Automated Synthesis of Quantum Circuits using Symbolic Abstractions and Decision Procedures

Alvaro Velasquez[1], Sumit Kumar Jha[2], Rickard Ewetz[3], and Susmit Jha[4]
[1]*Information Directorate, Air Force Research Laboratory*
[2]*Department of Computer Science, University of Texas at San Antonio*
[3]*Department of Computer Science, University of Central Florida*
[4]*SRI International*
alvaro.velasquez.1@us.af.mil, sumit.jha@utsa.edu, rickard.ewetz@ucf.edu, susmit.jha@sri.com

***Abstract*—Quantum algorithms are notoriously hard to design and require significant human ingenuity and insight. We present a new methodology called Quantum Automated Synthesizer (QUASH) that can automatically synthesize quantum circuits using decision procedures that perform symbolic reasoning for combinatorial search. Our automated synthesis approach constructs finite symbolic abstract models of the quantum gates automatically and discovers a quantum circuit as a composition of quantum gates using these symbolic models. Our key insight is that most current quantum algorithms work on a finite number of classical inputs, and hence, their correctness proof relies only on reasoning about a finite set of quantum states that can be represented using finite symbolic systems. We demonstrate the potential of our approach by automatically synthesizing four quantum circuits and re-discovering the Bernstein-Vazirani quantum algorithm using state-of-the-art decision procedures. Our synthesis approach only requires distinguishing between a finite set of symbolic quantum states; for example, the synthesis of the Bernstein-Vazirani quantum algorithm only requires reasoning about the following qubit states: $|0\rangle$, $|1\rangle$, $-i|0\rangle$, $i|1\rangle$, $|+\rangle$, $|-\rangle$, $e^{i\pi/2}|1\rangle$, $e^{i\pi/4}|1\rangle$ and a remaining symbolic state representing all other possible quantum states. Our approach leverages decision procedures and theorem provers to assist in the discovery of new quantum algorithms and is a step towards the automation of quantum algorithm design.**

I. Introduction

Quantum computers with 5 to 15 qubits are now available to the public through the IBM Q platform. IBM's 53-qubit system has recently been publicly announced, and a 53-qubit quantum chip called Sycamore has been introduced by Google. Hence, the era of the Noisy Intermediate-Scale Quantum (NISQ) system with as many as 50 qubits in restricted quantum topologies is already a reality. However, despite these leaps in our ability to create NISQ systems, our ability to develop new quantum algorithms to program these systems to solve new problems is very limited and dependent on deep human insights and ingenuity. A significant barrier in creating new quantum algorithms is the non-intuitive mapping of a computational problem over quantum operations; hence, human experts face an inherent difficulty in reasoning about quantum systems. Our goal is to develop a design assistant, Quantum Automated Synthesizer (QUASH), that can bridge this intuition gap using decision procedures, and synthesize simple quantum algorithms as a composition of provided quantum gates.

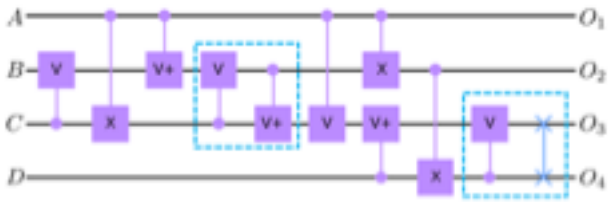

Fig. 1. Synthesized TSG gate using QUASH with gates $X, V, V^+, SWAP$, where $O_1 = A, O_2 = (\neg A \wedge \neg C) \oplus \neg B, O_3 = ((\neg A \wedge \neg C) \oplus \neg B) \oplus D, O_4 = (((\neg A \wedge \neg C) \oplus \neg B) \wedge D) \oplus ((A \wedge B) \oplus C)$ are the outputs of the circuit. Blue squares denote quantum gates that can be merged into a single operation and thus have a quantum cost of 1.

Algorithms and machine learning systems have been used to map known quantum algorithms onto quantum computers with restricted topologies and noisy qubits to maximize the performance of quantum circuits and reduce the error rate in the operation of the quantum circuit. These investigations have successfully shown the potential of algorithms and machine learning methods in effectively compiling known quantum circuits to a specific implementation of a quantum computer.

In this paper, we pursue a different design automation objective: we seek to automatically discover quantum algorithms or circuits from the problem specification provided as examples of the expected input/output pairs. We focus on quantum circuits that have classical (non-quantum) inputs and outputs, that is, the problem being solved by the quantum algorithm is over classical bits. We make the following new contributions:

1) For quantum circuits with classical inputs/outputs, even though the intermediate states are quantum, we create an automated synthesis methodology that uses decision procedures to explore the space of possible quantum circuit designs using symbolic abstractions of quantum states. These symbolic states are automatically discovered and represent one or more quantum states.
2) We automatically synthesize four different quantum circuits with minimum quantum cost from their input/output specifications using decision procedures. We also automatically re-discover the Bernstein-Vazirani algorithm or circuit using a symbolic abstraction of only 9 symbolic states: one for each of the qubits $|0\rangle$, $|1\rangle$, $-i|0\rangle$, $i|1\rangle$, $|+\rangle$, $|-\rangle$, $e^{i\pi/2}|1\rangle$, $e^{i\pi/4}|1\rangle$, and a symbolic state representing all other possible quantum states.

II. Related Work

A. Quantum Computation and Reasoning

Optimization methods, including decision procedures, have been used to compile high-level quantum programs onto target quantum computing platforms with the goal of improving error resiliency against variability [1] and noise [2]. The mapping of quantum algorithms to quantum computers that respect the topological constraints imposed by specific quantum computers has also been investigated using automated reasoning methods [3], [4]. Formal methods and other logic-based symbolic reasoning systems have been used to reason about the correctness of quantum programs and quantum computing systems [5]–[7].

However, our approach automatically synthesizes circuits from input/output specifications by reasoning over symbolic abstractions of quantum states using decision procedures. We are also the first to automatically re-discover a non-trivial quantum circuit, such as the Bernstein-Vazirani quantum algorithm, using a symbolic abstraction of 9 symbolic states from examples of input/output pairs.

B. Automated Synthesis

The use of symbolic reasoning approaches to synthesis dates back to Church [8]. A rich theory of symbolic reasoning [9], [10] has developed since then, ranging from fully-automated reasoning over restricted fragments of logic [11]–[13] to proof assistants [14]–[16] that can help humans reason over higher-order logic. Decision procedures [17]–[19] have emerged as an effective symbolic reasoning technique that can automate combinatorial search and find models over logical fragments such as those including bitvectors (array of Boolean bits), abstract uninterpreted functions and arithmetic operations. In this paper, we use a state-of-the-art reasoning systems Z3 [20] to implement the QUASH quantum automated synthesizer.

The use of symbolic reasoning and decision procedures in synthesis of traditional hardware and software has also received significant attention. Synthesis of hardware circuits has been explored in literature [21]–[23] and has been adopted in the industry [24], [25], particularly for design optimization. The use of combinatorial methods in automated synthesis of software has also been recently explored [26], [27] to generate non-intuitive code snippets.

However, our approach is the first to apply automated synthesis using symbolic abstractions to quantum algorithms. Even if the inputs and outputs of quantum algorithms are classical bits, the intermediate states can be qubits. The decision procedures cannot reason directly over qubits or vectors with irrational or complex coefficients. Even reasoning over integer arithmetic with multiplication (nonlinearity) is known to be undecidable [28], [29].

We exploit the observation that the *interesting* quantum states in a typical quantum circuit are finitely enumerable, and we can symbolically abstract all quantum states to create a finite set of symbols where each state represents one or more qubits. QUASH uses this insight to automatically synthesize quantum circuits from input/output specifications.

III. Quantum Automated Synthesis (QUASH) using Symbolic Reasoning

Quantum systems perform linear reversible transformations on quantum states composed of qubits, or quantum bits. In principle, a decision procedure should be able to reason directly on this algebraic representation of a quantum circuit and search for a quantum circuit capable of performing a given operation. However, this direct approach is not feasible as existing combinatorial search algorithms cannot search over high-dimensional spaces with irrational/complex values. Such a direct search for the correct quantum algorithm is prohibitively expensive because of the interaction between the search for the correct overall topology of the network representing the quantum circuit and the individual correct quantum gates to be placed in this topology.

A. Symbolic Representation of Quantum Gates

Our approach towards developing the Quantum Automated Synthesizer (QUASH) adopts a symbolic approach towards reasoning about quantum states and quantum gates. Each quantum state is provided a symbol; for example, the quantum state $|0\rangle$ is represented by the symbol α and $|1\rangle$ is represented by the symbol β. Then, for example, we can represent the operation of the X quantum gate on these two inputs by adopting the following symbolic rules:

1) $X(\alpha) = \beta$
2) $X(\beta) = \alpha$
3) $X(\perp) = \perp$, read as bottom (of a lattice).

The third rule essentially says that the X quantum gate maps any state that is not α or β to some state that is not known to be either α or β. This symbolic abstract model of the X gate is not enough to represent its behavior on all inputs. But, as we will demonstrate, it is enough to symbolically represent the execution of quantum gates on a few quantum states for each qubit in order to automatically synthesize interesting quantum algorithms like the Bernstein-Vazirani quantum algorithm.

In our approach, we build the symbolic model of quantum gates automatically by focusing on inputs $|0\rangle$, $|1\rangle$, and then expanding the inputs to include the outputs of the quantum gates on these inputs. Consider the Hadamard H gate. We first build a Symbolic Input/Output (I/O) Model for the H gate using the inputs $|0\rangle$ and $|1\rangle$ represented symbolically as α and β:

H gate	Inputs		
	$\alpha \equiv \|0\rangle$	$\beta \equiv \|1\rangle$	$\perp$
Output	$\perp$	$\perp$	$\perp$

It is clear from this table that this symbolic model of the Hadamard H gate is overly abstract and not useful for analysis as all its outputs are unknown, represented symbolically by $\perp$. This occured as we have no symbolic representation for the output of the Hadamard gate on $|0\rangle$ i.e. for $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. Hence, we create a new symbol γ for this quantum state $|+\rangle$. For similar reasons, we also create a new symbol state for the

quantum state $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. Then, we extend the Symbolic I/O Model for the Hadamard H gate to the following:

H gate	Inputs				
	$\alpha \equiv \|0\rangle$	$\beta \equiv \|1\rangle$	$\gamma \equiv \|+\rangle$	$\delta \equiv \|-\rangle$	$\perp$
Output	γ	δ	α	β	$\perp$

We could have continued creating a larger symbolic Input/output (I/O) Model for the Hadamard quantum gate to build a better abstraction of its behavior. Our key observation in this paper is that a finite symbolic I/O abstraction is sufficient to discover many quantum circuits; for example, an abstraction involving only 9 symbols is sufficient for us to re-discover the Bernstein-Vazirani algorithm using automated synthesis. The symbolic I/O abstractions are automatically constructed from the definition of the quantum gates and the quantum input states that we chose to represent using the symbolic variables.

B. Symbolic Search in QUASH

The symbolic search procedure in QUASH defines a search space of circuits where the desired quantum computation can be performed by at most n gates from a given list of m known quantum gate types. For example, for one execution of our symbolic search for the TSG circuit in Figure 1, we required that we only use the following $m = 4$ gate types: X, V, V^+, and $SWAP$. Further, we required that the circuit contain no more than a total of 11 quantum gates. A variety of other designs can be obtained by varying these parameters.

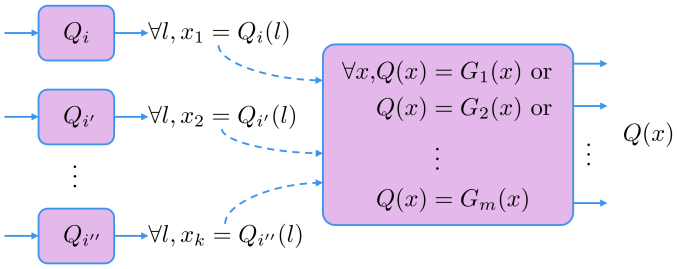

Fig. 2. Illustration of the symbolic search using decision procedures. Each quantum gate Q corresponds to one of the component quantum gates G in the library. The input to each quantum gate comes from the output of some other quantum gate or an input to the quantum circuit itself.

The search procedure begins by encoding the desired circuit as a *component-based design* problem. For the sake of simplicity, we discuss quantum gates with only one input. The complete quantum circuit is composed of n component quantum gates: Q_1, Q_2, $\ldots Q_n$. Each gate can be one of the m different types of quantum gates $G_1, G_2 \ldots G_m$, i.e. the output of each component must agree with one of the gates on all inputs. Formally, we have:

$$\forall x, \big((Q_i(x) = G_1(x)) \text{ or } \ldots \text{ or } (Q_i(x) = G_m(x)) \big)$$

That is, for all values of the input x, the output of the quantum gate Q_i on this input agrees with the output of one the gates in our component library $G_1, G_2 \ldots G_m$ of m gates.

The next set of logical specifications for the symbolic reasoning decision procedure determines the connections between the quantum gates $Q_1, Q_2 \ldots Q_n$. For each input port inp_{Q_i} of the quantum gate Q_i, its value must be derived from either one of the inputs to the quantum circuit $I_1, \ldots, I_k$ or one of the outputs out_{Q_j} of another quantum gate Q_j. Formally,

$$\forall I, \big((inp_{Q_i} = out_{Q_1}(I)) \text{ or } \ldots \text{ or } (inp_{Q_i} = out_{Q_n}(I)) \text{ or } (inp_{Q_i} = I_1) \text{ or } \ldots \text{ or } (inp_{Q_i} = I_k) \big)$$

Informally, the input to a quantum gate Q_i is either one of the inputs to the circuit $I_1, \ldots, I_k$ or the result of one of the other quantum gates Q_j.

IV. Results: Synthesized Quantum Circuits

We study the performance of the QUASH automatic quantum synthesizer of four quantum circuits: TSG [30], HNG [31], PFAG [32], and MKG [33]. These quantum circuits are important components for reversible logic and the implementation of more complex quantum multiplier circuits and quantum ALUs [34], [35] as they enable the simultaneous computation of various logic functions, including full addition. Our approach is agnostic to the choice of these benchmarks and could be readily applied to other quantum circuits whose input/output specifications are available.

We use the proposed QUASH approach to generate state-of-the-art designs for the TSG [30], HNG [31], PFAG [32], and MKG [33] in terms of minimizing the quantum cost.

	Quantum Cost		
Circuit	QUASH	[36]	[32]
HNG (Figure 3)	**6**	**6**	-
PFAG (Figure 4)	**6**	-	8
TSG (Figure 1)	**9**	14	-
MKG (Figure 5)	**10**	13	-

TABLE I
Quantum cost of various arithmetic gates synthesized using QUASH and other approaches.

Our experiments were performed on a system with 32 3.3 GHz Intel cores and 1 TB of memory. The results of our QUASH approach can be seen in Table I and in Figures 3, 4, 1 and 5. Here, quantum cost is the number of 2-qubit gates used in the synthesized circuit. However, it is worth noting that when multiple 2-qubit gates are used in sequence on the same qubits, this has unit cost. This follows from the observation that applying a sequence of unitary transformations on the same qubits is itself a unitary operation and can thus be represented by some unitary matrix on said qubits. Such cases are demarcated in both Figures 1 and 5 by dashed blue squares.

Our circuits are minimal using this circuit topology and using these quantum gates. We show this by asking QUASH to obtain quantum circuits with fewer gates and the decision procedure produces an unsatisfiable response. For example, the circuit in Figure 3 is shown to be optimal as QUASH takes 115.51 seconds to produce an unsatisfiable response when asked to produce a circuit with no more than 5 quantum gates.

Circuit	Performance	
	Time (sec)	Memory (MB)
HNG (Figure 3)	224.50	300.536
PFAG (Figure 4)	193.92	297.272
TSG (Figure 1)	144,326.49	364.255
MKG (Figure 5)	159,473.78	387.840

TABLE II
QUASH PERFORMANCE FOR SUCCESSFUL AUTOMATED SYNTHESIS.

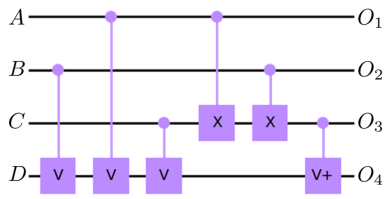

Fig. 3. Our automatically synthesized HNG circuit using our QUASH approach with gates X, V, V^+. Here, A, B, C and D are the inputs, and $O_1 = A, O_2 = B, O_3 = A \oplus B \oplus C, O_4 = ((A \oplus B) \wedge C) \oplus ((A \wedge B) \oplus D)$ are the outputs of the circuit.

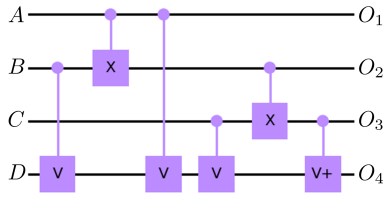

Fig. 4. Synthesized PFAG gate using QUASH with gates X, V, V^+, where $O_1 = A, O_2 = A \oplus B, O_3 = A \oplus B \oplus C, O_4 = ((A \oplus B) \wedge C) \oplus ((A \wedge B) \oplus D)$.

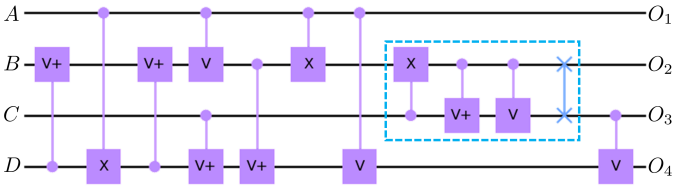

Fig. 5. Synthesized MKG gate using QUASH with gates $X, V, V^+, SWAP$, where $O_1 = A, O_2 = C, O_3 = ((\neg A \wedge \neg D) \oplus \neg B) \oplus C, O_4 = (((\neg A \wedge \neg D) \oplus \neg B) \wedge C) \oplus ((A \wedge B) \oplus D)$.

A. Synthesis of Bernstein-Vazirani Circuit

We applied our algorithmic synthesis approach towards automatically re-discovering the Bernstein-Vazirani quantum algorithm. We selected an over-approximation of the set of quantum gates that are known to be sufficient to implement the Bernstein-Vazirani quantum algorithm: (i) the Hadamard H gate, (ii) the T gate, (iii) the S gate, (iv) the X gate, (v) the Y gate, (vi) the Z gate, (vii) the phase kickback U_f sub-circuit, and (viii) the identity I gate.

We constructed a symbolic I/O model for each of the quantum gates and implemented our component-based search using these symbolic models. The X, Y and Z gates are described using a symbolic state α representing the quantum state $|0\rangle$, another symbolic state β representing the quantum state $|1\rangle$ and a symbolic state $\perp$ representing those quantum states that are not captured by any other symbolic state.

X gate	Inputs		
	$\alpha \equiv \|0\rangle$	$\beta \equiv \|1\rangle$	$\perp$
Output	β	α	$\perp$

Z gate	Inputs		
	α	β	$\perp$
Output	β	$\perp$	$\perp$

The symbolic I/O model of the Hadamard H gate is refined as follows:

H gate	Inputs				
	$\alpha \equiv \|0\rangle$	$\beta \equiv \|1\rangle$	$\gamma \equiv \|+\rangle$	$\delta \equiv \|-\rangle$	$\perp$
Output	γ	δ	α	β	$\perp$
	Inputs				
	$\epsilon \equiv -i\|0\rangle$	$\zeta \equiv i\|1\rangle$	$\eta \equiv e^{i\pi/2}\|1\rangle$	$\theta \equiv e^{i\pi/4}\|1\rangle$	
Output	$\perp$	$\perp$	$\perp$	$\perp$	

The phase kickback circuit U_{s_i} has a symbolic I/O model that maps $\gamma \equiv |+\rangle$ to γ if the particular $s_i = 1$; otherwise, it maps the input symbolic state $\gamma \equiv |+\rangle$ to the symbolic state $\delta \equiv |-\rangle$. In the interest of brevity, we omit the descriptions of the symbolic I/O models of the other quantum gates. But, it should be clear that these symbolic representations can be computed automatically from the definitions of the quantum gates. An example synthesized quantum circuit for this problem can be seen in Figure 6.

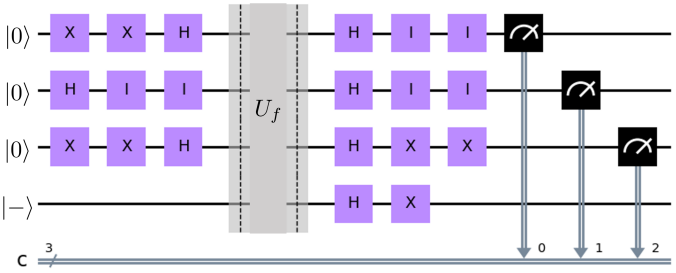

Fig. 6. Synthesized Bernstein-Vazirani quantum circuit.

V. Conclusions and Future Work

This paper is a first step towards the automatic discovery of quantum algorithms using symbolic abstractions and decision procedures. Our success in automatically synthesizing four optimal quantum circuits (see Table I as well as Figures 3, 4, 1 and 5) and an implementation of the Bernstein-Vazirani quantum algorithm relies on a novel symbolic abstract modeling of quantum gates. Symbolic abstractions of quantum gates provide an efficient mapping from continuous high-dimensional tensor representations of quantum circuits to symbolic representations that can be readily interpreted by decision procedures. For future work, we are investigating the use of circuit optimizations to reduce the runtime of the synthesis procedure. Indeed, this would allow the QUASH to return correct sub-optimal circuit designs more quickly such that optimization techniques can then be employed to yield optimal designs.

REFERENCES

[1] S. S. Tannu and M. K. Qureshi, "A case for variability-aware policies for nisq-era quantum computers," *arXiv preprint arXiv:1805.10224*, 2018.

[2] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2019, pp. 1015–1029.

[3] X. Zhang, H. Xiang, T. Xiang, L. Fu, and J. Sang, "An efficient quantum circuits optimizing scheme compared with QISKit," *arXiv preprint arXiv:1807.01703*, 2018.

[4] P. Murali, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Formal constraint-based compilation for noisy intermediate-scale quantum systems," *Microprocessors and Microsystems*, vol. 66, pp. 102–112, 2019.

[5] D. Unruh, "Quantum hoare logic with ghost variables," *arXiv preprint arXiv:1902.00325*, 2019.

[6] Y. Huang and M. Martonosi, "Qdb: from quantum algorithms towards correct quantum programs," *arXiv preprint arXiv:1811.05447*, 2018.

[7] R. Rand, J. Paykin, D.-H. Lee, and S. Zdancewic, "Reqwire: Reasoning about reversible quantum circuits," *arXiv preprint arXiv:1901.10118*, 2019.

[8] C. Alonzo, "Application of recursive arithmetic to the problem of circuit synthesis summaries of talks presented at the Summer Institute for Symbolic Logic Cornell University, 1957, 2nd Edition., Communications Research Division, Institute for Defense Analyses, Princeton, NJ," pp. 3–50, 1960.

[9] J. Buchi and L. Landweber, "Solving sequential conditions by finite-state strategies. Trans. Amer. Math. Soc., 138," 1969.

[10] M. O. Rabin, *Automata on infinite objects and Church's problem*. American Mathematical Soc., 1972, vol. 13.

[11] L. Zhang, C. F. Madigan, M. H. Moskewicz, and S. Malik, "Efficient conflict driven learning in a boolean satisfiability solver," in *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*. IEEE Press, 2001, pp. 279–285.

[12] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking*. Springer, 2018, pp. 305–343.

[13] S. Jha, R. Limaye, and S. A. Seshia, "Beaver: Engineering an efficient smt solver for bit-vector arithmetic," in *International Conference on Computer Aided Verification*. Springer, 2009, pp. 668–674.

[14] S. Owre, J. M. Rushby, and N. Shankar, "Pvs: A prototype verification system," in *International Conference on Automated Deduction*. Springer, 1992, pp. 748–752.

[15] Y. Bertot and P. Castéran, *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[16] L. de Moura, S. Kong, J. Avigad, F. Van Doorn, and J. von Raumer, "The lean theorem prover (system description)," in *International Conference on Automated Deduction*. Springer, 2015, pp. 378–388.

[17] C. W. Barrett, D. L. Dill, and J. R. Levitt, "A decision procedure for bit-vector arithmetic," in *Proceedings 1998 Design and Automation Conference. 35th DAC.(Cat. No. 98CH36175)*. IEEE, 1998, pp. 522–527.

[18] R. E. Bryant, D. Kroening, J. Ouaknine, S. A. Seshia, O. Strichman, and B. Brady, "Deciding bit-vector arithmetic with abstraction," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2007, pp. 358–372.

[19] A. Zeljić, C. M. Wintersteiger, and P. Rümmer, "Deciding bit-vector formulas with mcsat," in *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 2016, pp. 249–266.

[20] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[21] S. Malik, A. R. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Logic verification using binary decision diagrams in a logic synthesis environment," in *[1988] IEEE International Conference on Computer-Aided Design (ICCAD-89) Digest of Technical Papers*. IEEE, 1988, pp. 6–9.

[22] Z. S. Andraus and K. A. Sakallah, "Automatic abstraction and verification of verilog models," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 218–223.

[23] H. Jain, D. Kroening, N. Sharygina, and E. Clarke, "Word level predicate abstraction and refinement for verifying rtl verilog," in *Proceedings. 42nd Design Automation Conference, 2005.* IEEE, 2005, pp. 445–450.

[24] L. Amarú, P. Vuillod, J. Luo, and J. Olson, "Logic optimization and synthesis: Trends and directions in industry," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 1303–1305.

[25] S. Hassoun and T. Sasao, *Logic synthesis and verification*. Springer Science & Business Media, 2012, vol. 654.

[26] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari, "Oracle-guided component-based program synthesis," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010, pp. 215–224.

[27] R. Alur, R. Bodik, G. Juniwal, M. M. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa, "Syntax-guided synthesis," in *2013 Formal Methods in Computer-Aided Design*. IEEE, 2013, pp. 1–8.

[28] M. Davis, "Hilbert's tenth problem is unsolvable," *The American Mathematical Monthly*, vol. 80, no. 3, pp. 233–269, 1973.

[29] P. Beame and V. Liew, "Toward verifying nonlinear integer arithmetic," *Journal of the ACM (JACM)*, vol. 66, no. 3, p. 22, 2019.

[30] H. Thapliyal and M. Srinivas, "A novel reversible tsg gate and its application for designing reversible carry look-ahead and other adder architectures," in *Asia-Pacific Conference on Advances in Computer Systems Architecture*. Springer, 2005, pp. 805–817.

[31] M. Haghparast, S. J. Jassbi, K. Navi, and O. Hashemipour, "Design of a novel reversible multiplier circuit using hng gate in nanotechnology," in *World Appl. Sci. J.* Citeseer, 2008.

[32] M. S. Islam, M. M. Rahman, Z. Begum, and M. Z. Hafiz, "Low cost quantum realization of reversible multiplier circuit," *Information technology journal*, vol. 8, no. 2, pp. 208–213, 2009.

[33] M. Shams, M. Haghparast, and K. Navi, "Novel reversible multiplier circuit in nanotechnology," *World Appl. Sci. J*, vol. 3, no. 5, pp. 806–810, 2008.

[34] H. Thapliyal and M. Srinivas, "Novel reversible tsg gate and its application for designing components of primitive reversible/quantum alu," in *2005 5th International Conference on Information Communications & Signal Processing*. IEEE, 2005, pp. 1425–1429.

[35] M. K. Thomsen, R. Glück, and H. B. Axelsen, "Reversible arithmetic logic unit for quantum arithmetic," *Journal of Physics A: Mathematical and Theoretical*, vol. 43, no. 38, p. 382002, 2010.

[36] M. Haghparast, M. Mohammadi, K. Navi, and M. Eshghi, "Optimized reversible multiplier circuit," *Journal of Circuits, Systems, and Computers*, vol. 18, no. 02, pp. 311–323, 2009.