An Examination of the NCV-$|v_1\rangle$ Quantum Library Based on Minimal Circuits

Arman Allahyari-Abhari Robert Wille Rolf Drechsler
Institute of Computer Science, University of Bremen, Bremen, Germany
Cyber Physical Systems, DFKI GmbH, Bremen, Germany
{abhari, rwille, drechsle}@informatik.uni-bremen.de

***Abstract*—In the recent past, significant effort has been put on the investigation of design methods for quantum circuits. Based on different physical realizations, several gate libraries have been proposed for this purpose. Recently, the so-called NCV-$|v_1\rangle$ library has been introduced in this context. In contrast to established libraries, the NCV-$|v_1\rangle$ library seems to inherit some significant advantages compared to established ones, e.g. with respect to the mapping from reversible circuits or the satisfaction of nearest neighbor constraints. However, all these conclusions have been drawn based on heuristical results. In this work, we perform a more in-depth examination of the NCV-$|v_1\rangle$ library based on minimal circuits. For this purpose, an exact synthesis scheme is proposed which utilizes the power of solvers for Boolean satisfiability. Our examination clearly unveiled that, from a logic design perspective, the NCV-$|v_1\rangle$ library indeed superiors the currently established library.**

I. Introduction

The development of conventional circuit technologies is going to reach physical limits in the near future. Therefore, significant effort is put in the investigation of alternative paths. One of the most promising alternatives are quantum circuits [1]. These circuits rely on qubits rather than conventional (Boolean) bits. Besides the established Boolean states 0 and 1, qubits can be set into superposition, i.e. may assume states which represent 0 and 1 at the same time (each of which with a certain probability). This allows for performing computations with massive parallelism which, in turn, enables to solve certain problems with a polynomial, sometimes even an exponential, speed-up compared to existing (conventional) solutions. The states of such qubits are thereby manipulated by inherently reversible quantum operations.

From a design perspective, synthesis of respective quantum circuit descriptions constitutes an elemental step in the design flow for this technology. Since many well-known quantum applications (e.g. Shor's Algorithm for factorization [2] or Groover's Iteration for database search [3]) rely on variable Boolean components, how to realize arbitrary Boolean functions in quantum logic is an important and heavily considered research area. As additionally various technologies for the physical realization of these circuits are being investigated right now, different gate libraries are assumed for this purpose. The NCV library [4] and the Clifford+*T* library [5] are prominent examples for such libraries which have intensely been considered in the past (see e.g. [4], [6], [7], [8], [9], [10] or [5], respectively).

Besides that, another library – the so-called *NCV-$|v_1\rangle$ library* – recently found attention and provides significant advantages over existing ones. In fact, initial investigations unveiled benefits (1) with respect to the mapping of reversible gates to NCV-$|v_1\rangle$ circuits (still, the most common design scheme to realize quantum circuits for Boolean functions; see [11]) and (2) in the satisfaction of so-called nearest neighbor constraints (an important criterion for many physical quantum architectures; see [12]). Nevertheless, although a physical realization of this library has (theoretically) been discussed e.g. in [13], no thorough examination from a logic design perspective has been conducted yet. This is mainly caused by the fact that existing investigations heavily relied on heuristics. No exact results are available for this library yet.

In this work, we address this issue. We conduct an exact experimental evaluation of the NCV-$|v_1\rangle$ library from a logic design perspective. For this purpose, an exact synthesis scheme is provided which realizes a minimal quantum circuit for a given arbitrary reversible Boolean function. This exact synthesis scheme enables a detailed examination of the costs of NCV-$|v_1\rangle$ circuits for Boolean functions. In order to tackle the complexity of exact synthesis, solving solutions for Boolean satisfiability are exploited.

Our experimental analysis unveiled interesting properties of NCV-$|v_1\rangle$ circuits. We prove that the mapping from reversible gates as proposed in [11] indeed is minimal for many cases. Besides that, NCV-$|v_1\rangle$ circuits seem to be more costly than circuits relying on the previously introduced NCV library (with respect to gates). Nevertheless, a detailed examination unveiled that this can be compensated by a proper encoding of the respective basis states. Then, with respect to the number of gates and a logic design perspective, NCV-$|v_1\rangle$ circuits provide a more efficient alternative compared to the established NCV library. These results provide a further foundation to the discussion about the applicability of the NCV-$|v_1\rangle$ library for quantum computation.

The remaining paper is structured as follows. Section II provides the background on quantum computation and the considered gate libraries. Afterwards, the exact synthesis scheme is described in Section III which has been used in order to conduct the anticipated experimental examination. Section IV eventually describes the performed experiments and summarizes their results. Finally, Section V concludes the paper and outlines some further directions for future work.

TABLE I
QUANTUM OPERATIONS NOT, V AND $\mathrm{V}^\dagger$

x	$NOT(x)$	$V(x)$	$\mathrm{V}^\dagger(x)$
0	1	v_0	v_1
v_0	v_1	1	0
1	0	v_1	v_0
v_1	v_0	0	1

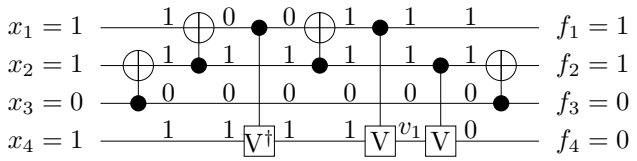


Fig. 1. Quantum circuit using the NCV gate library

II. QUANTUM GATES & CIRCUITS

The basic information unit for a quantum computer is the qubit. Theoretically, a qubit can assume an infinite number of different states. However, from a logic design perspective and in accordance to the established gate libraries, a simplified state model is assumed. Here, the possible states of a qubit are restricted to the four values $S = \{|0\rangle, |v_0\rangle, |1\rangle, |v_1\rangle\}$, i.e. a four-valued logic is considered. This includes representations for the established Boolean values $|0\rangle$ and $|1\rangle$ as well as special quantum values $|v_0\rangle$ and $|v_1\rangle$. The fundamental operations needed in order to realize a Boolean function in a quantum computer are NOT, V, and $\mathrm{V}^\dagger$. The precise definition of both, the quantum values as well as their respective operations, differs depending on the assumed gate library. In this work, the following two libraries are considered.

A. NCV Library

The quantum gate library introduced by Barenco et al. [4] is the most-applied one in the logic design for quantum circuits. The library is universal, i.e. any arbitrary reversible Boolean function can be realized by cascades of NCV gates. Here, the Boolean values $|0\rangle$ and $|1\rangle$ are *basis states*, while the states $|v_0\rangle$ and $|v_1\rangle$ originate from superposition and are explicitly given by $|v_0\rangle = \frac{1+i}{2}\binom{1}{-i}$ and $|v_1\rangle = \frac{1+i}{2}\binom{-i}{1}$. The respective operations NOT, V, and $\mathrm{V}^\dagger$ are accordingly defined by the unitary matrices

$$NOT = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right), \mathrm{V} = \frac{1+i}{2}\left(\begin{smallmatrix} 1 & -i \\ -i & 1 \end{smallmatrix}\right), \text{ and } \mathrm{V}^\dagger = \frac{1-i}{2}\left(\begin{smallmatrix} 1 & i \\ i & 1 \end{smallmatrix}\right).$$

The V operation is also known as the square root of NOT, since two consecutive V operations are equivalent to an inversion. If fed with a Boolean value (i.e. $|0\rangle$ or $|1\rangle$), a V operation leads to one of the quantum values (i.e. $|v_0\rangle$ or $|v_1\rangle$, respectively). The $\mathrm{V}^\dagger$ gate performs the inverse operation of the V gate, i.e. $\mathrm{V}^\dagger = \mathrm{V}^{-1}$. Table I provides an overview of all possible transformations of quantum states (denoted in the rows) and operations (denoted in the columns).

Having this as basis, a quantum circuit in the NCV library is defined as follows:

Definition 1. *A quantum circuit based on the* NCV library *is a cascade of quantum gates over one or two qubits. Unary quantum gates, i.e. gates over one qubit, apply the NOT operation at the respective qubit (denoted* target qubit*). Two-qubit gates are additionally composed of a* control qubit*. Those gates perform one of the respective NOT, V, or $\mathrm{V}^\dagger$ operations at the target qubit only if the control qubit is assigned* 1*. The number of qubits is defined by n, while the number of gates is defined by d.*

Example 1. *Fig. 1 illustrates an NCV quantum circuit composed of $n = 4$ qubits and $d = 7$ quantum gates, which maps e.g. the input pattern* 1101 *to the output pattern* 1100*.*

B. NCV-$|v_1\rangle$ Library

Although the NCV gate library is universal, i.e. every Boolean function can be realized by it [4], different extensions and alternatives have also been introduced. In this work, we additionally consider the NCV-$|v_1\rangle$ library recently introduced in [11] and based on the physical foundations described in [13].

In contrast to the NCV library, *qudits* instead of qubits are assumed here, i.e. not a two level quantum system but a (multiple-valued) 4-level quantum system is assumed. More precisely, the NCV-$|v_1\rangle$ library assumes all states $|0\rangle$, $|v_0\rangle$, $|1\rangle$, and $|v_1\rangle$ introduced above to be basis states. Accordingly, the respective operations NOT, V, or $\mathrm{V}^\dagger$ are not defined in terms of two-level unitary matrices anymore, but four-level descriptions, i.e.

$$NOT = \left(\begin{smallmatrix} 0&0&1&0 \\ 0&0&0&1 \\ 1&0&0&0 \\ 0&1&0&0 \end{smallmatrix}\right), \mathrm{V} = \left(\begin{smallmatrix} 0&0&0&1 \\ 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \end{smallmatrix}\right), \mathrm{V}^\dagger = \left(\begin{smallmatrix} 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&0&0&0 \end{smallmatrix}\right).$$

Based on this, the transformations shown in Table I remain the same but are applied on basis states rather than on (partially) superposition states. This enables quantum gates which are not controlled by 1 but instead by other basis states (v_1 in this case). More precisely, a quantum circuit in the NCV-$|v_1\rangle$ library is defined as follows:

Definition 2. *A quantum circuit based on the* NCV-$|v_1\rangle$ *library is a cascade of quantum gates over one or two qudits. Unary quantum gates apply one of the operations NOT, V, or $\mathrm{V}^\dagger$ at the respective qudit (denoted* target qubit*), while two-qubit gates are additionally composed of a* control qudit*. However, in contrast to the NCV library, the respective operation at the target qudit is not performed if the control qudit is* 1*, but if it is assigned v_1.*

For sake of simplicity, qubits and qudits are used interchangeably in the remainder of this paper.

Example 2. *Fig. 2 shows a quantum circuit with $n = 4$ qubits and $d = 8$ quantum gates, which realizes the same reversible function as the circuit from Fig. 1, but is composed of gates from the NCV-$|v_1\rangle$ library. The different control value is emphasized by v_1 at the corresponding control qubits.*

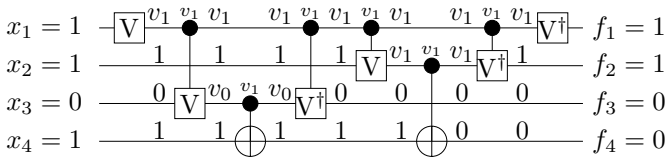


Fig. 2. Quantum circuit using the NCV-$|v_1\rangle$ gate library

III. Applied Exact Synthesis Scheme

Initial evaluations on NCV-$|v_1\rangle$ circuits already unveiled promising properties, but entirely relied on results obtained heuristically. In order to conduct an *exact* examination, we developed a corresponding *exact* synthesis scheme. This holds the challenge of a significant complexity: Determining exact, i.e. minimal, circuits realizing a given function requires to consider all possible gate combinations. To cope with this complexity, we exploited the power of existing solving engines for Boolean satisfiability[1]. This section describes the applied solution. First, a brief background on the utilized solving engines is provided. Afterwards, the exact synthesis scheme is described in detail.

A. Solvers for Boolean Satisfiability

The *Boolean satisfiability problem* (SAT problem) is one of the most important core problems in computer science and, therefore, well studied. It is defined as follows:

Definition 3. *Let f be a Boolean function in* Conjunctive Normal Form *(CNF), i.e. as a* Product of Sums *(PoS). The SAT Problem is to determine an assignment of the variables of f such that the function evaluates to true or to prove that no such assignment exists. If there exists such an assignment, then f is* sat*isfiable; otherwise it is* unsat*isfiable.*

A CNF is a conjunction of clauses, whereas a clause is a disjunction of literals. A literal can be either a variable in its positive or negative phase.
The task of searching for a satisfying assignment or proving that no such assignment exists is performed by automatic theorem proving algorithms, so-called SAT solvers. Most of the modern SAT solvers are based on the main ideas originally proposed in [15], which include three main steps:

1) the decision heuristic assigning values to free variables,
2) the *Boolean Constraint Propagation* (BCP) procedure deducing all implications of the last decision, and
3) the conflict analysis trying to resolve appearing conflicts by backtracking.

Modern state-of-the-art SAT solvers involve many improvements of these basic procedures, e.g. more efficient BCP schemes like in [16], better conflict analysis and learning [17], or sophisticated decision heuristics [18]. Due to their efficiency, nowadays SAT solvers are suitable search engines for many applications like automatic test pattern generation, logic synthesis, as well as equivalence, model, and property checking.

[1]This was inspired from approaches such as [7], [8], [14] where SAT solvers have been utilized for synthesis as well.

B. SAT-based Exact Synthesis

Utilizing SAT solvers, a minimal NCV-$|v_1\rangle$ quantum circuit for a given function f is synthesized by iteratively applying the following two steps:

1) Formulate the task "Generate an NCV-$|v_1\rangle$ circuit with a fixed number of d gates" as an instance of Boolean satisfiability and
2) apply a SAT solver in order to check whether this instance is satisfiable or not.

The SAT instance needs to be satisfiable if and only if there is a quantum circuit with *d* gates for realizing the reversible function *f*. The exact synthesis problem is formulated as a sequence of such SAT instances starting with $d = 1$. In case of unsatisfiability, *d* is increased for the next iteration. This continues until a solution is found. By this, a minimal solution is found once the respective SAT instance turns satisfiable. The following sections describe the structure of the respective instance, i.e. the used variables as well as the applied constraints, in detail.

C. Used Variables

In order to formulate the synthesis problem in terms of a SAT instance, the following variables are used:

Definition 4. *Let $f : \mathcal{B}^n \to \mathcal{B}^n$ be a reversible Boolean function over n variables to be synthesized with d NCV-$|v_1\rangle$ gates. Then, variables $z_{i,j,k}$ are introduced in order to represent the quantum states for each truth table line $(0 \leq i < 2^n)$, at each position in the circuit $(0 \leq j \leq d)$, and for each qubit $(0 \leq k < n)$.*

Since these $z_{i,j,k}$-variables have to represent all states from $S = \{|0\rangle, |v_0\rangle, |1\rangle, |v_1\rangle\}$, they are actually substituted by a Boolean variable pair $x_{i,j,k}y_{i,j,k}$ eventually representing a four-valued encoding as follows:

$z_{i,j,k}$	$x_{i,j,k}y_{i,j,k}$
0	0 0
v_0	0 1
1	1 0
v_1	1 1

That is, each assignment to $z_{i,j,k}$-variables is actually an assignment to the variables $x_{i,j,k}y_{i,j,k}$, whereby if $y_{i,j,k}$ is assigned 0, a Boolean value is assumed and, if it is 1, one of the quantum values v_0 and v_1 is assumed.

Besides the representation of states, also variables are introduced representing which gate type is chosen for a corresponding gate position.

Definition 5. *Let $f : \mathcal{B}^n \to \mathcal{B}^n$ be a reversible Boolean function to be synthesized with d NCV-$|v_1\rangle$ gates. Then, $\vec{q}_j = (q_{j,0} \dots q_{j,s-1})$ is a vector of $s = \lceil log_2(g) \rceil$ Boolean variables representing the chosen quantum gate type for the j^{th} gate of the circuit with $0 \leq j < d$. The variable g denotes the total number of possible gate types available in the NCV-$|v_1\rangle$ library for n qubits.*

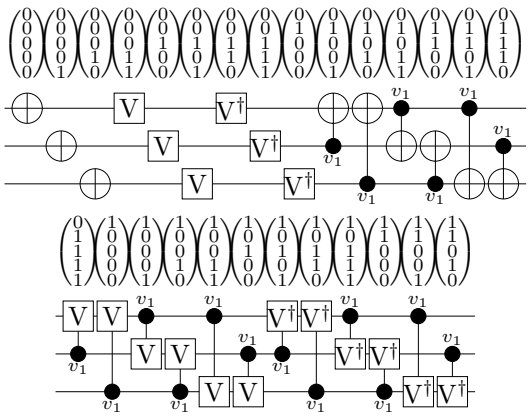


Fig. 3. Mapping of vector assignments to quantum gates for $n = 3$

The total number g of possible gate types for n qubits is calculated as follows:

1) Each of the three unary gate types NOT, V, and V† can be applied to n qubits, resulting in a total of $3n$ different unary gates.
2) The controlled versions of the respective operations have n possible target qubits and $n-1$ possible control qubits, resulting in a total of $n(n-1)$ gates for each of the three controlled gate types.
3) Altogether this makes $g = 3n + 3n(n-1) = 3n(1 + (n-1)) = 3n^2$ different gates types.

Each of these $3n^2$ possible quantum gate types is assigned a particular unique identifier. If the respective $\vec{q}_j$-variables are assigned such an identifier, then the corresponding gate type is assumed at position j in the circuit. Fig. 3 exemplarily provides the relation between gate types and identifiers for a quantum circuit composed of $n = 3$ qubits.

Over all these variables introduced above, the exact synthesis problem can be formulated. Fig. 4 exemplarily summarizes all needed variables for a circuit to be synthesized composed of $n = 3$ qubits and $d = 3$ gates. As can be seen, for each truth table line, a corresponding set of $x_{i,j,k}y_{i,j,k}$-variable pairs representing the states for each qubit and each circuit position is introduced. At the same time, for each gate position, $\vec{q}_j$-variables representing the respective gate type are introduced.

D. Applied Constraints

Passing the variables introduced above to a SAT solver, assignments representing arbitrary gate types as well as arbitrary quantum states would be chosen. Hence, additional constraints are added ensuring that only valid assignments are allowed which (1) follow the definition of the library and (2) indeed realize the desired function f. More precisely:

1) The variables representing the in- and outputs of the circuit are assigned the corresponding values in the truth table defined by the given function f. Therefore, constraints for the in- and outputs are applied on the $x_{i,0,k}y_{i,0,k}$- and $x_{i,d,k}y_{i,d,k}$-variables, respectively. Since the truth table values are Boolean, variables $y_{i,0,k}$ and $y_{i,d,k}$ are always assigned 0. In contrast, $x_{i,0,k}$ and $x_{i,d,k}$ take the corresponding value of truth table line i for qubit k, i.e.

$$\bigwedge_{i=0}^{2^n-1}\bigwedge_{k=0}^{n-1} x_{i,0,k} = i[k] \quad \wedge y_{i,0,k} = 0 \quad \wedge x_{i,d,k} = f(i[k]) \wedge y_{i,d,k} = 0.$$

2) The vector $\vec{q}_j$ is composed of s Boolean variables used to represent all possible gate types. As this vector may assume 2^s possible assignments, while only $3n^2$ different gates types exist, this results in an overhead of unnecessary assignments which must be blocked. Therefore,

$$\bigwedge_{j=0}^{d-1}(0 \leq \vec{q}_j < 3n^2)$$

is added.
3) Finally, constraints ensuring the correct functionality based on the chosen gate types are added. For this purpose, each gate type represented by an assignment to the $\vec{q}_j$-variables is associated to a function $q(x_{i,j,k}, y_{i,j,k}, \vec{q}_j)$. This function defines the output states $x_{i,j+1,k}y_{i,j+1,k}$ of the respective gate based on the corresponding input states $x_{i,j,k}y_{i,j,k}$ and the gate type $\vec{q}_j$. More formally:

$$\bigwedge_{i=0}^{2^n-1}\bigwedge_{j=0}^{d-1}\bigwedge_{j=0}^{n-1} x_{i,j+1,k}, y_{i,j+1,k} = q(x_{i,j,k}, y_{i,j,k}, \vec{q}_j).$$

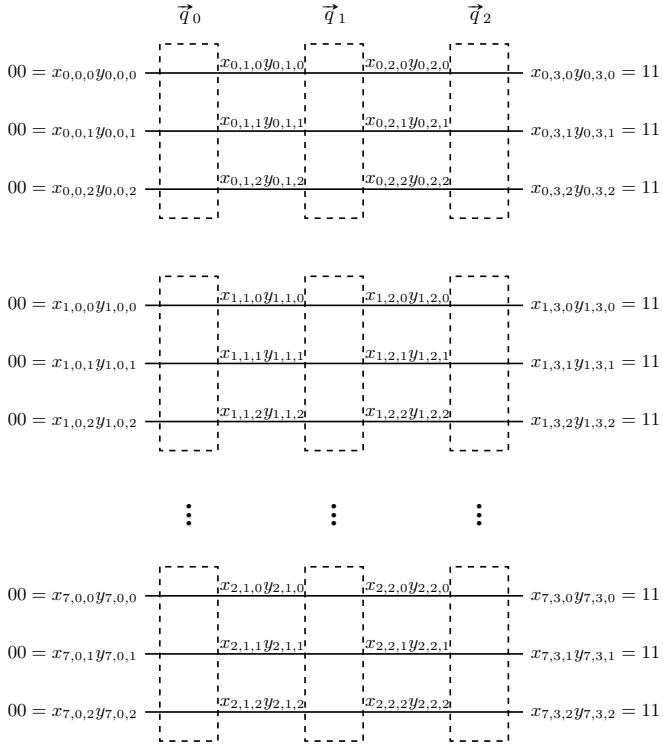


Fig. 4. Variable structure for $n = 3$ qubits and $d = 3$ gates

Passing the resulting SAT instance composed of all variables and all constraints introduced above to a SAT solver, a satisfying solution results if f can be realized with d gates. If this is the case, the precise NCV-$|v_1\rangle$ quantum circuit can be derived from the assignment to all $\vec{q}_j$-variables. If the SAT solver returns unsatisfiable instead, it has been proven that no realization for f with d gates exists. In this case, d is increased by one and solved again. By this, minimal circuits eventually result.

IV. Experimental Examination

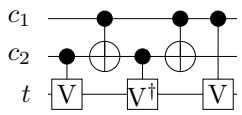
Synthesis of Boolean components for quantum circuits is usually approached from two complementary directions: (1) realizing the desired functionality as a reversible circuit to be mapped into an equivalent quantum circuit or (2) realizing the desired quantum circuit directly. Hence, in order to exactly analyze the suitability of a gate library from a logical design perspective, both directions should be considered. This is done in this section. For this purpose, the exact synthesis scheme proposed above has been implemented in C++. *MiniSAT* [19] together with its *SatELite* preprocessor [20] was used as back-end solving engine. Due to the exponential nature of such an exact synthesis problem, even modern SAT solvers are limited in terms of applicability. Still, minimal circuits could be synthesized for many functions by the proposed approach within acceptable time.

A. Mapping Reversible Circuits

Quantum circuits are inherently reversible [1], i.e. every reversible circuit can be transformed to a quantum circuit. This has been exploited during quantum circuit design and eventually led to an established synthesis flow which, first, realizes the desired functionality as a reversible circuit (using synthesis approaches such as [21], [22], [23], [24], [25], [26]) and, afterwards, applies mapping schemes (such as [4], [6], [7], [8], [9], [10]), to convert the resulting reversible circuit into a quantum circuit.

Reversible circuits are thereby composed of so-called Toffoli gates. A *Toffoli gate* is composed of a *target line* t and a set C of *control lines* with $C = \{c_1, \ldots, c_{|C|}\}$ and $|C| < n$. The target line is inverted if all control lines are set to 1; otherwise the value of the target line is passed through unchanged. This functionality can easily be realized using NCV gates as shown in Fig. 5(a) for a Toffoli gate with $|C| = 2$ control lines. Similar mappings exist for Toffoli gates with a different amount of control lines. With an increasing number of control lines, the resulting quantum circuits become more expensive, i.e. require more quantum gates. Fig. 5(b) provides some numbers on that based on the current state-of-the-art mapping scheme introduced in [9]. The resulting quantum circuits are used as *building blocks* for the mapping-based synthesis flow sketched above.

Using the NCV-$|v_1\rangle$ library results in significantly cheaper mappings and, thus, building blocks. The synthesis scheme proposed in Section III enabled us to realize *minimal* NCV-$|v_1\rangle$ circuits for Toffoli gates composed of up to $|C| = 4$ control gates. An example for $|C| = 2$ control lines is provided in Fig. 6(a). In contrast, minimality of the NCV mappings

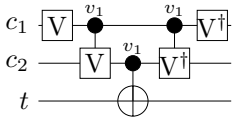


(a) For $|C| = 2$

$\|C\|$	No. of Gates	Opt.?
0	1	✓
1	1	✓
2	5	✓
3	14	?
4	20	?
5	32	?

(b) For $|C| < 6$

Fig. 5. Mapping Toffoli gates to NCV circuits



(a) For $|C| = 2$

$\|C\|$	No. of Gates	Opt.?
0	1	✓
1	3	✓
2	5	✓
3	7	✓
4	9	✓
5	11	?

(b) For $|C| < 6$

Fig. 6. Mapping Toffoli gates to NCV-$|v_1\rangle$ circuits

has only been shown for Toffoli gates with no, one, or two control lines thus far (see the respective right-most columns of Fig. 5(b) and Fig. 6(b)). Moreover, as can clearly been seen, the NCV-$|v_1\rangle$ library is particularly more suitable to realize Toffoli gates than the NCV library (only the case with $|C| = 1$ control lines is an exception). Although also the number of NCV-$|v_1\rangle$ gates increases with more control lines, this increase is linear. In contrast, the NCV library has a non-linear increase. This has already been observed before in [11], but until today it remained unclear whether this mapping is minimal.

Overall, this examination unveiled that the NCV-$|v_1\rangle$ library allows for determining smaller building blocks than it was the case for the NCV library. At the same time, the respective building blocks are significantly cheaper and, hence, allow for more compact realizations than the established synthesis flow, i.e. realizing a reversible circuit first and, afterwards, mapping it to a quantum realization, is applied.

B. Direct Quantum Circuit Synthesis

In a second examination, the sizes of quantum circuits obtained by a direct synthesis scheme are compared. For this purpose, we additionally applied the exact synthesis scheme for NCV circuits as proposed in [8]. Using this approach together with the approach proposed in Section III allowed us to compare the *minimal* realizations for both libraries, NCV and NCV-$|v_1\rangle$, for certain functions (taken from *RevLib* [27]).

Table II gives a summary of the results. The first column provides the name of the considered function, while the following two columns denote the number of gates obtained by direct synthesis assuming an NCV and NCV-$|v_1\rangle$ library, respectively. Based on these numbers, NCV-$|v_1\rangle$ circuits seem to be more costly (with respect to gates) than circuits relying on the NCV library – a clear disadvantage of the NCV-$|v_1\rangle$ library.

However, our investigations showed that these additional costs are almost entirely caused by the fact that controlled NCV-$|v_1\rangle$ gates are sensitive to the $|v_1\rangle$-state. In contrast, NCV circuits are sensitive to the $|1\rangle$-state (cf. Section II). This leads to an unfair advantage for the NCV library, since the considered functions are usually encoded by means of $|0\rangle$ and $|1\rangle$ (due to the $|1\rangle$-sensitivity, the best possible encoding for

TABLE II
COMPARISON OF NCV AND NCV-$|v_1\rangle$ CIRCUITS

	No. of Gates		
	NCV	NCV-$|v_1\rangle$	adj. NCV-$|v_1\rangle$
Toffoli	5	5	4
Peres	4	5	4
Fredkin	7	9	6
Miller	8	10	6
Half-adder-v0	5	8	5
3_17	10	11	8
Toffoli-double	7	8	6
Peres-double	6	10	6
q4example	6	9	6
Low-High-v0	7	7	5
rd32	6	8	6
Zero-One-Two	7	9	6

NCV circuits). But since the applied encoding is only a matter of definition, different encodings, potentially more suited for the other gate library, are valid as well.

To explicitly demonstrate this effect, another series of experiments has been conducted. Here, the functions to be realized have not been encoded conventionally with the basis states $|0\rangle$ and $|1\rangle$, but respectively with the basis states $|v_0\rangle$ and $|v_1\rangle$. Corresponding results are provided in the right-most column of Table II. As can clearly be seen, the adjusted encoding leads to significantly cheaper NCV-$|v_1\rangle$ circuits. Moreover, the resulting circuits even provide a more efficient alternative compared to the established NCV library – in almost all cases circuits with less gates result.

Overall, this examination unveiled that the NCV-$|v_1\rangle$ library also leads to cheaper realizations for quantum circuits in general. Although the encoding of the function has to be adjusted for this purpose, this is acceptable and only a matter of definition. From a logic design perspective, the NCV-$|v_1\rangle$ gate library clearly superiors the currently established NCV library.

V. CONCLUSION

In this work, we conducted an exact experimental examination of quantum circuits based on the NCV-$|v_1\rangle$ library. For this purpose, an exact synthesis scheme has been applied which utilized the power of solvers for Boolean satisfiability. Our examination clearly unveiled that, from a logic design perspective, the NCV-$|v_1\rangle$ library is a promising alternative with certain advantages compared to the established NCV library. It provides a better mapping scheme from reversible circuits to quantum circuits and, assuming an respectively adjusted input/output encoding, allows for more compact realizations of arbitrary functions.

The results of this examination provide an important step towards the completion of the discussion on the applicability of the NCV-$|v_1\rangle$ library for quantum computation. In the past, a theoretical and conceptual discussion on the physical applicability of this library has been conducted e.g. in [13]. Now, we complemented these findings by exact results on logic synthesis. Future work focuses on physical issues such as the direct realizations of the qudits, the emulation of qudits e.g. by existing qubit-realizations, or the compatibility to existing fault-tolerant quantum error correction protocols as well as the derivation of more precise cost metrics for this library.

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[2] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Foundations of Computer Science*, pp. 124–134, 1994.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Theory of computing*, 1996, pp. 212–219.

[4] A. Barenco, C. H. Bennett, R. Cleve, D. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *The American Physical Society*, vol. 52, pp. 3457–3467, 1995.

[5] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. on CAD*, vol. 32, no. 6, pp. 818–830, 2013.

[6] A. Abdollahi and M. Pedram, "Analysis and synthesis of quantum circuits by using quantum decision diagrams," in *Design, Automation and Test in Europe*, 2006, pp. 317–322.

[7] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis." *IEEE Trans. on CAD*, vol. 25, no. 9, pp. 1652–1663, 2006.

[8] D. Große, R. Wille, G. W. Dueck, and R. Drechsler, "Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares," in *Int'l Symp. on Multi-Valued Logic*, 2008, pp. 214–219.

[9] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control toffolli gates," in *Int'l Symp. on Multi-Valued Logic*, 2011, pp. 288–293.

[10] R. Wille, M. Soeken, C. Otterstedt, and R. Drechsler, "Improving the mapping of reversible circuits to quantum circuits using multiple target lines," in *ASP Design Automation Conf.*, 2013, pp. 145–150.

[11] Z. Sasanian, R. Wille, and D. M. Miller, "Realizing reversible circuits using a new class of quantum gates," in *Design Automation Conf.*, 2012, pp. 36–41.

[12] R. Wille, A. Lye, and R. Drechsler, "Considering nearest neighbor constraints of quantum circuits at the reversible circuit level," *Quantum Information Processing*, vol. 13, no. 2, pp. 185–199, 2014.

[13] A. Muthukrishnan and C. R. Stroud, "Multi-valued logic gates for quantum computation," vol. 62, no. 052309, 2000.

[14] D. Große, R. Wille, G. W. Dueck, and R. Drechsler, "Exact multiple-control Toffoli network synthesis with SAT techniques," *IEEE Trans. on CAD*, vol. 28, no. 5, pp. 703–715, 2009.

[15] M. Davis, G. Logeman, and D. Loveland, "A machine program for theorem-proving," *Comm. of the ACM*, vol. 5, pp. 394–397, 1962.

[16] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Design Automation Conf.*, 2001, pp. 530–535.

[17] J. P. Marques-Silva and K. A. Sakallah, "GRASP – a new search algorithm for satisfiability," in *Int'l Conf. on CAD*, 1996, pp. 220–227.

[18] E. Goldberg and Y. Novikov, "BerkMin: a fast and robust SAT-solver," in *Design, Automation and Test in Europe*, 2002, pp. 142–149.

[19] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Lecture Notes in Computer Science*, vol. 2919. Springer, 2003, pp. 502–518.

[20] N. Eén and A. Biere, "Effective preprocessing in SAT through variable and clause elimination," in *Conference on Theory and Applications of Satisfiability Testing*. Springer, 2005, pp. 61–75.

[21] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 25, no. 11, pp. 2317–2330, 2006.

[22] D. Maslov, G. W. Dueck, and D. M. Miller, "Techniques for the synthesis of reversible Toffoli networks," *ACM Trans. on Design Automation of Electronic Systems*, vol. 12, no. 4, 2007.

[23] R. Wille, D. Große, G. W. Dueck, and R. Drechsler, "Reversible logic synthesis with output permutation," in *VLSI Design*, 2009, pp. 189–194.

[24] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conf.*, 2009, pp. 270–275.

[25] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler, "Synthesis of reversible circuits with minimal lines for large functions," in *ASP Design Automation Conf.*, 2012, pp. 85–92.

[26] C. Chandak, A. Chattopadhyay, S. Majumder, and S. Maitra, "Analysis and improvement of transformation-based reversible logic synthesis," in *Int'l Symp. on Multi-Valued Logic*, 2013, pp. 47–52.

[27] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: an online resource for reversible functions and reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2008, pp. 220–225, RevLib is available at http://www.revlib.org.