, respectively. In total, there can be 16 combinations of equivalences, whose dominance relations are shown in the graph of Figure 1, where an edge (A, B) from A to B indicates equivalence B is subsumed by A. Note that the I-I equivalence is just the typical combinational equivalence problem without needing to identify negation and permutation conditions.

This work contributes to the first comprehensive characterization of the computational complexity in terms of the number of oracle access required to compute the negation/permutation conditions for the 16 equivalences. Figure 1 summarizes the results. The equivalences in ovals and rectangles correspond to easy (solvable in classical or quantum polynomial time) and hard (no easier than UNIQUE-SAT [4]) cases, respectively. As positive results, we provide concrete polynomial-time

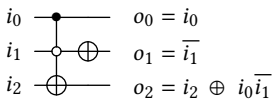

Figure 2: A reversible circuit example.

classical or quantum algorithms for checking the tractable equivalences. We note that the two gray-blue colored ovals (for N-I and NP-I equivalences) indicate their quantum, but not classical, polynomial-time solvability. To solve N-I and NP-I equivalences, we develop a quantum algorithm utilizing the *swap test* [3] as a subroutine.[2] The new algorithm achieves an exponential speedup over classical computation, adding to the rare quantum algorithm examples of attaining exponential speedup. Arguably, it is the first quantum algorithm with an exponential speedup in solving design automation and quantum program compilation problems. On the other hand, the dashed oval (for N-P equivalence) in Figure 1 indicates conditional classical polynomial-time solvability but with unknown quantum complexity. As negative results, we prove that N-N and P-P equivalences cannot be solved in quantum polynomial time unless UNIQUE-SAT can. Consequently, all other equivalences that dominate N-N or P-P equivalences are UNIQUE-SAT hard.

The rest of this paper is organized as follows. Section 2 provides essential backgrounds on reversible circuits and quantum computation. Section 3 formulates the Boolean matching problem of reversible circuits. For the equivalences that are polynomially solvable, Section 4 presents the corresponding algorithms and analyzes their complexities. Section 5 studies the equivalences that we prove to be harder than UNIQUE-SAT. Finally, we conclude this work in Section 6.

2 PRELIMINARIES

2.1 Reversible Circuits

An n-bit reversible circuit contains n input bits and n output bits (depicted with n lines with inputs at the left ends and outputs at the right ends) and implements a one-to-one and onto function $f : \mathbb{B}^n \to \mathbb{B}^n$, i.e., a permutation mapping. Clearly, f is reversible as its inverse function f^{-1} always exists. A reversible circuit can be generally represented by a circuit composed of multiple-controlled Toffoli (MCT) gates [12]. An MCT gate has $k = 0, 1, 2, \ldots$ control bits, each of which can be of a positive (indicated with a solid dot) or negative polarity (indicated with an empty circle), and conditionally flips a target bit (indicated with sign $\oplus$) when and only when all the negative and positive control bits are of values 0 and 1, respectively. In the special case of $k = 0$ and 1, the MCT gate corresponds to the NOT- and CNOT-gate (for a positive control bit), respectively. Figure 2 shows an example.

2.2 Quantum Computation

A *quantum bit* (*qubit*) can be in a superposition state over states $|0\rangle$ and $|1\rangle$, generally specified, in the Dirac notation, by $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and $\beta \in \mathbb{C}$ are referred to as the *probability amplitudes*, with $|\alpha|^2 + |\beta|^2 = 1$. Specifically, $|\alpha|^2$ and $|\beta|^2$ correspond to the probabilities for measuring the state $|\psi\rangle$ be in $|0\rangle$ and $|1\rangle$, respectively. A quantum system of n qubits can be specified by $|\psi\rangle = a_{0\ldots0}\,|0\ldots0\rangle + \ldots + a_{1\ldots1}\,|1\ldots1\rangle$, or alternatively written as a column vector $|\psi\rangle = [a_{0\ldots0}, \ldots, a_{1\ldots1}]^{\mathrm{T}}$. The evolution of a quantum system of n qubits is governed by a sequence of quantum gates $U_1, \ldots, U_k$, for each U_i being a $2^n \times 2^n$ unitary matrix (or operator) over complex numbers satisfying $U_i^\dagger = U_i^{-1}$, i.e., its conjugate transpose equals its inverse. The gate sequence constitutes a quantum circuit with a global unitary matrix U equal to the matrix product $U_k \cdots U_1$. Applying U on an initial quantum state $|\psi\rangle$ leads to the final state $|\psi'\rangle = U\,|\psi\rangle$. Given two arbitrary states $|\psi_1\rangle$ and $|\psi_2\rangle$, a unitary matrix U preserves their inner product, i.e., $\langle\psi_1|U^\dagger U|\psi_2\rangle = \langle\psi_1|\psi_2\rangle$, where $\langle\psi_1|\psi_2\rangle$ denotes the inner product of $|\psi_1\rangle$ and $|\psi_2\rangle$, and $\langle\psi_1|U^\dagger U|\psi_2\rangle$ is the inner product of $U\,|\psi_1\rangle$ and $U\,|\psi_2\rangle$. Note that the function mapping defined by a reversible circuit can be represented as a permutation matrix, which is a unitary operator.

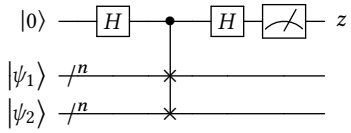

Figure 3: The circuit of quantum swap test.

In this work, we utilize the *swap test* [3], a quantum computation technique used to check how much two quantum states differ, in some of our algorithms. Given two n-qubit quantum states $|\psi_1\rangle$ and $|\psi_2\rangle$, the swap test applies the circuit shown in Figure 3 and measures the first qubit. After measurement, z is either in $|0\rangle$ of probability $\frac{1}{2} + \frac{1}{2}|\langle\psi_1|\psi_2\rangle|^2$ or in $|1\rangle$ of probability $\frac{1}{2} - \frac{1}{2}|\langle\psi_1|\psi_2\rangle|^2$. Hence, if $|\psi_1\rangle$ and $|\psi_2\rangle$ are identical, i.e., $|\langle\psi_1|\psi_2\rangle| = 1$, then the outcome z is always in $|0\rangle$. At the other extreme, if $|\psi_1\rangle$ and $|\psi_2\rangle$ are orthogonal, i.e, $|\langle\psi_1|\psi_2\rangle| = 0$, then the outcome z is in $|0\rangle$ and $|1\rangle$ with equal probability.

3 PROBLEM FORMULATION

The Boolean matching problem for reversible circuits can be stated as follows.

Problem 1 (Boolean matching reversible circuits). *Given two n-bit black-box reversible circuits C_1 and C_2 with C_1 and C_2 being promised to be X-Y equivalent for X, Y $\in$ {I, N, P, NP}, we are asked to find the corresponding negation functions $\nu_x, \nu_y : \{1, \ldots, n\} \to \mathbb{B}$ and permutation functions $\pi_x, \pi_y : \{1, \ldots, n\} \to \{1, \ldots, n\}$ of X and Y that make C_1 and C_2 X-Y equivalent, where $\nu(i) = 1$ indicates the i^{th} bit being negated and $\pi(i) = j$ indicates the i^{th} bit being permuted to the j^{th} bit.*

In the sequel, we omit the subscripts x, y in ν and π when the input/output information is clear from the context or immaterial to the discussion. We denote the reversible circuits of functions ν and π as C_ν and C_π, respectively. Furthermore, we do not distinguish a reversible circuit C from its underlying unitary matrix. Hence, concatenating a reversible circuit C_A (on the left) followed by circuit C_B (on the right) is represented as the matrix product $C_B C_A$. For example, N-P equivalent C_1 and C_2 means there exist C_ν and C_π to make $C_1 = C_\pi C_2 C_\nu$ for some input negation function ν and output permutation function π.

Notice that Problem 1 is a promise problem; that is, the circuits under test are promised to be Boolean matchable under certain equivalences. Finding a solution to the promise problem is crucial to even answering the non-promise version of the problem. It is because as long as we can solve Problem 1, a solution of the negation and permutation conditions can be obtained even if the equivalence relation is not promised. With the found negation and permutation conditions, only a single round of equivalence checking is needed to validate the equivalence relation. In contrast, without knowing the negation and permutation conditions, one may need to try an exponential number of equivalence checking rounds for all ν and π options.

Also note that Problem 1 assumes C_1 and C_2 are given as black boxes (oracles). The computational complexity of finding ν and π functions is measured in terms of the number of queries to the oracles. A variant problem of Problem 1 is to relax the assumption given not only C_1 and

[2]Besides the swap-test-based algorithm, we develop two more algorithms inspired by Simon's algorithm [13] and have to omit them due to space limit.

C_2 but also their inverse circuits C_1^{-1} and C_2^{-1}. Surely, if C_1 and C_2 are given as white boxes, we can always derive their inverse circuits as they are reversible. In Section 4, we will discuss how this relaxation may possibly simplify the computation.

4 TRACTABLE EQUIVALENCES

In this section, we investigate Boolean matching equivalences {I-NP, N-P, P-N, NP-I, N-I, I-N, I-P, P-I} that permit polynomial time identification of negation and permutation functions. For these equivalences, we develop efficient algorithms and analyze their complexities as summarized in Table 1, where n is the bit size of the circuits and ϵ is the failure probability for randomized algorithms.

4.1 I-N Equivalence

Proposition 1. *For I-N equivalence, finding ν for $C_1 = C_\nu C_2$ is of $O(1)$ query complexity.*

We set all inputs of C_1 and C_2 to 0 and bit-wisely compare their outputs. Then, for $i = 1, \ldots, n$, we have $\nu(i) = 1$ if and only if the output patterns of C_1 and C_2 differ at the i^{th} bit. The computation requires only one oracle query.

4.2 I-P Equivalence

Proposition 2. *For I-P equivalence, finding π for $C_1 = C_\pi C_2$ is of $O(\log(n))$ query complexity if C_1^{-1} or C_2^{-1} is available, and of query complexity $O(\log(n)+\log(1/\epsilon))$ by a randomized algorithm with success probability $1-\epsilon$ if C_1^{-1} and C_2^{-1} are unavailable.*

If C_2^{-1} is available, let $C = C_1 C_2^{-1}$ by concatenating C_2^{-1} and C_1, and thus C is functionally equivalent to C_π. We use $\lceil \log_2 n \rceil$ input patterns to decide π as follows. For the i^{th} input pattern, the input of the j^{th} bit of the circuit is the i^{th} least significant bit of the binary code of index j. In other words, the input sequence to the j^{th} bit of the circuit is the binary code of j with the least significant bit arriving first. Then $\pi(p) = q$ if and only if the sequential input of the p^{th} bit is the same as the sequential output of the q^{th} bit. Hence, $O(\log n)$ oracle queries of C are needed. Similarly, if C_1^{-1} is available, we let $C = C_2 C_1^{-1}$, which equals $C_{\pi^{-1}}$. The same way applies to find π^{-1}, and thus π.

If both C_1^{-1} and C_2^{-1} are unavailable, the following randomized algorithm is applied to find π in two steps. First, repeat k times to feed random input patterns to both C_1 and C_2 and record their output patterns, where k is a number related to the failure probability to be explained shortly. Second, for each output bit $b_1 = 1, \ldots, n$ of C_1, find the unique output bit b_2 of C_2 such that they share the same output sequence, which indicates $\pi(b_1) = b_2$.

Table 1: Complexity of computing various equivalences.

Inverse circuit	Equivalence type	Computing paradigm	Complexity
available	N-I*, I-N*	classical	$O(1)$
	I-P*, P-I*, N-P**, P-N*, I-NP*, NP-I*	classical	$O(\log n)$
not available	I-N	classical	$O(1)$
	I-P, I-NP	classical	$O(\log n + \log(1/\epsilon))$
	P-I, P-N	classical	$O(n)$
	N-I	quantum	$O(n \log(1/\epsilon))$
	NP-I	quantum	$O(n^2 \log(1/\epsilon))$

"*" indicates one inverse circuit of either C_1 or C_2 is required.
"**" indicates both inverse circuits of C_1 and C_2 are needed.

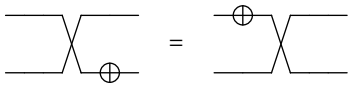

Figure 4: Exchanging the order between negation and permutation circuits.

To ensure the second step finds a unique b_2 for each b_1, there cannot be two bits sharing the same output sequence. Hence, we require k iterations of the first step to make the output sequence long enough. Note that the range under the mapping of a reversible circuit must cover all 2^n combinations, and thus, different input patterns must yield different output patterns. As long as the input patterns are uniformly generated at random, the output patterns are also uniformly at random. Let $B_1(j)$ be the output sequence of the j^{th} bit in C_1. Since there are 2^k possible output sequences for any output bit j, the success probability Pr that $B_1(j_1) \neq B_1(j_2)\ \forall j_1 \neq j_2$ is

$$\begin{aligned} \Pr &= \frac{(2^k)(2^k-1)\ldots(2^k-n+1)}{(2^k)^n} \geq \left(\frac{2^k-n+1}{2^k}\right)^n \\ &\approx 1 - \frac{n(n-1)}{2^k} \quad (\text{when } n \ll 2^k). \end{aligned} \tag{1}$$

To make $\Pr \geq 1-\epsilon$, we need $k \geq \log_2 \frac{n(n-1)}{\epsilon} = O(\log(n) + \log(1/\epsilon))$. Therefore, the query complexity of the algorithm is $O(\log(n)+\log(1/\epsilon))$.

4.3 I-NP Equivalence

Proposition 3. *For I-NP equivalence, finding ν and π for $C_1 = C_\pi C_\nu C_2$ is of $O(\log(n))$ query complexity if either C_1^{-1} or C_2^{-1} is available, and of complexity $O(\log(n)+\log(1/\epsilon))$ by a randomized algorithm with success probability $1-\epsilon$ if C_1^{-1} and C_2^{-1} are unavailable.*

Observe that the order of C_ν and C_π can be exchanged according to the identity shown in Figure 4. If C_2^{-1} is available, let $C = C_1 C_2^{-1}$ by concatenating C_2^{-1} and C_1, which is functionally equivalent to $C_\pi C_\nu$. We temporally let $C_\pi C_\nu = C_{\nu'} C_{\pi'}$. To decide ν', we set all inputs to 0. Since all inputs have the same value, the initial permutation circuit $C_{\pi'}$ has no effect, and thus, $\nu'(i) = 1$ if and only if the output of the i^{th} bit is 1. This step requires only one oracle query. After $\nu'(i)$ is decided, the method mentioned in Section 4.2 can be slightly modified to find π', which needs $O(\log(n))$ oracle queries. Finally, ν' and π' can be transformed back to get ν and π. Similarly, if C_1^{-1} is available, we let $C = C_2 C_1^{-1}$ and follow a similar procedure to obtain ν^{-1} and π^{-1}, and transform them back to ν and π.

If both C_1^{-1} and C_2^{-1} are unavailable, a randomized algorithm similar to the one in Section 4.2 can be employed to find π. The modification is that when we look for the unique b_2 in C_2 for each bit b_1 in C_1, we now recognize two kinds of b_2. One is the same as before, where the b_1^{th} bit in C_1 has the same output sequence as the b_2^{th} bit in C_2, which indicates $\pi(b_1) = b_2$ and $\nu(b_2) = 0$. The other is that the b_1^{th} bit in C_1 has exactly the bitwise-flipped output sequence to the b_2^{th} bit in C_2, which indicates $\pi(b_1) = b_2$ and $\nu(b_2) = 1$. An analysis similar to that in Section 4.2 gives the $O(\log(n) + \log(1/\epsilon))$ complexity.

4.4 P-I Equivalence

Proposition 4. *For P-I equivalence, finding π for $C_1 = C_2 C_\pi$ is of $O(\log(n))$ query complexity if C_1^{-1} or C_2^{-1} is available, and of $O(n)$ complexity if C_1^{-1} and C_2^{-1} are unavailable.*

If C_1^{-1} (resp. C_2^{-1}) is available, we obtain $C = C_1^{-1}C_2$ (resp. $C = C_1C_2^{-1}$), which is functionally equivalent to $C_{\pi^{-1}}$ (resp. C_π). As proposed in Section 4.2, π can be decided in $O(\log(n))$ number of oracle queries.

If both C_1^{-1} and C_2^{-1} are unavailable, π can be found in $O(n)$ number of oracle queries with the following algorithm in two steps. First, prepare n one-hot input patterns. The i^{th} pattern assigns 0 to all bits except one 1 to the i^{th} bit. For the i^{th} input pattern, collect the corresponding output patterns $P_{o1,i}$ of C_1 and $P_{o2,i}$ of C_2. Let $M_1[P_{o1,i}] = i$ and $M_2[i] = P_{o2,i}$. Second, for each $i = 1, \ldots, n$, let $\pi(i) = M_1[M_2[i]]$. Thereby, π is found in $O(n)$ oracle queries.

4.5 N-I Equivalence

Proposition 5. *For N-I equivalence, finding ν for $C_1 = C_2C_\nu$ is of $O(\log(n))$ query complexity if either C_1^{-1} or C_2^{-1} is available, and of $O(n\log(1/\epsilon))$ query complexity by a randomized quantum algorithm with success probability $1-\epsilon$ if C_1^{-1} and C_2^{-1} are unavailable but C_1 and C_2 can take quantum states as inputs.*

If C_2^{-1} is available, let $C = C_2^{-1}C_1 = C_\nu$. By setting all inputs of C to 0, $\nu(i) = 1$ if and only if the i^{th} bit at output is 1. Similarly, if C_1^{-1} is available, let $C = C_1^{-1}C_2 = C_{\nu^{-1}}$. Observing $C_{\nu^{-1}} = C_\nu$, the same approach can be applied to decide ν.

If C_1^{-1} and C_2^{-1} are both unavailable, Theorem 1 provides the complexity lower bound of classical algorithms.

Theorem 1. *If C_1^{-1} and C_2^{-1} are unavailable, a classical algorithm to find the negation function for N-I equivalence needs at least $\Omega(2^{n/2})$ oracle queries.*

Proof. Given black boxes C_1 and C_2, they can be accessed only by queries, specifically, one output response per input query. Only when C_1 and C_2 yield the same output upon two input patterns to C_1 and C_2, we can infer ν by comparing these two input patterns. A classical algorithm can only randomly try different input patterns until C_1 and C_2 produce the same output pattern, i.e., a collision happens. Assuming k queries to C_2 are made to match an output pattern of C_1, then the probability of collisions not happening is

$$\begin{aligned} \Pr &= \frac{(2^n)(2^n-1)\ldots(2^n-k+1)}{(2^n)^k} \geq \left(\frac{2^n-k+1}{2^n}\right)^k \\ &\approx 1-\frac{k(k-1)}{2^n} \quad (\text{when } k \ll 2^n). \end{aligned} \tag{2}$$

For Pr being smaller than some constant threshold c, the query number k should be at least $\sqrt{(1-c)\cdot 2^n}$, which yields the lower bound $\Omega(2^{n/2})$. □

Fortunately, assuming that the reversible circuits C_1 and C_2 can take quantum states as input, finding ν for N-I equivalence without C_1^{-1} and C_2^{-1} is solvable in quantum polynomial time by Algorithm 1. The key idea is to utilize superposition states to disable NOT-gates, such that only one NOT-gate is active at a time. In addition, the swap test [3] is exploited as a subroutine to compare two quantum states. Algorithm 1 proceeds in n iterations, where the i^{th} iteration decides the value of $\nu(i)$. In the i^{th} iteration, the inputs of the i^{th} qubit of both circuits are set to $|0\rangle$, while all other qubits are initialized to $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ in lines 2 to 3. Line 4 sets $\nu(i) = 0$ in default. In lines 5 to 8, k iterations of swap tests are performed for k being determined by the failure probability as to be explained. In line 6, we execute C_1 and C_2 on input pattern $P_{in(i)}$ to obtain their final states $C_1(P_{in(i)})$ and $C_2(P_{in(i)})$, and these two final states are sent to the swap test. If the measurement outcome of the swap test is 1, we conclude that $\nu(i) = 1$, and the iteration is terminated, as shown in lines 7 to 8. Otherwise, if the measurement outcome is 0 for k times, then we have high confidence that $\nu(i) = 0$.

Algorithm 1: Quantum Algorithm for N-I Equivalence

```
Input  : C1, C2: Two n-bit reversible circuits
Output : ν: Negation function making C1 = C2Cν
1 for i = 1, 2, ..., n
2 |   P_in(i) ← [|+⟩, |+⟩, ..., |+⟩]
3 |   P_in(i)[i] ← |0⟩
4 |   ν(i) ← 0
5 |   for j = 1, 2, ..., k
6 |   |   if SWAP_TEST(C1(P_in(i)), C2(P_in(i))) = 1
7 |   |   |   ν(i) ← 1
8 |   |   |   break
9 return ν
```

The correctness of Algorithm 1 is established below. We first show that $\nu(1)$ can be correctly decided, and $\nu(i)$ for $i = 2, \ldots, n$ follow with the same derivation. When deciding $\nu(1)$, the input states of C_1 and C_2 are both prepared as $|\psi\rangle = |0\rangle \otimes |+\rangle \otimes \ldots \otimes |+\rangle$. The output states of C_1 and C_2 are $C_1|\psi\rangle$ and $C_2|\psi\rangle$, respectively. Since $C_1 = C_2C_\nu$, the output state of C_1 can also be written as $C_2C_\nu|\psi\rangle$. Let $C_\nu|\psi\rangle = |\psi'\rangle$. Note that a NOT-gate is a Pauli X operator, whose application on $|+\rangle$ has no effect as $X|+\rangle = |+\rangle$. Hence, only the NOT-gate, if it exists, at the first bit in C_ν can make an effect on $|\psi\rangle$. There are two cases: In the first case, the first bit in C_ν has a NOT-gate, i.e., $\nu(1) = 1$. Then $|\psi'\rangle = |1\rangle \otimes |+\rangle \otimes \ldots \otimes |+\rangle$, and $\langle\psi'|\psi\rangle = \langle 1|0\rangle\langle +|+\rangle \cdots \langle +|+\rangle = 0 \cdot 1 \cdots 1 = 0$. Note that the final state of C_1 and C_2 are $C_2|\psi'\rangle$ and $C_2|\psi\rangle$, respectively. As mentioned in Section 2, quantum circuits must preserve the inner product of states. Therefore, after applying C_2 on both $|\psi'\rangle$ and $|\psi\rangle$, the final states of the two circuits still have inner product 0. Finally, when applying the swap test on the output states of C_1 and C_2, there is a 50% probability of obtaining a measurement outcome 1.

In the second case, the first bit in C_N does not have a NOT-gate, i.e., $\nu(1) = 0$. In this case, the output states of C_1 and C_2 are identical. When we apply the swap test to them, we always obtain a measurement outcome of 0. Hence, once the measurement outcome is 1, we can conclude $\nu(1) = 1$. Otherwise, if the measurement outcome is 0 for k times, $\nu(1) = 0$ is of high confidence. The probability of success is $1 - 1/2^k$. To make the failure probability $\leq \epsilon$, we require $k \geq \log_2(1/\epsilon)$. Since a similar process is used to decide $\nu(i)$ for all $i = 1, \ldots, n$, the query complexity of Algorithm 1 is $O(n\log(1/\epsilon))$.

4.6 NP-I Equivalence

Proposition 6. *For NP-I equivalence, finding ν and π for $C_1 = C_2C_\pi C_\nu$ is of $O(\log(n))$ query complexity if C_1^{-1} or C_2^{-1} is available, and of $O(n^2\log(1/\epsilon))$ query complexity by a randomized quantum algorithm with success probability $1-\epsilon$ if C_1^{-1} and C_2^{-1} are unavailable but C_1 and C_2 can take quantum states as inputs.*

If C_1^{-1} or C_2^{-1} is available, a method similar to that in Section 4.3 can decide ν and π in $O(\log(n))$ oracle queries. If C_1^{-1} and C_2^{-1} are both unavailable, the following quantum algorithm can be applied. First, find C_π by deciding whether the b_1^{th} bit in C_1 is mapped to the b_2^{th} bit in C_2 for all $b_1, b_2 \in \{1, 2, \ldots, n\}$. For each (b_1, b_2) pair, initialize the b_1^{th} qubit in C_1 and the b_2^{th} qubit in C_2 to $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, while all other qubits are initialized to $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Then, execute both circuits and compare their final states by the swap test. Let $\pi(b_2) = b_1$ if and only if the measurement outcome of the swap test is 0 for k times. After π is found, Algorithm 1 can be slightly modified to find ν further.

The key idea of the algorithm is to disable C_ν first, so as to focus on finding C_π. To disable C_ν, choose the input state to be $|+\rangle$ and

$|-\rangle$. Note that a NOT-gate, i.e., a Pauli X operator, acting on $|-\rangle$ yields $X|-\rangle = -|-\rangle$, where the global phase -1 can be ignored. Moreover, the input states of C_1 and C_2 are identical when $b_1 = b_2$ and are orthogonal when $b_1 \neq b_2$. Therefore, the final states of C_1 and C_2 are identical if $\pi(b_2) = b_1$, and orthogonal otherwise. By applying the swap test k times on the final states of C_1 and C_2, we can decide whether $\pi(b_2) = b_1$. Again, we can derive that $k = \log_2(1/\epsilon)$ to make the failure probability $\leq \epsilon$, so the overall complexity of the algorithm is $O(n^2 \log(1/\epsilon))$.

4.7 P-N Equivalence

Proposition 7. *For P-N equivalence, finding π and ν for $C_1 = C_\nu C_2 C_\pi$ is of $O(\log(n))$ query complexity if C_1^{-1} or C_2^{-1} is available, and of $O(n)$ query complexity if C_1^{-1} and C_2^{-1} are unavailable.*

The result comes from the fact that this problem can be reduced to P-I equivalence testing, so it has the same complexity as P-I equivalence. Regardless of whether C_1^{-1} and C_2^{-1} are available, we first decide ν by setting all inputs to 0 and observing the outputs. Since all inputs are the same, input permutation makes no effect. Then $\nu(i) = 1$ if and only if the output patterns of C_1 and C_2 differ at the i^{th} bit. This step only takes $O(1)$ oracle queries and does not affect the complexity analysis. After ν is found, we obtain $C_3 = C_\nu C_2$. Note that C_1 and C_3 are P-I equivalent now, and the P-N equivalence can be reduced to P-I equivalence.

4.8 N-P Equivalence

Proposition 8. *For N-P equivalence, finding ν and π for $C_1 = C_\pi C_2 C_\nu$ is of $O(\log(n))$ query complexity if C_1^{-1} and C_2^{-1} are both available.*

If inverses of C_1 and C_2 are both available, the method mentioned in Section 4.7 is applicable to find ν and π by the fact that $C_1^{-1} = C_{\nu^{-1}} C_2^{-1} C_{\pi^{-1}}$. Then ν^{-1} and π^{-1} are transformed to ν and π. The complexity is the same as P-N equivalence testing.

If ν and π when both C_1^{-1} and C_2^{-1} are unavailable, whether there exists an efficient quantum algorithm remains an open problem for our future work.

5 INTRACTABLE EQUIVALENCES

We show that the equivalence classes not solved in Section 4 are more difficult than the promise *UNIQUE-SAT problem* [4]. Given a conjunctive normal form (CNF) Boolean formula φ that is promised to have at most one satisfying assignment, the UNIQUE-SAT problem asks to decide whether it is satisfiable. It has been proved that SAT is randomly reducible to UNIQUE-SAT [17], and thus UNIQUE-SAT is believed to be a difficult problem. In what follows, we will show that the UNIQUE-SAT problem is polynomially reducible to N-N and P-P equivalences. Since equivalences {NP-NP, N-NP, NP-N, NP-P, P-NP} subsume N-N or P-P, they are thus harder than UNIQUE-SAT.

5.1 Hardness of N-N Equivalence

Theorem 2. *For N-N equivalence, finding ν_x and ν_y for $C_1 = C_{\nu_y} C_2 C_{\nu_x}$ is no easier than UNIQUE-SAT.*

Proof. We establish a polynomial-time reduction from UNIQUE-SAT to N-N equivalence as follows. Consider a CNF formula $\varphi = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ over variables $\{x_1, x_2, \ldots, x_n\}$ promised to have at most one satisfying assignment. Let clause $c_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ldots \vee \ell_{i,k_i})$, a disjunction of k_i literals. We construct the UNIQUE-SAT encoding reversible circuit C_1 as shown in the black part of Figure 5(a) (excluding the red part), where the first n bits $b_{x_1}, \ldots, b_{x_n}$ correspond to the input variables, the next m ancilla bits $b_{a_1}, \ldots, b_{a_m}$ correspond to the valuations of the clauses, and two extra ancilla bits b_b and b_z are used to generate the value of φ. Each $U(\varphi) = U(c_m) \cdots U(c_1)$ is the concatenation of the clause-encoding circuits. For a clause $c_i = \ell_{i,1} \vee \ldots \vee \ell_{i,k_i}$, its clause-encoding circuit is an MCT gate followed by a NOT-gate. In the MCT gate, each literal $\ell_{i,j}$ in c_i corresponds to a control bit. If $\ell_{i,j}$ equals x_s (resp. $\overline{x_s}$), then a negative (resp. positive) control bit is applied on b_{x_s}. The target bit of the MCT gate and the NOT-gate are applied on b_{a_i}. E.g., the clause-encoding circuit of clause $c_1 = \overline{x_1} \vee x2 \vee \overline{x_3}$ is shown in Figure 5(b). Hence, for the whole $U(\varphi)$ block, if the input value of b_{x_j} equals x_j and the input value of b_{a_i} equals a_i, then the output values of all b_{x_j} remain the same as their input values, while the output value of b_{a_i} equals $a_i \oplus (\overline{\ell_{i,1}} \ldots \overline{\ell_{i,k_i}}) \oplus 1 = a_i \oplus c_i$. Note that $U(\varphi)^{-1} = U(\varphi)$.

For the overall UNIQUE-SAT encoding circuit in Figure 5(a), let v denote the input value of bit b_v. We note that the value of b_{a_i} is changed to $a_i \oplus c_i$ at time t_2 and t_4 and is returned back to a_i at time t_2 and the end of the circuit. Moreover, the value of b_b is changed to $b \oplus (\overline{a_1} \ldots \overline{a_m})$ at time t_1 and returned back to b at time t_3. It can be derived that the output values of the first $n+m+1$ bits are the same as their input values, while the output value of b_z is $z \oplus f$, for

$$\begin{aligned} f &= \left[\bigwedge_{i=1}^{m} (a_i \oplus c_i)(b \oplus (\overline{a_1} \ldots \overline{a_m}))\right] \oplus \left[\bigwedge_{i=1}^{m} (a_i \oplus c_i) b\right] \\ &= [(a_1 \oplus c_1) \ldots (a_m \oplus c_m)] \wedge (\overline{a_1} \ldots \overline{a_m}) \\ &= (c_1 \ldots c_m) \wedge (\overline{a_1} \ldots \overline{a_m}) \quad = \varphi \wedge (\overline{a_1} \ldots \overline{a_m}). \end{aligned} \tag{3}$$

Hence, $f = 1$ if and only if all a_i's are 0 and $(x_1, \ldots, x_n)$ is a satisfying assignment of φ.

On the other hand, we can construct a circuit C_2 shown in Figure 5(c). Let v denote the input value of bit b_v. It is easy to derive that the output values of the first $n + m + 1$ bits are the same as their input values, while the output value of b_z is $z \oplus g$, where $g = (x_1 \ldots x_n) \wedge (\overline{a_1} \ldots \overline{a_m})$. Taking the UNIQUE-SAT encoding circuit as C_1 and comparing f and g, it can be verified that C_1 and C_2 are N-N equivalent if and only if φ is satisfiable. The intuition is as follows. First, $\pi_x(i) = \pi_y(i)$ for $i = 1, \ldots, n + m + 1$ must hold to ensure the output values of the first $n + m + 1$ bits are the same as their input values. Then we observe that two NOT-gates applied before and after a control bit will flip the control polarity. Therefore, if $v_1(i) = v_2(i) = 1$, then x_i is in fact negatively controlling, indicating that $x_i = 0$ is the necessary condition to make $\varphi = 1$. Otherwise, $v_1(i) = v_2(i) = 0$ indicates that x_i is positively controlling, so $x_i = 1$ is the necessary condition to make $\varphi = 1$. Therefore, once we can find ν_x and ν_y to make C_1 and C_2 N-N equivalent, the unique satisfying assignment of φ is found. Even if we do not know whether $C_1 = C_{\nu_y} C_2 C_{\nu_x}$, we can still try the process and obtain a candidate solution. The validity of the candidate solution can be easily verified in linear time by substituting it into φ. Moreover, the reduction process is polynomial since there are only $8m + 4$ MCT gates in the UNIQUE-SAT encoding circuit. Hence, UNIQUE-SAT is polynomially reducible to the N-N equivalence problem, and the theorem follows. □

5.2 Hardness of P-P Equivalence

Theorem 3. *For P-P equivalence, finding π_x and π_y for $C_1 = C_{\pi_y} C_2 C_{\pi_x}$ is no easier than UNIQUE-SAT.*

Proof. Consider again the CNF formula φ in Section 5.1. We create another CNF formula φ' by adding n extra variables $y_1, \ldots, y_n$ and setting $\varphi' = \varphi \wedge c_{m+1} \wedge \ldots \wedge c_{m+2n}$, where $(c_{m+2j-1}) \wedge (c_{m+2j}) = (x_j \vee y_j) \wedge (\overline{x_j} \vee \overline{y_j})$, $j = 1, \ldots n$. That is, we make a dual-rail encoding on the variables of φ in φ' and set $y_j = \overline{x_j}$ for all y_j. Hence, φ is satisfiable if and only if φ' is satisfiable. Then the same method mentioned in *Section* 5.1 is applied to encode φ' into C_1, as shown in Figure 5(a) (including both the black and red parts). Again, the output values of the first $4n + m + 1$ bits are always the same as their input values. For the last bit b_z, if its input value is z, then its output value is $z \oplus f$, where

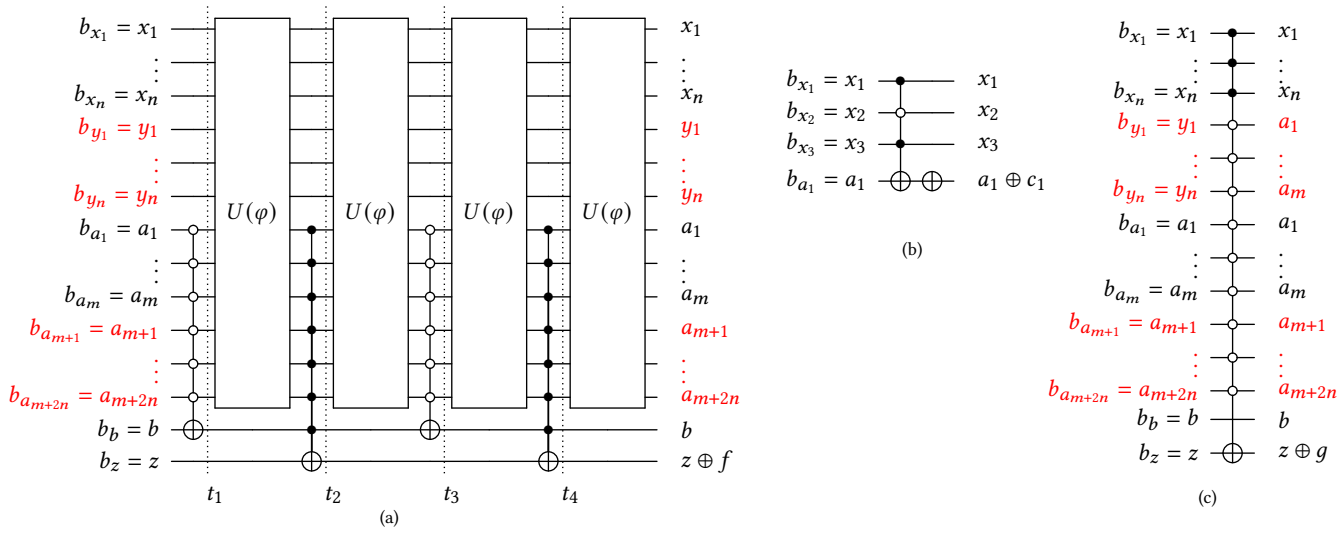

Figure 5: (a) The UNIQUE-SAT encoding circuit, (b) a clause-encoding circuit $U(c)$ of clause $c = \overline{x_1} \vee x2 \vee \overline{x_3}$, (c) the C_2 circuit for Boolean matching.

$f = \varphi' \wedge (\overline{a_1} \ldots \overline{a_{m+2n}})$. On the other hand, we can construct a circuit C_2 shown in Figure 5(c) (including both black and red parts), where the first n bits are positive-control bits, the $(n+1)^{\text{th}}$ to the $(4n+m)^{\text{th}}$ bits are negative-control bits, and the last bit b_z is the target bit.

By comparing f and g, it can be verified that C_1 and C_2 are P-P equivalent if and only if φ' is satisfiable. The intuition is as follows. First, we note that $\pi_x^{-1} = \pi_y$ must hold to ensure the output values of the first $4n + m + 1$ bits are the same as their input values. Therefore, the input and output permutations are equivalently just permuting the control bits. Second, the i^{th} bit is positively controlling if it falls in the positive-control region ($0 < \pi_x^{-1}(i) \leq n$), or it is negatively controlling if it falls in the negative-control region ($n < \pi_x^{-1}(i) \leq 4n + m$). If x_i is permuted to the positive-control region and y_i is permuted to the negative-control region, then $x_i = \overline{y_i} = 1$ is the necessary condition to make $\varphi' = 1$. Otherwise, if x_i is permuted to the negative-control region and y_i is permuted to the positive-control region, then $x_i = \overline{y_i} = 0$ is the necessary condition to make $\varphi' = 1$. Therefore, once we can find π_x and π_y to make C_1 and C_2 P-P equivalent, the unique satisfying assignment of φ' is found, which can be easily transformed to the unique satisfying assignment of φ. Even if we do not know whether C_1 and C_2 are P-P equivalent or not, we can still try the process and obtain a candidate solution. The validity of the candidate solution can be easily verified by substituting it into φ. Moreover, the reduction process is polynomial since there are only $8m + 4$ MCT gates in the UNIQUE-SAT encoding circuit. Hence, UNIQUE-SAT is polynomially reducible to the P-P equivalence problem, and the theorem follows. □

6 CONCLUSIONS AND FUTURE WORK

This work provided the first comprehensive study on various equivalences for Boolean matching of reversible circuits by characterizing their computational complexities. For the tractable equivalences, polynomial-time (classical or quantum) algorithms were devised. For the intractable equivalences, their hardness results were established. The foundation paved in this work may open new Boolean matching applications, e.g., in template-based reversible logic synthesis and in quantum program compilation for oracle circuit minimization. Moreover, our swap-test-based algorithm demonstrates the first example with an exponential quantum speedup over classical computation in design automation research. It may inspire the development of new types of quantum algorithms and applications. For future work, we intend to resolve the remaining open problem regarding the quantum complexity of N-P equivalence.

ACKNOWLEDGMENTS

This work was supported in part by the National Science and Technology Council of Taiwan under grants 112-2119-M-002-017 and 113-2119-M-002-024, and the NTU Center of Data Intelligence: Technologies, Applications, and Systems under grant NTU-113L900903. The authors thank IBM Q Hub at NTU and Quantum Technology Cloud Computing Center at NCKU for supporting experimental validation.

REFERENCES

[1] Luca Benini and Giovanni De Micheli. 1997. A survey of Boolean matching techniques for library binding. *ACM Transactions on Design Automation of Electronic Systems* 2, 3 (1997), 193–226.
[2] Charles H Bennett. 1973. Logical reversibility of computation. *IBM journal of Research and Development* 17, 6 (1973), 525–532.
[3] Harry Buhrman, Richard Cleve, John Watrous, and Ronald De Wolf. 2001. Quantum fingerprinting. *Physical Review Letters* 87, 16 (2001), 167902.
[4] Oded Goldreich. 2006. On promise problems: A survey. In *Theoretical Computer Science: Essays in Memory of Shimon Even*. Springer, 254–290.
[5] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proc. STOC*. 212–219.
[6] Hadi Katebi and Igor L. Markov. 2010. Large-scale Boolean matching. In *Proc. DATE*. 771–776.
[7] Smita Krishnaswamy, Haoxing Ren, Nilesh Modi, and Ruchir Puri. 2009. DeltaSyn: An efficient logic difference optimizer for ECO synthesis. In *Proc. ICCAD*. 789–796.
[8] Chih-Fan Lai, Jie-Hong R. Jiang, and Kuo-Hua Wang. 2010. BooM: A decision procedure for Boolean matching with abstraction and dynamic learning. In *Proc. DAC*. 499–504.
[9] Rolf Landauer. 1961. Irreversibility and heat generation in the computing process. *IBM journal of research and development* 5, 3 (1961), 183–191.
[10] D. Michael Miller, Dmitri Maslov, and Gerhard W. Dueck. 2003. A transformation based algorithm for reversible logic synthesis. In *Proc. DAC*. 318–323.
[11] M. A. Nielsen and I. L. Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge University Press.
[12] Mehdi Saeedi and Igor L Markov. 2013. Synthesis and optimization of reversible circuits—a survey. *ACM Computing Surveys* 45, 2 (2013), 1–34.
[13] Daniel R Simon. 1997. On the power of quantum computation. *SIAM Journal on Computing* 26, 5 (1997), 1474–1483.
[14] Mathias Soeken, Stefan Frehse, Robert Wille, and Rolf Drechsler. 2012. Revkit: A toolkit for reversible circuit design. *Journal of Multiple-Valued Logic and Soft Computing* 18, 1 (2012), 55–65.
[15] Mathias Soeken, Martin Roetteler, Nathan Wiebe, and Giovanni De Micheli. 2017. Hierarchical reversible logic synthesis using LUTs. In *Proc. DAC*. 1–6.
[16] Mathias Soeken, Robert Wille, Oliver Keszocze, D Michael Miller, and Rolf Drechsler. 2015. Embedding of large Boolean functions for reversible logic. *ACM Journal on Emerging Technologies in Computing Systems* 12, 4 (2015), 1–26.
[17] Leslie G Valiant and Vijay V Vazirani. 1985. NP is as easy as detecting unique solutions. In *Proc. STOC*. 458–463.