A Quantum SMT Solver for Bit-Vector Theory

Shang-Wei Lin
Singapore Institute of Technology
shangwei.lin@singaporetech.edu.sg

Si-Han Chen, Lei-Han Yao, Yu-Chung Chen, and Yean-Ru Chen
National Cheng Kung University, Taiwan
chenyr@mail.ncku.edu.tw

***Abstract*—Satisfiability modulo theory (SMT) has a wide range of applications because of its powerful ability of problem encoding. Given an SMT formula F, the classical lazy approach for SMT solving tries to (1) abstract F to a Boolean formula F_B, (2) find a Boolean solution to F_B, and (3) check if the Boolean solution is consistent with the theory. If the solution found is not consistent, the SMT solver needs to perform steps (2) and (3) back and forth until a consistent solution is found. This classical approach becomes unscalable when the search space is huge. In this work, we develop a quantum SMT solver based on Grover's algorithm to solve formulas on the theory of bit-vectors. With the characteristic of superposition in quantum computing, the proposed approach is able to consider all the possible inputs in parallel and check their consistency between two domains in one shot, which brings significant speed-up for SMT solving.**

***Index Terms*—quantum computing, Grover's algorithm, SMT solver, bit-vector theory**

I. Introduction

Satisfiability modulo theory (SMT) is the problem of determining the satisfiability of a first-order formula with respect to a decidable first-order theory. SMT is widely used in plenty of applications, particularly in formal verification, ranging from recent topics like neural networks [6], [13], smart contracts [3], [5], hardware circuit designs [2], [10], etc.

In this work, we focus on quantifier free bit-vectors [20], denoted by $\mathcal{BV}$. Consider the following SMT formula on the theory of unsigned bit-vectors. Let us call it formula F :

$$\Big[(a>b)\vee(a<b)\vee(a=b)\Big]\wedge\Big[\neg(a>b)\vee\neg(a<b)\vee\neg(a=b)\Big],$$

where a, b and c are 2-bit integer variables. Formula F is *satisfiable* as there exists an assignment $a = 2$ and $b = 2$, with binary representation 10 and 10, respectively, making F evaluate to True. Such an assignment is called a *solution*.

To solve an SMT problem, most SMT solvers adopt the classical *lazy approach* [20], consisting of three main steps: (1) abstract the original formula as a Boolean formula, (2) find a solution to the abstract Boolean formula, and (3) check the consistency between Boolean and the original theory domains. If the solution found in step (2) is not consistent in both domains, step (2) continues until a consistent solution is found, or the formula is concluded with no solution.

For instance, if we use Boolean variables x, y, and z to denote the predicate $(a > b)$, $(a < b)$, and $(a = b)$, respectively, formula F can be abstracted as the Boolean formula F_B: $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$.

A SAT (satisfiability) solver can be used to find a solution to formula F_B, and a bit-vector theory solver can help to check the consistency. Fig. 1 illustrates how the lazy approach solves formula F. In the first three iterations, the SAT solver finds an assignment but rejected by the theory solver. A solution to formula F_B is found in iteration 4, which is consistent with the theory domain, and the solving process finishes.

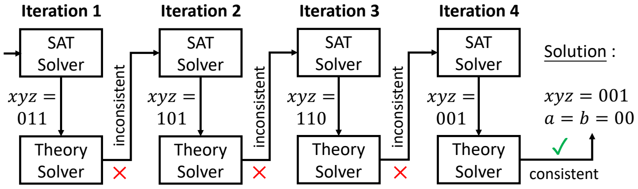


Fig. 1. SMT solving by the classical lazy approach

One can observe that each iteration of this back-and-forth process only checks the consistency for one Boolean solution (against the theory domain) at a time. If the search space is huge, this approach becomes unscalable.

Recently, quantum computing is widely used to solve traditionally difficult problems by taking advantage of the nature of superposition in quantum systems. Grover's algorithm [12], one of the famous quantum algorithms, helps to search targets among a search space. There are two components in Grover's algorithm: (1) an *oracle*, and (2) the *diffuser*. In a nutshell, the oracle answers the question whether an object is our target in the search space, while the diffuser tries to maximize the probability of the target when measured. Interested readers can refer to the appendix of the longer version [1].

To use Grover's algorithm, the key is to provide the oracle. As long as the oracle can correctly identify the targets in the search space, the diffuser, which is standard and independent from the search problem, can help to "extract" the targets. In this work, we develop a quantum SMT solver based on Grover's algorithm. More specifically, given an arbitrary SMT formula, we develop a methodology to generate its oracle required by Grover's algorithm to search the solutions.

Let us get back to formula F. Variables a and b can be represented by two binary strings a_1a_2 and b_1b_2, respectively, where a_1, a_2, b_1, b_2 are Boolean variables. Together with formula F_B, seven bits can be used to represent each element $|v\rangle$ in the search space, as a vector $|x, y, z, a_1, a_2, b_1, b_2\rangle$. Of course, there are requirements for an element $|v\rangle$ to be a solution. Firstly, in the Boolean domain, variables x, y, z have to satisfy the following condition:

$$F_B(x, y, z) = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) = 1 \qquad (1)$$

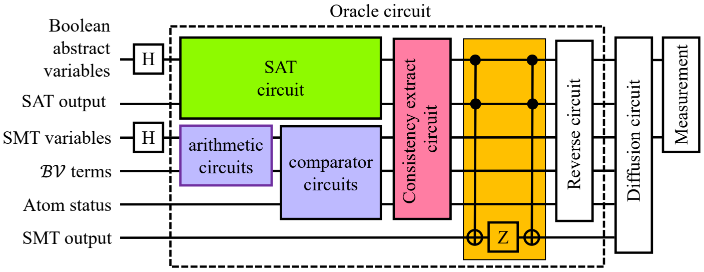


Fig. 2. Quantum circuit for our $\mathcal{BV}$ SMT solver

Secondly, in the domain of the bit-vector theory, variables a_1, a_2, b_1, b_2 need to be consistent with their Boolean counterparts. Thus, they have to satisfy the following conditions:

$$(x = 1 \iff a_1a_2 > b_1b_2) \tag{2}$$
$$(y = 1 \iff a_1a_2 < b_1b_2) \tag{3}$$
$$(z = 1 \iff a_1a_2 = b_1b_2) \tag{4}$$

Here comes the interesting part. If we can construct an oracle Ψ, takeing care of the four conditions simultaneously, the diffuser can then proceed to extract the solution. That is, the functionality of our oracle Ψ performs as follows:

$$\Psi(|v\rangle) = \begin{cases} -1 \cdot |v\rangle & \text{, if} \qquad (1) \wedge (2) \wedge (3) \wedge (4) \\ |v\rangle & \text{, otherwise} \end{cases}$$

If an input vector $|v\rangle$ is a solution, our oracle adds a "-1" phase to it; otherwise, $|v\rangle$ is not changed. Since $|v\rangle$ can be placed in superposition, representing every possible input, our oracle is able to check all the possible inputs in parallel and further identify all the solutions to F in one shot. Then, the diffuser recognizes those inputs with the "-1" phase and tries to maximize their probability being measured when the measurement operation is performed. With the proposed approach, our oracle is able to identify all the 16 solutions to formula F, among the 128 inputs, in one shot.

Fig. 2 shows the structure of our oracle design and summarizes our contributions. There are four components: (1) **SAT Circuit** (c.f. Section II-A) determines the solutions for the Boolean domain. (2) **Theory Circuit** (c.f. Section II-B) determines the truth value of predicates in the theory domain. (3) **Consistency Extractor** (c.f. Section II-C) identifies those solutions that are consistent in both domains. (4) **Solution Inverser** (c.f. Section II-D) inverses the solutions to the SMT formula by adding a "-1" phase to them for the diffuser's further processing. We have also proved the correctness of each component design. All the proofs can be found in [1].

II. Methodology

In this section, we introduce our quantum SMT solver for the $\mathcal{BV}$ theory based on Grover's algorithm. We assume that readers are familiar with the basic concepts of quantum computing and primitive quantum gates such as `X`, `Z`, `H`, `CCNOT`, etc. Fig. 2 shows the overall structure of our approach. The oracle is the key component for marking the solutions so that the difusser knows which elements to maximize their probability when measured. Since the diffuser is standard and independent from the search problem, we will not discuss it here. Instead, we put emphasis on how to construct the oracle in the following sessions.

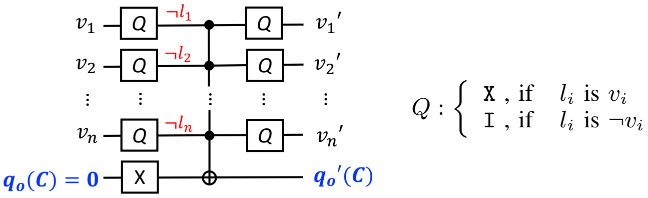

v_1
v_2
$\vdots$
v_n
$q_o(C) = 0$
Q
Q
Q
X
$\neg l_1$
$\neg l_2$
$\neg l_n$
Q
Q
Q
${v_1}'$
${v_2}'$
${v_n}'$
$q_o{}'(C)$
$Q : \begin{cases} \texttt{X} \text{ , if } & l_i \text{ is } v_i \\ \texttt{I} \text{ , if } & l_i \text{ is } \neg v_i \end{cases}$

Fig. 3. Quantum circuit for a clause $C : l_1 \vee l_2 \vee \cdots \vee l_n$

A. SAT Circuit

Fernandes et al. [11] showed an example of constructing the oracle circuit for a 3-SAT problem and solved the problem based on Grover's algorithm. However, [11] did not provide a systematic way to construct the oracle circuit. Here, we propose a method to constructively generate the oracle circuit for any arbitrary formula in conjunctive normal form (CNF). Consider the syntax for an arbitrary CNF formula:

$$\begin{array}{lcl} F & \simeq & C_1 \wedge C_2 \wedge \cdots \wedge C_m \\ C & \simeq & l_1 \vee l_2 \vee \cdots \vee l_n \\ l & \simeq & v \mid \neg v \end{array}$$

A formula F is a conjunction of m clauses $C_1, C_2, \ldots, C_m$ where $m \in \mathbb{N}$. Each clause C_i is a disjunction of n literals $l_1, l_2, \ldots, l_n$ where $n \in \mathbb{N}$ and $i \in \{1, 2, \ldots, m\}$. Notice that the value of n could be different for different clauses. A literal l_j could be a Boolean variable v and called a *positive* literal, or the negation of a Boolean variable $\neg v$ and called a *negative* literal, where $j \in \{1, 2 \ldots, n\}$. A Boolean formula F, over a set of Boolean variables V, is a function $F : \{0,1\}^{|V|} \mapsto \{0,1\}$ mapping an input vector $\vec{v} \in \{0,1\}^{|V|}$ to true/false, where $|V|$ denotes the cardinality of V. A formula F is *satisfiable* if there exists some $\vec{v}$ such that $F(\vec{v}) = 1$, which we call a *solution* to F. A formula F is *unsatisfiable* if it does not have any solution.

Fig. 3 shows how to construct a quantum circuit for a clause $C : l_1 \vee l_2 \vee \cdots \vee l_n$. Notice that the Q gate depends on each literal l_i. If l_i is a negative literal $\neg v_i$, Q would be the `I` gate, i.e., the identity gate; otherwise, Q would be the `X` gate, also called the `NOT` gate. The qubit $q_o(C)$ is the output bit for the truth value of C.

Theorem 1. [Clause Correctness]

1) *$q'_o(C) = 1$ if and only if clause C is true, and*
2) *$v'_i = v_i$ for all $i \in \{1, 2, \ldots, n\}$.*

Fig. 4 shows how to construct a quantum circuit for a given arbitrary formula $F : C_1 \wedge C_2 \wedge \cdots \wedge C_m$. It is required to construct the quantum circuits for all the clauses $C_1, C_2, \cdots, C_m$ first, which are then conjuncted by a `CCNOT` gate with m control qubits. The qubit $q_o(F)$ is the output bit for the truth value of the formula F.

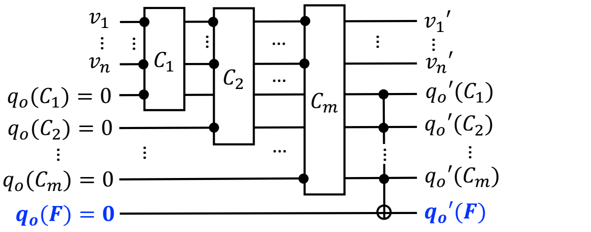


Fig. 4. Quantum circuit construction for formula $F : C_1 \wedge C_2 \wedge \cdots \wedge C_m$

$$\begin{array}{lcl} E & \simeq & \tilde{a} \mid \mathcal{C} \mid E_1 \odot E_2 \quad , \quad \odot \text{ is an arithmetic operator} \\ atom & \simeq & E_1 \rhd E_2 \qquad , \quad \rhd \in \{<,>,=,\geq,\leq,\neq\} \end{array}$$

Fig. 5. Syntax of $\mathcal{BV}$ atoms

Theorem 2. [Formula Correctness]
$q_o'(F) = 1$ if and only if formula F is true.

B. Theory Circuit

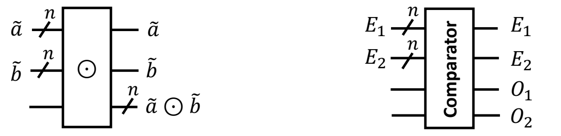
Now, we illustrate how to construct the theory circuit for atoms consisting of arithmetic and comparison operations. The syntax of $\mathcal{BV}$ atoms is given in Fig. 5. An expression E could be a variable $\tilde{a}$ with n bits, a constant $\mathcal{C}$, or the result of an arithmetic operation between two expressions E_1 and E_2. Notice that once the data width (n bits) is determined, it is fixed for all expressions. An atom is composed of two expressions and one comparison operator $\rhd$ in between, where $\rhd \in \{<,>,=,\geq,\leq,\neq\}$. The arithmetic operator includes word-concatenation, modulo-sum, modulo-multiplication, bit-wise operation, shift operation, etc.

Constructing the circuit for a variable $\tilde{a}$ or a constant $\mathcal{C}$ is trivial, which is omitted here. The rest is the case of $\tilde{a} \odot \tilde{b}$. Here, we do not list all the arithmetic operations, as there are many. They can either be constructed based on primitive quantum gates or were developed in related works (c.f. Section IV). Instead, we abstract the circuit for arithmetic operations as the general one, shown in Fig. 6 (a), for easy illustration. Depending on different arithmetic operations, one can obtain their corresponding quantum circuits constructively in a bottom-up manner.

For each atom of the form $E_1 \rhd E_2$, we adopt the comparator [19] to compare E_1 and E_2. The structure of the comparator is shown in Fig. 6 (b). Notice that the output of an arithmetic circuit (in the form of $\tilde{a} \odot \tilde{b}$) could be an input of a comparator circuit. The two output bits O_1 and O_2 indicate the relation between E_1 and E_2, as follows.

$$(O_1, O_2) = \begin{cases} (0,1) & \text{if} \quad E_1 < E_2 \\ (1,0) & \text{if} \quad E_1 > E_2 \\ (0,0) & \text{if} \quad E_1 = E_2 \end{cases}$$

Based on the two output bits O_1 and O_2, one can construct the corresponding atom for the three cases, as shown in Fig. 7 (a)–(c).



(a) Arithmetic Circuit (b) Comparator Circuit

Fig. 6. Theory Circuit

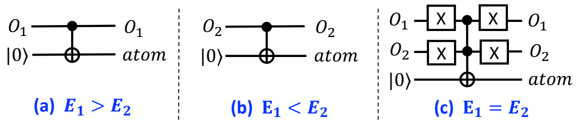


Fig. 7. Quantum circuit construction for atoms

Theorem 3. [Atom correctness]
For each atom $E_1 \rhd E_2$ and its corresponding atom bit, we have $atom = 1$ if and only if $(E_1 \rhd E_2) = 1$.

C. Consistency Extractor

The task of *consistency extractor* is to extract those assignments that are consistent in the Boolean domain as well as the theory domain. That is, given an $atom_i$ with its Boolean abstract variable v_{B_i}, we want to make sure that $v'_{B_i} = 1$ if and only if $atom_i$ and v_{B_i} are logically equivalent.

As shown in Fig. 8 (a), consistency extractor is composed of a `CNOT` gate and a `X` gate. The input $atom_i$ serves as the control bit of the `CNOT` gate, while the other input v_{B_i} serves as the target bit, followed by a `X` gate to flip its result.

Theorem 4. [Correctness of Consistency Extractor]
$v'_{Bi} = 1$ if and only if $v_{B_i} \equiv atom_i$.

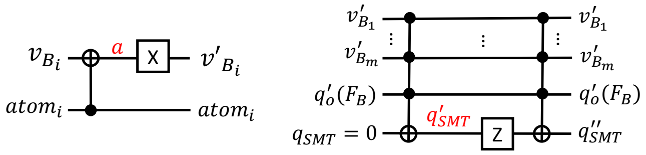
D. Solution Inverter

Solution inverter is the key component of the oracle. It marks the solutions to the SMT formula by inversing them, i.e., giving them a "-1" phase such that the diffuser knows which elements to increase their probability being measured when the quantum measurement operation is performed.

The circuit for solution inverter, as shown in Fig. 8 (b), is composed of two $(m+1)$-`CNOT` gates and a `Z` gate, where m is the number of Boolean abstract variables.

Theorem 5. [Correctness of Solution Inverter]

1) *$q'_{\text{SMT}} = 1$ if and only if $q'_o(F_B) = 1$ and $v_{B_i} \equiv atom_i$, $\forall i \in \{1, 2, \ldots, m\}$.*
2) *All the solutions are added a "-1" phase.*



(a) Consistency Extractor (b) Solution Inverter

Fig. 8. Consistency Extractor

E. Reverse Circuit

The last part of the oracle is the *reverse circuit*, which is constructed by the reversed circuits corresponding to the aforementioned components. That is, once the layout of the quantum circuit consisting of the first three colored components in Fig. 2, namely (1) SAT circuit, (2) theory circuit, and (3) consistency extractor, is constructed (let us call it G), the reverse circuit would be the exact symmetric layout of G.

The purpose of the reverse circuit is to restore the qubits back to their original states, following the reversible nature of quantum computing. In addition, Grover's algorithm may take several iterations to amplify the probability of solutions. Thus, all the ancilla bits need to be restored to be used for the following iterations as well. All the value-restoring actions are responsible by the reverse circuit.

III. Evaluation

To experimentally evaluate our approach, we applied our quantum SMT solver on six complicated formulas. Due to page limit, we will not go through each of them. Instead, we select one formula to illustrate the experiment process. Consider formula $\mathcal{F}_1$, whose Boolean abstract formula is $\mathcal{F}_B : (x \vee y \vee z) \wedge (x \vee \neg y \vee z)$ with atom $x : (a + b < a \oplus b)$, $y : (a + b > a \oplus b)$, and $z : (a + b = 1)$, where variables a, b, c are all 2-bit vectors; '$+$' is the modulo-sum operation and '$\oplus$' is the exclusive-or operation. The quantum circuit was implemented and simulated in Qiskit [4] and SliQsim [9], [22]. The simulation result was obtained by performing a certain number, $\mathcal{N}$, of Grover iterations as one shot, repeated for $1,024$ shots to get $1,024$ measurements. One may wonder how to determine the value of $\mathcal{N}$, for the required number of Grover iterations?

To apply Grover's algorithm for a search problem, the ratio of the number of solutions divided by the size of the search space) needs to be known a priori such that one can obtain the best number of Grover iterations to be applied to get the maximum probability of solutions being measured. Interested readers are referred to the appendix (Section B) of the longer version [1].This ratio can be obtained by quantum counting [7]. However, it requires additional quibts, and the complexity is much higher than Grover's algorithm itself. Thus, we did not use it in our experiments. Instead, we start at one iteration and increase the number of iterations until the measurement probability distribution degrades.

The detail about each formula can be found in the appendix (Section A) of the longer version [1].Table I summarizes all the experimental results. The column "V-Size" indicates the width of the variables, column "signed?" indicates whether the formula is for unsigned or singed operations, column "#Qubits" indicates the total number of qubits required to solve the formula, column "#Iter" shows the number of Grover's iteration required, and column "Sol-Prob" shows the probability of the solutions being meacured. One can observe that the probabilities of the solutions being measured are added up extremely closed to 100%, which experimentally confirms the correctness of our approach.

TABLE I
Experimental Results

$\mathcal{F}$	**V-Size**	**Signed?**	**#Qubits**	**#Iter**	**Sol-Prob**
$\mathcal{F}_1$	2-bit	No	28	3	99.80 %
$\mathcal{F}_2$	3-bit	Yes	34	7	99.67 %
$\mathcal{F}_3$	5-bit	No	62	29	99.93 %
$\mathcal{F}_4$	5-bit	Yes	84	201	99.99 %
$\mathcal{F}_5$	3-bit	No	132	15	99.85 %

IV. Related Works

A. SAT circuit

SAT is NP-complete and has motivated hardware acceleration efforts. Leveraging quantum superposition, Alberto et al. [16] proposed three SAT approaches (quantum Fredkin circuit, quantum register machine, quantum P system) that evaluate all inputs at once, but scarce satisfying assignments make measurement difficult and a non-unitary operator [16] or chaotic dynamics [18] are impractical. Grover's search offers amplitude amplification for satisfiable assignments; Fernandes et al. [11] outline an oracle construction for 3-SAT.

B. Quantum arithmetic and comparator circuit

For arithmetic, Chakrabarti et al. [8] optimize the VBE ripple-carry adder [23] by reorganizing carry logic and replacing the second-stage sum/carry chain with `CNOT`s; we adopt this adder for modulo-sum. Kotiyal et al. [15] propose a multiplier that cutting ancilla by 60.34% and garbage by 52.27% via a binary-tree partial-product scheme and the ancilla-free adder of Takahashi et al. [21]. They also present a bidirectional barrel shifter [14] using `CSWAP`-based 2:1 MUXes with `CNOT` fan-out. For comparison, Oliveira et al. [19] design a quantum comparator that uses fewer resources and more parallelism than subtraction-based NKO [17].

C. Relation to Vinod et al. (2021)

GM Vinod et al. [24] develop a Grover-based solver for integer constraints. Our work differs in that: (i) [24] supports only comparison operators, whereas we support bit-vector arithmetic $(+, -, \times, \div, \oplus)$; (ii) [24] provides examples but no systematic circuit-construction method or correctness proofs, while we provide both; (iii) we separate propositional logic from bit-vector theory, facilitating extensions to other theories, whereas [24] focuses on non-arithmetic integer constraints.

V. Conclusion

In this paper, we proposed a quantum SMT solver for the bit vector theory, based on Grover's algorithm. Our proposed approach not only provides a theoretically quadratic speedup compared to the traditional method, but also has the capability of discovering all solutions. Since solving SMT problems for other theories also consists of two parts (namely SAT and theory solving), our proposed oracle construction is general to be extended to other theories, which is our future work.

References

[1] Smt solver for the question mentioned in article. https://anonymous.4open.science/r/Longer-Version-of-A-Quantum-SMT-Solver-for-Bit-Vector-Theory--1B7D.

[2] Alif Ahmed, Farimah Farahmandi, and Prabhat Mishra. Directed test generation using concolic testing on rtl models. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1538–1543, 2018.

[3] Elvira Albert, Shelly Grossman, Noam Rinetzky, Clara Rodríguez-Núñez, Albert Rubio, and Mooly Sagiv. Taming callbacks for smart contract modularity. *Proc. ACM Program. Lang.*, 4(OOPSLA), nov 2020.

[4] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, et al. Qiskit: An Open-source Framework for Quantum Computing, January 2019.

[5] Leonardo Alt and Christian Reitwiessner. Smt-based verification of solidity smart contracts. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice*, pages 376–388, Cham, 2018. Springer International Publishing.

[6] Guy Amir, Haoze Wu, Clark Barrett, and Guy Katz. An smt-based approach for verifying binarized neural networks. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 203–222, Cham, 2021. Springer International Publishing.

[7] Gilles Brassard, Peter Hoyer, and Alain Tapp. Quantum Counting. *arXiv e-prints*, pages quant–ph/9805082, May 1998.

[8] Amlan Chakrabarti and Susmita Sur-Kolay. Designing quantum adder circuits and evaluating their error performance. In *2008 International Conference on Electronic Design*, pages 1–6, 2008.

[9] Tian-Fu Chen and Jie-Hong R. Jiang. Sliqsim: A quantum circuit simulator and solver for probability and statistics queries. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 129–138, 2025.

[10] Nicole Fern, Ismail San, and Kwang-Ting Tim Cheng. Detecting hardware trojans in unspecified functionality through solving satisfiability problems. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 598–504, 2017.

[11] Diogo Fernandes, Carla Silva, and Inês Dutra. Using grover's search quantum algorithm to solve boolean satisfiability problems, part 2. *XRDS*, 26(2):68–71, nov 2019.

[12] Lov K. Grover. A fast quantum mechanical algorithm for database search. *arXiv e-prints*, pages quant–ph/9605043, May 1996.

[13] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. *arXiv e-prints*, page arXiv:1702.01135, February 2017.

[14] Saurabh Kotiyal, Himanshu Thapliyal, and Nagarajan Ranganathan. Design of a reversible bidirectional barrel shifter. In *2011 11th IEEE International Conference on Nanotechnology*, pages 463–468, 2011.

[15] Saurabh Kotiyal, Himanshu Thapliyal, and Nagarajan Ranganathan. Circuit for reversible quantum multiplier based on binary tree optimizing ancilla and garbage bits. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, pages 545–550, 2014.

[16] Alberto Leporati and Sara Felloni. Three "quantum" algorithms to solve 3-sat. *Theoretical Computer Science*, 372(2):218–241, 2007. Membrane Computing.

[17] A. L. Nascimento, L. A. B. Kowada, and W. R. de Oliveira. A reversible ula. In *Proceedings of the First Workshop-school in Quantum Information and Computation (WECIQ 2006)*, Brazil, 2006.

[18] Masanori Ohya and Igor V. Volovich. New quantum algorithm for studying np-complete problems. *Reports on Mathematical Physics*, 52(1):25–33, 2003.

[19] David Sena Oliveira and Rubens Viana Ramos. Quantum bit string comparator: circuits and applications. *Quantum Comput. Comput*, 7(1):17–26, 2007.

[20] Roberto Sebastiani. Lazy satisfiability modulo theories. *Journal of Satisfiability, Boolean Modeling and Computation*, 3, 12 2007.

[21] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. Quantum addition circuits and unbounded fan-out. *Quantum Info. Comput.*, 10(9):872–890, sep 2010.

[22] Yuan-Hung Tsai, Jie-Hong R. Jiang, and Chiao-Shan Jhang. Bit-slicing the hilbert space: Scaling up accurate quantum circuit simulation. In *Design Automation Conference (DAC)*, pages 439–444, 2021.

[23] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 54:147–153, Jul 1996.

[24] Gayathree M Vinod and Anil Shaji. Finding solutions to the integer case constraint satisfiability problem using grover's algorithm. *IEEE Transactions on Quantum Engineering*, 2:1–13, 2021.

APPENDIX

PROOFS FOR SECTION II (METHODOLOGY)

Theorem 1. [Clause Correctness]

1) $q'_o(C) = 1$ *if and only if clause C is true, and*
2) $v'_i = v_i$ *for all* $i \in \{1, 2, \dots, n\}$.

Proof. Given an arbitrary clause $C : l_1 \vee l_2 \vee \cdots \vee l_n$, if l_i is a negative literal $\neg v_i$, Q would be the I gate; otherwise, Q would be the X gate. Thus, we have $Q(v_i) = \neg l_i$, as the red notations in Fig. 3.

1) $q'_o(C) = 0$ if and only if $(\neg l_1 = 1) \wedge \cdots \wedge (\neg l_n = 1)$. If we apply negation on both sides, we have $q'_o(C) = 1$ if and only if $\neg(\neg l_1 = 1) \vee \cdots \vee \neg(\neg l_n = 1)$. That is, $q'_o(C) = 1$ if and only if $(l_1 = 1) \vee \cdots \vee (l_n = 1)$, i.e., clause C is true.
2) Because Q is either X or I gate, we have $QQ = I$. Thus, $v'_i = Q(\neg l_i) = Q(Q(v_i)) = I(v_i) = v_i$.

□

Theorem 2. [Formula Correctness]
$q'_o(F) = 1$ *if and only if formula F is true.*

Proof. $q'_o(F) = 1$ if and only if $q'_o(C_i) = 1$ for all $i \in \{1, 2, \dots, m\}$. Based on Theorem 1, $q'_o(C_i) = 1$ if and only if clause C_i is true. Thus, we can conclude that $q'_o(F) = 1$ if and only if formula F is true. □

Theorem 3. [Atom correctness]
For each atom $E_1 \rhd E_2$ and its corresponding atom bit, we have $atom = 1$ if and only if $(E_1 \rhd E_2) = 1$.

Proof. The proof is straightforward by examining the $atom$ output for each case based on the truth table of primitive quantum gates. □

Theorem 4. [Correctness of Consistency Extractor]
$v'_{Bi} = 1$ *if and only if* $v_{B_i} \equiv\ atom_i$.

Proof. Let a be the result of v_{Bi} after applying the CNOT gate, as marked in red in Fig. 8.

$$\begin{aligned} v'_{Bi} = 1 \quad &\iff \quad a = 0 \\ &\iff \quad (v_{Bi} \oplus atom_i) = 0 \\ &\iff \quad (v_{Bi} = 0 \wedge atom_i = 0)\ \vee \\ &\qquad\quad (v_{Bi} = 1 \wedge atom_i = 1) \\ &\iff \quad v_{B_i}\ \equiv\ atom_i \end{aligned}$$

□

Theorem 5. [Correctness of Solution Inverter]

1) $q'_{\texttt{SMT}} = 1$ *if and only if* $q'_o(F_B) = 1$ *and* $v_{B_i} \equiv atom_i$, $\forall i \in \{1, 2, \dots, m\}$.
2) *All the solutions are added a "-1" phase.*

Proof. Let $q'_{\texttt{SMT}}$ be the value of the $q_{\texttt{SMT}}$ bit after applying the first $(m+1)$-CNOT gate, as marked in red in Fig. 8 (b). $q_{\texttt{SMT}}$ is initialized as 0, and it is the target bit of the $(m+1)$-CNOT gate. Because v'_{B_i} is one of the m controlled bits, we know that $q'_{SMT} = 1$ if and only if $(v'_{B_1} \wedge v'_{B_2} \wedge ... \wedge v'_{B_m} \wedge q'_o(F_B)) = 1$. By Theorem 4, we can conclude that condition (1) holds.

To prove condition (2), let us consider the state of the quantum circuit. Let $|v'_{B_1}, \dots, v'_{B_m}, \dots, q'_o(F_B), q'_{\texttt{SMT}}\rangle$ be the quantum state after applying the first $(m+1)$-CNOT gate. As the value of $q'_{\texttt{SMT}}$ could be 0 or 1, the system space S can be split into two disjoint sets S_0 and S_1. That is, $S = S_0 \cup S_1$, where $S_0 = \{|v'_{B_1}, \dots, v'_{B_m}, \dots, q'_o(F_B), 0\rangle\}$ and $S_1 = \{|v'_{B_1}, \dots, v'_{B_m}, \dots, q'_o(F_B), 1\rangle\}$. By condition (1) we just proved, S_1 is exactly the set of all solutions to the SMT problem. After that, a Z gate is applied on the $q'_{\texttt{SMT}}$ bit. Since $\texttt{Z}(|0\rangle) = |0\rangle$ and $\texttt{Z}(|1\rangle) = -1|1\rangle$, the Z gate will give a "-1" phase for each element in S_1, i.e., all solutions are added a "-1" phase. □

A. Experiment Details

In this section, we present our quantum circuit design for solving multiple $\mathcal{BV}$ SMT formulas $\mathcal{F}$ (Table II), where the Boolean abstract variables are x, y, z, while a, b, c are the SMT bit-vector variables. We use the following operators: "$+$" denotes modulo addition; $\text{carry}(\cdot)$ denotes the addition carry-out; "$-$" denotes modulo subtraction; "$\oplus$" denotes bit-wise exclusive-or; "$|$" denotes bit-wise or; "$\times$" denotes modulo multiplication; and $\text{sign}(\cdot)$ returns the sign bit of a bit-vector.

Using $\mathcal{F}_1$ as a example (see Fig. 2), we first apply Hadamard gates to the input registers to create a superposition. The oracle consists of (i) a SAT circuit synthesized from the Boolean abstract formula, (ii) arithmetic subcircuits implementing modulo addition and bit-wise exclusive-or, and (iii) two comparators that compare $(a + b)$ with $(a \oplus b)$ and with the constant 1. A consistency-extraction stage ties the SAT and theory parts; A solution inverter flips the phase of satisfying states; and uncomputation completes the oracle. We then apply the diffusion operator, repeat 3 Grover iterations, and perform measurement. The iteration count can be obtained via quantum counting [BHT98]; alternatively, start at one iteration and increase until the measurement probability distribution degrades.

The simulation results, based on 1,024 shots, are summarized in Table III. Six highest-probability bit strings map—via our encoding—to assignments of x, y, z, a, b and exactly cover all solutions of $\mathcal{F}_1$. Averaged over 100 independent runs, the solution measurement rate is $\mathbf{99.80}\%$. To demonstrate generality, we solved multiple $\mathcal{BV}$ SMT formulas $\mathcal{F}$:$\mathcal{F}_1$–$\mathcal{F}_2$ were simulated on Qiskit [AAB$^+$19] and $\mathcal{F}_3$–$\mathcal{F}_5$ on SliQsim [CJ25], [TJJ21]. The implementation can be found in [SMT]. Experimentally, this also confirms the correctness of our approach.

B. Grover's Algorithm

Grover's algorithm [Gro96] is one of the most famous quantum algorithms. It is used to solve the searching problem for finding one target element in a disordered database with N elements. Due to the characteristic of parallel computation in quantum systems, Grover's algorithm takes $O(\sqrt{N})$ operations to find the target element, which is a quadratic speed up compared with classical methods requiring $O(N)$ operations. Grover's algorithm is widely used in many applications, such as cryptography [GLRS16], pattern matching [TNSY22], etc.

TABLE II
$\mathcal{F}$ SET FOR TESTING

$\mathcal{F}$	**Boolean abstract formula**	x	y	z	**bit-width of** a, b, c
$\mathcal{F}_1$	$(x \vee y \vee z) \wedge (x \vee \neg y \vee z)$	$a + b < a \oplus b$	$a + b > a \oplus b$	$a + b = 1$	2-bit unsigned
$\mathcal{F}_2$	$x \wedge (y \vee \neg z) \wedge (\neg y \vee z)$	$a + b = 0$	$sign(a) \oplus sign(b)$	$sign(a - b)$	3-bit signed
$\mathcal{F}_3$	$(x \vee y) \wedge (\neg x \vee z) \wedge (y \vee \neg z)$	$a \mid b = 18$	$a < b$	$carry(a + b)$	5-bit unsigned
$\mathcal{F}_4$	$(x \vee \neg z) \wedge (\neg x \vee z) \wedge y$	$sign(a) \oplus sign(b) \oplus sign(c)$	$a \mid b \mid c = -16$	$a - b + c > a \oplus c$	5-bit signed
$\mathcal{F}_5$	$(\neg x \vee y) \wedge (x \vee z) \wedge (\neg y \vee \neg z)$	$b \times c + a > a \mid b \mid c$	$a \times c > a \oplus b \oplus c$	$a \times b \times c = 5$	3-bit unsigned

TABLE III
SOLUTIONS OF THE $\mathcal{F}$ OBTAINED BY QUANTUM CIRCUIT, AND THE INFORMATION OF THE QUANTUM CIRCUIT

$\mathcal{F}$	**#Qubits**	**Iter.**	**Format**	**High probability output of the simulation result**	**P(measuring)**
$\mathcal{F}_1$	28	3	$\{xyz, b, a\}$	$\{100, 3, 1\}, \{100, 1, 3\}, \{001, 3, 2\}, \{001, 2, 3\}, \{001, 1, 0\}, \{001, 0, 1\}$	99.80 %
$\mathcal{F}_2$	34	7	$\{xyz, b, a\}$	$\{111, -3, 3\}, \{111, -2, 2\}, \{111, 2, -2\}, \{111, 1, -1\}, \{010, -4, -4\}, \{010, 0, 0\}$	99.67 %
$\mathcal{F}_3$	62	29	$\{xyz, b, a\}$	$\{100, 2, 18\}, \{100, 2, 16\}, \{100, 0, 18\}, \{101, 16, 18\}, \{111, 18, 16\}, \{101, 18, 18\}$	99.93 %
$\mathcal{F}_4$	84	201	$\{xyz, c, b, a\}$	$\{111, 0, 0, -16\}, \{010, -16, -16, 0\}, \{010, 0, -16, -16\}, \{111, -16, 0, 0\}$	99.99 %
$\mathcal{F}_5$	132	15	$\{xyz, c, b, a\}$	$\{001, 7, 3, 1\}, \{001, 7, 1, 3\}, \{001, 5, 3, 3\}, \{001, 3, 3, 5\}, \{001, 3, 1, 7\},$ $\{001, 1, 3, 7\}, \{110, 3, 3, 3\}, \{110, 3, 2, 0\}, \{110, 2, 3, 0\}, \{110, 1, 3, 1\}$	99.85 %

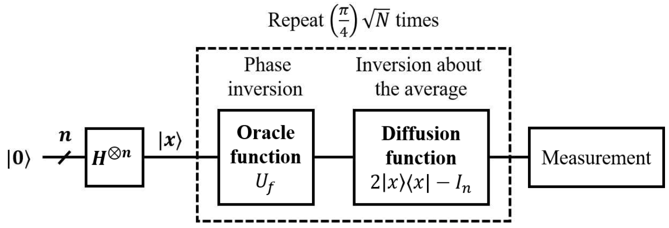

Repeat $\left(\frac{\pi}{4}\right)\sqrt{N}$ times
Phase inversion
Inversion about the average
n
$|0\rangle$
$H^{\otimes n}$
$|x\rangle$
Oracle function U_f
Diffusion function $2|x\rangle\langle x| - I_n$
Measurement

Fig. 9. Grover's algorithm

The overall structure of Grover's algorithm is shown in Fig. 9. The two main operations of it are *phase inversion* and *inversion about the average*, which are handled by the oracle and diffuser, respectively. Initially, the input will be placed in superposition ($|x\rangle$) to evaluate all elements in the database at once. Next, the oracle function U_f considers all the possible inputs and marks the target element by applying phase inversion, i.e., $U_f|x\rangle = (-1)^{f(x)}|x\rangle$, in which $f(x) = 1$ for the target element and $f(x) = 0$ for the others. The oracle is problem-dependent, while the diffuser is not. Thus, designing the correct oracle is the key to apply Grover's algorithm. After the target element is marked, the diffuser applies the *inversion about the mean* operation to amplify the probability of the target element, so that one can obtain the result by measurement. In order to achieve the optimal probability for the target element to be measured, the two operations (called a Grover iteration) need to be repeated $(\pi/4)\sqrt{N}$ iterations .

Grover's algorithm can also be deployed for problems with multiple target elements. In such a case, the number of required iterations becomes $(\pi/4)\sqrt{N/M}$, where M is the number of target elements. In addition, one can measure the correct result only when the number of the target elements is less than that of the nontarget elements (i.e. $M/N < 50\%$) due to the natural property of the algorithm [You08]. Since the number of target elements is usually unknown before the search, there are several ways to solve this issue. The most common one is to apply quantum counting [BHT98] to obtain the (approximate) number of target elements before using Grover's algorithm. After knowing the ratio of M/N, if it is more than 50%, one can double the search space by adding N nontarget elements, so the target elements will always be less than 50% of the search space.

C. Strategies for Operand Preservation in Theory Circuits

There is one concern when applying the arithmetic circuit, that is, whether to keep the original input operands. If the input operand is still necessary for the following procedure, there are two ways to solve this concern. The first approach is to use the reversible nature of quantum computing. If the calculation result can be stored in other qubits, a reversed arithmetic circuit can restore the operand qubits to their original state. The other way is to duplicate the input or the output of the arithmetic circuit before or after its calculation. The `CNOT` gate can be used to implement the duplication by placing the variable at the control qubit and setting the input of the target qubit to 0. Which approach is more appropriate depending on the adopted arithmetic circuit. Take the quantum adder [CSK08] that we adopt and modified in our framework for example, as shown in Fig. 13. Since the carry qubits are necessary during the calculation procedure but become useless after the computation, we can reverse the control and target qubit of the last layer's `CNOT` gates to store the summation result in those carry qubits, and then add `CNOT` gates between two operands to restore the variable (which is variable b in this circuit) back to its original condition.

REFERENCES FOR THE APPENDIX

[AAB$^+$19] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, et al. Qiskit: An Open-source Framework for Quantum Computing, January 2019.

[BHT98] Gilles Brassard, Peter Hoyer, and Alain Tapp. Quantum Counting. *arXiv e-prints*, pages quant–ph/9805082, May 1998.

[CJ25] Tian-Fu Chen and Jie-Hong R. Jiang. Sliqsim: A quantum circuit simulator and solver for probability and statistics queries. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 129–138, 2025.

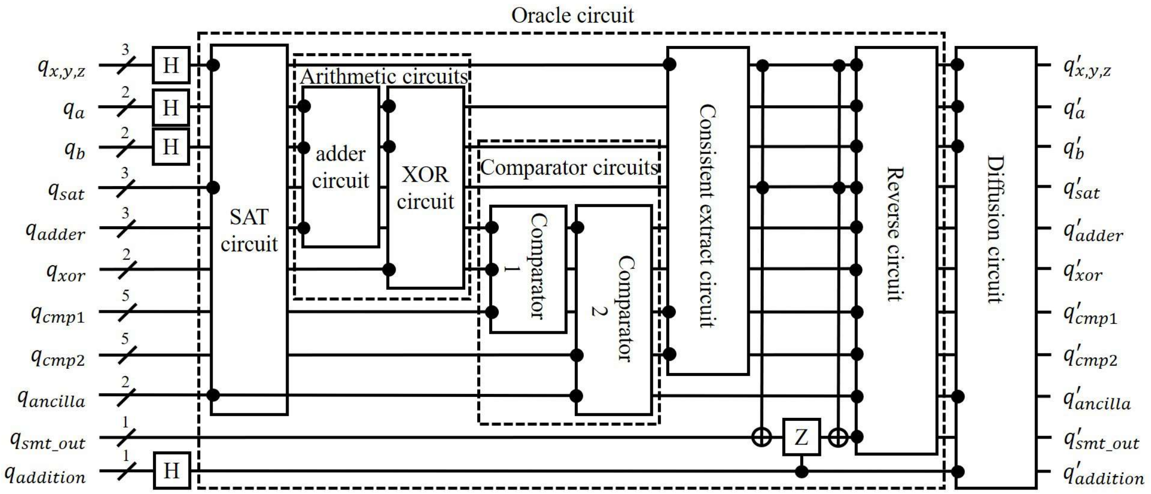


Fig. 10. The quantum circuit of solving formula $\mathcal{F}$.

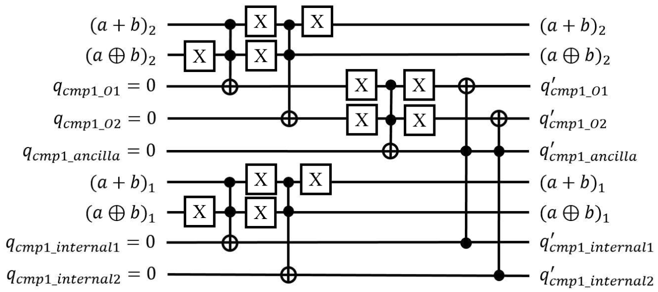


Fig. 11. The comparator 1 circuit of F_{SMT} solving circuit.

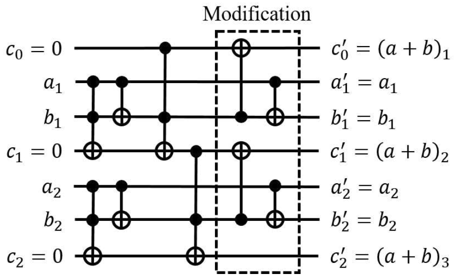


Fig. 13. The circuit of modified 2-bit adder.

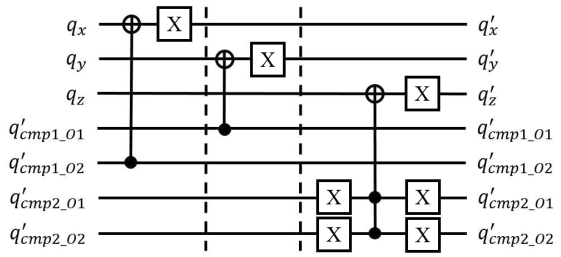


Fig. 12. The consistent extract circuit of F_{SMT} solving circuit.

[CSK08] Amlan Chakrabarti and Susmita Sur-Kolay. Designing quantum adder circuits and evaluating their error performance. In *2008 International Conference on Electronic Design*, pages 1–6, 2008.

[GLRS16] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying grover's algorithm to aes: Quantum resource estimates. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography*, pages 29–43, Cham, 2016. Springer International Publishing.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. *arXiv e-prints*, pages quant–ph/9605043, May 1996.

[SMT] Smt solver for the question mentioned in article. https://anonymous.4open.science/r/Longer-Version-of-A-Quantum-SMT-Solver-for-Bit-Vector-Theory--1B7D.

[TJJ21] Yuan-Hung Tsai, Jie-Hong R. Jiang, and Chiao-Shan Jhang. Bit-slicing the hilbert space: Scaling up accurate quantum circuit simulation. In *Design Automation Conference (DAC)*, pages 439–444, 2021.

[TNSY22] Hiroyuki Tezuka, Kouhei Nakaji, Takahiko Satoh, and Naoki Yamamoto. Grover search revisited: Application to image pattern matching. *Phys. Rev. A*, 105:032440, Mar 2022.

[You08] Ahmed Younes. Strength and Weakness in Grover's Quantum Search Algorithm. *arXiv e-prints*, page arXiv:0811.4481, November 2008.