Evaluating ESOP Optimization Methods in Quantum Compilation Flows

Giulia Meuli[1], Bruno Schmitt[1], Rüdiger Ehlers[2],
Heinz Riener[1], and Giovanni De Micheli[1]

[1]École Polytechnique Fédérale de Lausanne, Switzerland
[2]University of Bremen, Germany

Abstract. Exclusive-or sum-of-products (ESOP) expressions are used as intermediate representations in quantum circuit synthesis flows, and their complexity impacts the number of gates of the resulting circuits. Many state-of-the-art techniques focus on minimizing the number of product terms in a ESOP expression, either exactly or in a heuristic fashion.
In this paper, we investigate into ESOP optimization considering two recent quantum compilation flows with opposite requirements. The first flow generates Boolean functions with a small number of Boolean variables, which enables the usage of methods from exact synthesis; the second flow generates Boolean functions with many Boolean variables, such that heuristics are more effective. We focus on the reduction of the number of T gates, which are expensive in fault-tolerant quantum computing and integrate ESOP optimization methods into both flows. We show an average reductions of 36.32% in T-count for the first flow, while in the second flow an average reduction of 28.23% is achieved.

Keywords: reversible logic synthesis · logic optimization · ESOP · quantum circuit

1 Introduction

Quantum compilation is the problem of translating a computational description of a quantum algorithm into basic quantum operations. Two main approaches are used in practice: 1) manual compilation, where a designer manually synthesizes (and optimizes) each component of the computational description and generates the final quantum circuit by hand, and 2) automatic compilation, which supports designers in the synthesis task by offering fast and scalable solutions to systematically explore the design space. On the one hand, automatic synthesis allows designers to deal with larger problems that are too complex to be tackled manually; on the other hand, systematic design space exploration enables designers to identify optimization capabilities otherwise overlooked.

Recent attempts in the field of automatic quantum compilation include *LUT-based Hierarchical Reversible Logic Synthesis* (*LHRS*) [25] and *Decomposition Based Synthesis* (*DBS*) [23]. The former framework, *LHRS*, uses a hierarchical

method to synthesize quantum circuits from specifications provided in form of combinational logic designs. The designs are first decomposed into networks of look-up tables (LUTs). Then, a quantum circuit is assembled by translating each LUT into quantum gates. The latter framework, *DBS*, uses Young-subgroup based reversible synthesis [3] to compile quantum state permutations into quantum circuit. Both frameworks, *LHRS* and *DBS*, use *exclusive-or sum-of-products* (ESOPs) as representations of reversible logic gates generated during the translation process.

ESOPs are a classical two-level logic representation consisting of one level of AND-gates, followed by one level of XOR-gates. They provide a compact logic representation of Boolean functions, and are, for some classes of functions, exponentially more compact when compared to the sum-of-products (SOP) representation [21]. This compactness can be particularly recognized when XOR-intensive circuits, such as the parity function, need to be represented and makes ESOPs useful to describe arithmetic and cryptographic primitives [15].

Over the years, many advanced synthesis and optimization methods have been discovered for ESOPs. Exact methods [16, 19, 20] target the minimization of the number of product terms in an ESOP, such that the number becomes provably minimal. Their applicability, however, is limited to Boolean functions with at most 7 Boolean variables. Moreover, they often require large tables of precomputed information and need a substantial amount of runtime to guarantee minimality.

Heuristic methods [13, 20, 27] are capable of reducing large-scale ESOPs with thousands of cubes by repeatedly applying simple cube transformation rules that first expand and then collapse cubes. Such transformation-based optimization strategies are fast, lead to significant reductions, and can be applied even if ESOPs with many Boolean variables are considered. Heuristic methods, however, cannot guarantee optimality and their progress often strongly degrades over time—the chances of finding a pair of cubes that can be collapsed decreases and the improvement saturates.

Overall, in this work, we target fault-tolerant quantum computation and analyze the impact of ESOP optimization methods on the number of T gates of the final quantum circuit. The T gates have been recognized as the most expensive gates in fault-tolerant quantum computing [1].

We integrate advanced ESOP optimization methods, both heuristic and exact, into recent quantum compilation flows. In particular, we consider *LHRS* and *DBS* as two possible application scenarios with opposite requirements: *DBS* uses simple specifications, such that only a few Boolean functions with a relatively small number of Boolean variables have to be synthesized. In this case, exact synthesis methods are useful and allow us to generate ESOPs of provably minimal size. In *LHRS*, however, ESOP optimization has to deal with many and larger Boolean functions. In this case, we advocate heuristic ESOP optimization methods to keep the approach scalable.

In our analysis, we consider two de-facto standard cost functions from logic synthesis—the number of product terms and the number of literals—and propose

a novel exact synthesis procedure for ESOPs. Our procedure allows users to specify costs for each cube, considered during the synthesis process. We formulate the synthesis problem by introducing a weighted-version of the Helliwell equation [17], and solve the problem using partial weighted MAX-SAT.

2 Preliminaries

2.1 ESOP representation of Boolean functions

Definition 1. *An ESOP over n Boolean variables, $x_1, \ldots, x_n \in \mathbb{B}$, is an expression of form $t_1 \oplus \cdots \oplus t_k$, where each $t_i = l_{i,1} \cdots l_{i,l_i}$ is a product term (or cube) of literals $l_{i,j} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ for $1 \leq i \leq k$ and $1 \leq j \leq l_i$. The symbol $\oplus$ denotes the modulo-2 addition (XOR-operation), and $\bar{x}_i$ denotes the negated Boolean variable x_i for $1 \leq i \leq n$.*

An ESOP expression can be interpreted as a two-level logic circuit, which realizes a possibly incompletely-specified Boolean function $f : \{0, 1, -\}^n \to \mathbb{B}$, i.e., $f(x_1, \ldots, x_n) = t_1 \oplus \cdots \oplus t_k$ for all possible valuations of the Boolean variables $x_1, \ldots, x_n$.

2.2 ESOP-based reversible logic synthesis

Reversible circuits are logic networks with the same number of inputs and outputs, composed of reversible gates. The most commonly used gates are the single-target gates and the multiple-controlled Toffoli gates.

Definition 2. *Let $c : \mathbb{B}^k \to \mathbb{B}$ be a Boolean function, called control function. Also, let $C = \{x_1, \ldots, x_k\}$ be the control lines and let $x_t \notin C$ be a target line. Then the single-target gate $T_c(C, t) : \mathbb{B}^n \to \mathbb{B}^n$ is a reversible Boolean function which maps:*

$$(x_1, \ldots, x_n) \to \begin{cases} x_i & \text{if } \; i \neq t \\ x_t \oplus c(x_1, \ldots, x_k) & \text{otherwise} \end{cases}$$

Definition 3. *If the control function c can be expressed as a single product term $c = \bigwedge_{i=1}^{k}(x_i \oplus p_i)$ using a single-target gate $T_c(C, t)$, where p_i, $1 \leq i \leq k$, are the polarities of the controls, then we call the gate a multiple-controlled Toffoli gate.*

A multiple-controlled Toffoli gate is a reversible gate acting on the bits in $x_1, \ldots, x_k, x_t$, such that the bits in C remain unchanged and the bit x_t flips if the control function $c(x_1, \ldots, x_k)$ evaluates to true.

ESOP-based reversible synthesis methods are based on the observation that an ESOP can be directly translated into a reversible circuit, as each term of the expression corresponds to a multiple-controlled Toffoli gate [5, 6]. The method generates as many Toffoli gates as cubes in the expression, all cascaded and targeting the same bit.

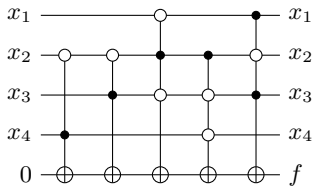
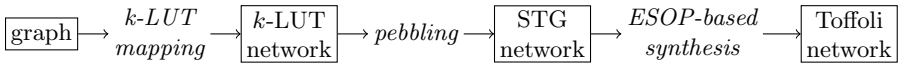
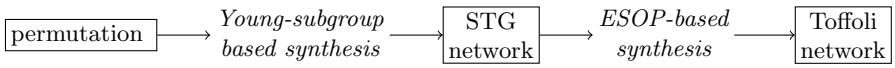


Fig. 1. Example of a reversible circuit of mixed-polarity multiple-controlled Toffoli gates



(a) *LHRS* flow with direct mapping of single-target gates



(b) *DBS* flow

Fig. 2. Two different state-of-the-art compilation flows for Boolean functions that use ESOP-based reversible synthesis

Example 1. The Toffoli network in Fig. 1 corresponds to the ESOP expression:

$$x_4\overline{x}_2 \oplus x_3\overline{x}_2 \oplus \overline{x}_3x_2\overline{x}_1 \oplus \overline{x}_4\overline{x}_3x_2 \oplus x_3\overline{x}_2x_1$$

Some optimization techniques aiming at reducing the cost of the generated reversible circuits have been proposed in literature [12, 28]. The final circuit reflects the quality of the ESOP expression, so the synthesis process is crucial for this application.

3 Optimal ESOP for quantum compilation

The problem of automatically compiling a Boolean function into a universal quantum library is largely addressed in literature [7, 9, 22].

Among the available synthesis methods, hierarchical flows have the capability of being scalable, as they are based on a logic network representation [18], e.g., *LHRS* [26]. The input to *LHRS* is a classical logic network, e.g., provided in a hardware description language; the output is a quantum network realized in terms of Clifford+T gates. The framework is based on the usage of k-feasible Boolean logic networks (k-LUT networks), which consist of look-up tables (LUTs) with at most k inputs. Synthesis proceeds in two steps: (i) each k-LUT is mapped into a reversible single-target gate with k control lines, (ii) each reversible single-target gate is mapped into a Clifford+T network. *LHRS* provides different methods to perform the second step. One method, the so-called *direct mapping*, makes use of the ESOP representation of the control function

of a reversible single-target gate, which can be *directly* translated into multiple-controlled Toffoli gates [6](see Section 2.2) and further translated into quantum gates [11]. The flow of this method is shown in Fig. 2(a).

A second strategy (Fig. 2(b)) for quantum compilation is based on decomposing the initial function, given as a permutation, using the Young-subgroup method described in [3]. It only differs from the first one for the function's specification and the decomposition strategy employed. Differences that will result in a less scalable flow. The final steps are shared between the two flows: ESOP-based reversible synthesis is used to generate a Toffoli network and successively each Toffoli gate is compiled into quantum operations from the Clifford+T library using the method described in [11].

In this work, we address the Clifford+T universal quantum library, and try to optimize the number of T gates by applying ESOP optimization to the compilation flows. Nevertheless, our analysis and methods are applicable to the other quantum libraries, as far as the implementations of Toffoli gates are known.

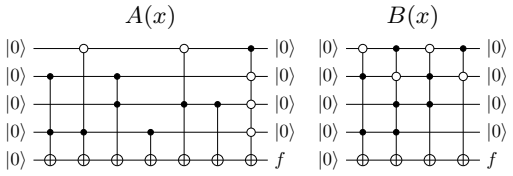
4 Motivation

In the following, we introduce the problem of finding the right ESOP synthesis method to generate reversible circuits, which can be compiled into quantum circuits with optimal characteristics: minimal number of T gates and reduced number of Clifford gates.

Example 2. Given the Boolean function $f(x) = \overline{x}_1\overline{x}_3x_4 \vee \overline{x}_2\overline{x}_3x_4 \vee \overline{x}_1x_2x_3\overline{x}_4 \vee x_1\overline{x}_2x_3\overline{x}_4$ with $x = x_1, \ldots, x_4$, two possible ESOP expressions for f are:

$$A(x) = x_3x_1 \oplus \overline{x}_4x_1 \oplus x_3x_2 \oplus x_1 \oplus \overline{x}_4x_2 \oplus x_2 \oplus x_4\overline{x}_3\overline{x}_2\overline{x}_1$$
$$B(x) = \overline{x}_4x_3x_1 \oplus x_4\overline{x}_3x_2x_1 \oplus \overline{x}_4x_3x_2 \oplus x_4\overline{x}_3$$

The first expression $A(x)$ is composed of 7 product terms while the other expression, $B(x)$, is smaller and has size 4. We can use these ESOPs to synthesize a reversible network for f and successively we can compile them into quantum gates using the algorithm described in [11]. The resulting networks and the composition of the quantum circuits are reported in Fig 3: H is the number of Hadamard gates, NOT and $CNOT$ are respectively the number of X and the number of controlled-X gates, T is the number of T gates. It is clearly shown how the second ESOP, independently from the smaller size, generates a quantum circuit with more gates. Differently, the first ESOP, that has larger size, shows characteristics allowing the compiler to create a circuit with reduced T gates, and fewer gates in general. We want to identify which are the characteristics that lead to a better quantum circuit. With this in mind, we can notice how the first ESOP has cubes with less literals, with respect to the second ESOP. Thus $A(x)$ generates a reversible circuit with multiple-controlled Toffoli gates with less controls and consequently a quantum circuit with less T gates.



ESOP	Gate types		
	H	NOT	CNOT T
$A(x)$	18	6	46 52
$B(x)$	24	6	54 63

Fig. 3. Synthesis results of two different ESOPs for the same function f

It is evident how the quantum compilation problem can take advantage of optimal ESOP synthesis strategies. Consequently, in this work we apply state-of-the-art ESOP synthesis and optimization methods, e.g., the heuristic *EXORCISM* [14], into recent quantum compilation flows. In addition, we present a constraint-based ESOP synthesis method that accepts an arbitrary cost function, as Example 2 suggests that different cost metrics should be considered for ESOPs in quantum compilation.

5 Constraint-based ESOP synthesis

The problem of finding an ESOP expression that realizes a Boolean function is known as *ESOP synthesis*. The seminal work of Perkowski and Chrzanowska-Jeske [17] introduces the *Helliwell decision function* to characterize the solution space of ESOP synthesis for a given Boolean function.

5.1 Helliwell decision function

The Helliwell decision function $H_f(g_1, \ldots, g_K)$, $K \leq 3^n$, for a given Boolean function $f(x_1, \ldots, x_n)$ describes synthesis as an odd-even covering problem in terms of the minterms of f. For each possible product term in n Boolean variables, a decision variable g_i, $1 \leq i \leq K$, is introduced. The Helliwell decision function is then defined by the logic equation

$$\bigwedge_{m \in f} \left(\left(\bigoplus_{g \in I(m)} g \right) \oplus f(m) \oplus 1 \right), \tag{1}$$

where $m \in f$ denotes that m is a minterm of f and I maps each minterm to the decision variables $g_{i_1}, \ldots, g_{i_l}$ whose product terms are covered by m.

The logic equation (1) is constructed in such a way that every satisfying assignment $\hat{g}$ for $g = g_1, \ldots, g_K$ for $H(g)$ directly corresponds to an ESOP expression functionally equivalent to f.

Example 3. Given the Boolean function $f(x_1, x_2) = x_1 \vee x_2$ with Boolean variables x_1 and x_2, the Helliwell decision function using 9 Boolean variables $g_1, \ldots, g_9$,

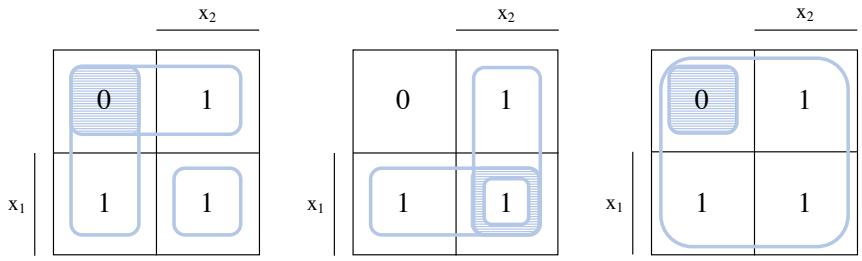


Fig. 4. Three possible ESOP covering for the function $f = x_1 \vee x_2$

that are,

$$g_1 = \overline{x}_1\overline{x}_2 \quad g_2 = \overline{x}_1 x_2 \quad g_3 = x_1\overline{x}_2 \quad g_4 = x_1 x_2$$
$$g_5 = x_1 \quad g_6 = \overline{x}_1 \quad g_7 = x_2 \quad g_8 = \overline{x}_2$$
$$g_9 = 1.$$

The SAT solver will find a selection of the cubes such that minterms for which f evaluates to one are covered an odd number of times, whether minterms for which f evaluates to false are covered an even number of times. Constraints must be added to the problem in order for the SAT solver to find a valid solution. The overall Helliwell decision function for f is:

$$\begin{aligned} H(g) =& (g_1 \oplus g_6 \oplus g_8 \oplus g_9 \oplus 0 \oplus 1) \ \wedge \ (g_2 \oplus g_7 \oplus g_6 \oplus g_9 \oplus 1 \oplus 1) \ \wedge \\ & (g_3 \oplus g_5 \oplus g_8 \oplus g_9 \oplus 1 \oplus 1) \ \wedge \ (g_4 \oplus g_5 \oplus g_7 \oplus g_9 \oplus 1 \oplus 1) \end{aligned}$$

Fig. 4 shows three possible ESOP covers on the Karnaugh map: g_4, g_6, g_8 and g_4, g_5, g_7 and g_6, g_9.

5.2 Size-minimal ESOP synthesis

Size-minimal ESOP synthesis is the problem of finding an ESOP expression for a given Boolean function f with a minimum number of product terms. Utilizing logic equation (1), the problem can be solved by computing minimum satisfying assignments for $H_f(g)$. An assignment $\hat{g}$ is minimum satisfying if the two conditions

$$\text{(a) } H_f(\hat{g}) \text{ and (b) } \forall g : \big(g \not\leftrightarrow \hat{g} \wedge H_f(g)\big) \implies g \not\rightarrow \hat{g}, \tag{2}$$

hold, i.e., if $\hat{g}$ satisfies H_f and no other assignment that satisfies H_f implies $\hat{g}$.

In the following, the idea of utilizing the Helliwell decision function for synthesizing size-minimum ESOP expression is generalized to synthesizing cost-minimal ESOP expressions, where the cost function is provided as a part of the input.

5.3 Cost-minimal ESOP synthesis

Given a Boolean function f over n Boolean variables and a cost function $\kappa : \{0,1,-\}^n \to \mathbb{N}_{>0}$, that maps product terms to positive integer values (costs), cost-minimal ESOP synthesis is the problem of finding an ESOP expression $t_1 \oplus \cdots \oplus t_k$ that realizes f such that $\bigwedge_{i=1}^{k} \kappa(t_i)$ is minimal.

We present two different cost function, κ_0 and κ_1 to illustrate the idea of cost-minimal ESOP synthesis. In general, the cost function should be picked keeping the usage of the ESOP expression in mind.

The constant function

$$\kappa_0(t) = 1 \tag{3}$$

defines unit costs for all product terms. If used, each ESOP expression obtained as solution of cost-minimal ESOP synthesis has a minimum number of product terms. The cost function

$$\kappa_1(t) = |t| + 1, \tag{4}$$

where $|t|$ counts the number of literals in t, weights each product term by the number of appearing literals. The additional 1 ensures that all costs—including the costs of the empty product term—are greater than 0.

Example 4. Consider the Boolean function $f_1(x) = \bar{x}_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4 \vee x_1x_2\bar{x}_3\bar{x}_4$ with $x = x_1, \ldots, x_4$. A cost-minimal ESOP expression that realizes f_1 with respect to cost function κ_0 is

$$\bar{x}_1x_2\bar{x}_4 \oplus x_2\bar{x}_3 \oplus \bar{x}_2x_3\bar{x}_4 \oplus \bar{x}_1\bar{x}_2x_3 \oplus x_1\bar{x}_3x_4,$$

whereas a cost-minimal ESOP expression for the same Boolean function with respect to cost functions κ_1 is

$$x_1 \oplus x_2 \oplus \bar{x}_3 \oplus x_4 \oplus \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \oplus x_1x_2x_3x_4.$$

5.4 Computing cost-minimal ESOPs

Next, we present the proposed SAT-based procedure for computing cost-minimal ESOP expressions using (weighted) maximum satisfiability (MAX-SAT) [10].

MAX-SAT deals with solving over-constrained constraint satisfaction problems modulo Boolean logic. The problems consist of hard and soft clauses, where each soft clause is associated with an integer weight greater than 0. The constraint satisfaction problem initially is unsatisfiable and the task of a MAX-SAT oracle is to find a minimal-cost relaxation of the soft clauses, i.e., the oracle has to remove a subset of the soft clauses, such that the problem becomes satisfiable while a given cost function is minimized.

Given a Boolean function f over n Boolean variables and a cost function $\kappa : \{0,1,-\}^n \to \mathbb{N}_{>0}$, cost-minimal ESOP synthesis is solved in three steps:

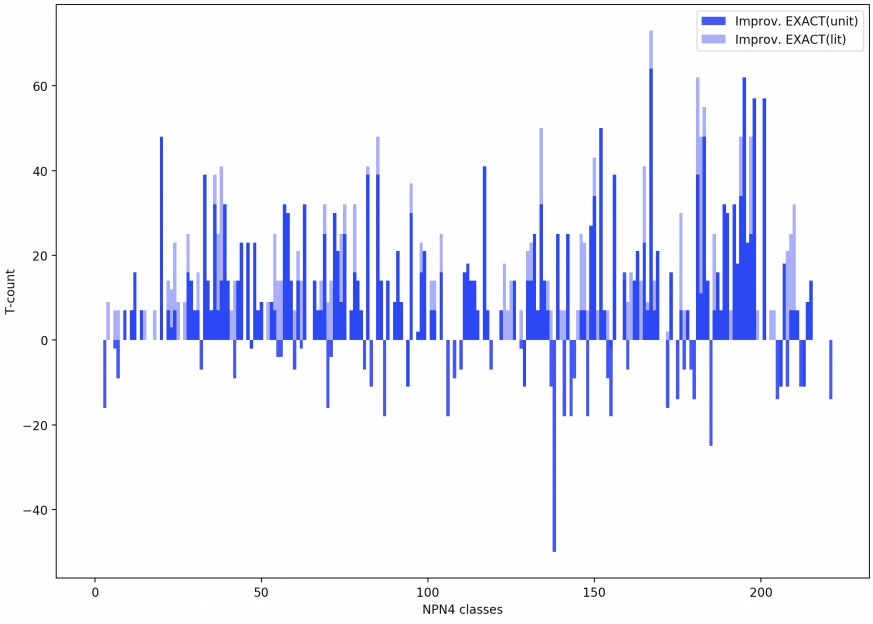


Fig. 5. Histogram showing the improvement over exact methods over *PKRM* with respect to two different cost functions: number of terms (*EXACT(unit)*) and number of literals (*EXACT(lit)*)

1. Formulate the Helliwell decision function $H(g)$ as described in (1).
2. Invoke a MAX-SAT oracle to find a satisfying assignment $\hat{g} = \hat{g}_1, \ldots, \hat{g}_K$ that minimizes $\sum_{i=1}^{K} \kappa(g_i)$ subject to CNF$[H(g)] \wedge (\bigwedge_{i=1}^{n} \bar{g}_i)$, where CNF translates the XOR-clauses to *conjunctive normal form* (CNF).
3. Construct the ESOP from the satisfying assignment $\hat{g}$.

The described approach is independent of the choice of the MAX-SAT oracle and the translation to CNF, but uses them as black-boxes.

6 Results

6.1 NPN4 equivalence classes

In this section, we evaluate the effect of different ESOP optimization methods on simple Boolean functions. As benchmarks, we use the 222 representatives of the NPN4 equivalence classes. We evaluate the number of product terms in the ESOP, as well as, the number of T gates in the generated quantum circuits considering different ESOP synthesis methods and the proposed constraint-based approach:

Cost function	**ESOP Synthesis Method**				
	PPRM	PKRM	EXORCISM	EXACT(unit)	EXACT(lit)
avg. ESOP size	7.77	4.69	3.41	3.41	3.42
avg. num. T gates	87.35	82.32	59.05	67.50	58.19

Table 1. Comparison of different ESOP synthesis methods

1. *Positive Polarity Reed Muller* (*PPRM*) [29],
2. *Pseudo-Kronecker Reed Muller* (*PKRM*) [4],
3. *EXORCISM* [14] and
4. *EXACT(unit)* and *EXACT(lit)* minimizing respectively κ_0 and κ_1

We report the average number of product terms (size) and the average number of T gates for each of the ESOP synthesis methods in Table 1. *PPRM* and *PKRM* are special cases of general ESOP expressions, that can be easily derived from a given Boolean function but are sub-optimal when considering the number of product terms. They are often used as starting covers for ESOP optimization approaches. We report them to enable better comparability of the achieved reduction. *EXORCISM* is a fast cube transformation heuristic, capable of finding close to optimal ESOP expressions, starting from a PKRM cover of the Boolean function. Nevertheless, *EXORCISM* is an heuristic method and does not guarantee the minimality of the solution. In many cases, reducing the size of an ESOP also leads to a reduction of the number of T gates. Consequently, *EXORCISM*, *EXACT(unit)*, and *EXACT(lit)* improve over *PPRM* and *PRKM*. Reducing the number of literals also has a positive effect on the T gates, i.e., *EXACT(lit)* achieves a better reduction than *EXACT(unit)*. Moreover, *EXORCISM* also improves over the *EXACT(unit)* method because its heuristic prefers don't cares over concrete values and reduces the overall number of literals in an ESOP expression.

The histogram in Fig. 5 gives a more detailed overview of the improvement in T-count of *EXACT(lit)* and *EXACT(unit)* over *PKRM*, respectively, for all the 222 representatives in NPN4 equivalent classes.

Optimizing size and literals, however, does not minimize the number of T gates, which we illustrate by example: consider the two equivalent ESOPs

$$C(x_1, x_2, x_3) = 1 \oplus \bar{x}_1 x_2 \oplus x_1 x_2 x_3 \quad \text{and} \quad D(x_1, x_2, x_3) = x_1 x_2 \bar{x}_3 \oplus \bar{x}_2 \oplus \bar{x}_1. \tag{5}$$

Both ESOPs have the same number of product terms and the same number of literals. To realize $C(x_1, x_2, x_3)$ as quantum circuit, however, 23 T gates are required, whereas for realizing $D(x_1, x_2, x_3)$ 16 T gates are needed. This results suggest that in future work it would be valuable to identify more fitting cost functions than the number of literals. In addition, future technology developments could themselves require different cost functions. Our proposed constraint-based method could provide the flexibility to enable future research in this direction.

6.2 Integration into quantum compilation flows

In this section, we show the result of integrating the advanced ESOP optimization methods into the quantum compilation flows *DBS* and *LHRS*.

To integrate optimized ESOP synthesis methods, we propose a pseudo-optimal portfolio approach as described in Alg. 1. For each symmetric control function, the ESOP expression *esop* is computed using the *PKRM* method, that is optimum in this case. If the number of inputs is smaller or equal to 4, we use the exact methods to minimize the number of literals. For larger functions the heuristic *EXORCISM* is used (command &*exorcism -q* of abc [2]).

First we evaluate the improvement of the proposed method integrated into *DBS*(Fig. 2(b)). In Table 2 we show the synthesis results for reversible permutations from Maslov's reversible benchmark[1]. In addition we created reversible functions $MOD_{n/g} : \mathbb{B}^n \to \mathbb{B}^n$, where:

$$MOD_{n/g} = \begin{cases} 0 & \text{if} \quad x = 0 \\ g^x mod(2^n - 1) & \text{if} \quad 1 \leq x \leq 2^n - 2 \\ 2^n - 1 & \text{otherwise} \end{cases}$$

The data are showing a reduction in the number of T gates, with respect to the *PKRM* method, for both the *EXACT* approaches. Nevertheless, we can see how, if the synthesis is performed to minimize the number of literals in each cube, the T-count can be further improved. In fact, the *unit* approach gets to 22.66% improvement, while *lit* gives 28.23% improvement.

In a second experiment, we evaluate the integration into the *LHRS* framework. In Table 3 we show results of synthesizing the arithmetic designs of the EPFL benchmark[2] into quantum circuits. As explained in the preliminary section, the first steps of the flow generate a reversible circuit made of single-target gates, each one with a control function of maximum k inputs, where k is the LUT size used to build the k-LUT network. An ESOP expression is synthesized for each control function and translated into quantum circuits as described in [6, 11]. We compare a flow integrating our pseudo-exact approach against a flow using *PKRM* for the mapping of single-target gates. We report synthesis results for LUT size (k) from 4 to 10. We obtain a maximum reduction of number of T gates in the case of $k = 10$ equal to 36.32% and a minimum reduction in the case of $k = 4$ equal to 17.86%.

7 Open source implementation

The proposed SAT-based exact synthesis method is implemented in the open source C++ library *easy*[3] [19, 24] using our own C++ implementation of R-C2 [8] as MAX-SAT oracle. The *easy* library provides implementations of various verification and synthesis algorithms for ESOP expressions.

[1] `http://webhome.cs.uvic.ca/~dmaslov`
[2] `https://github.com/lsils/benchmarks`
[3] `https://github.com/hriener/easy`

Algorithm 1: Pseudo-exact optimal ESOP

```
input  : control function f : 𝔹^n → 𝔹
output: optimized ESOP expression of f
begin
    if f ∈ cache then
        return cache[f]
    if f is symmetric then
        esop ← PKRM(f)
    else if n ≤ 4 then
        esop ← EXACT_LIT(f)
    else
        esop ← EXORCISM(f)
    cache.insert(f, esop)
    return esop
```

Table 2. Comparison between exact method and heuristic for small reversible functions

Permutation		PKRM		EXACT(lit)		EXACT(unit)	
	Q	T	t[s]	T	t[s]	T	t[s]
hwb4	4	123	0.0	109	0.1	116	0.0
hwb5	5	514	0.0	337	59.9	447	0.3
hwb6	6	1361	0.0	993	0.9	993	0.9
hwb7	7	5331	0.0	3066	1.0	3066	1.1
hwb8	8	13562	0.0	7654	1.2	7654	1.2
mod5_11	5	453	0.0	350	36.5	368	0.2
mod5_12	5	453	0.0	361	59.1	400	0.2
mod5_13	5	428	0.0	329	38.2	343	0.1
mod5_17	5	478	0.0	382	64.3	414	0.3
mod5_21	5	433	0.0	352	34.9	482	0.1
mod5_22	5	469	0.0	354	25.0	391	0.1
mod5_24	5	503	0.0	405	61.4	448	0.3
mod5_3	5	494	0.0	386	34.8	411	0.2
mod7_14	7	5201	0.0	2936	1.0	2936	1.0
mod7_3	7	4945	0.0	2957	1.0	2957	1.0
mod7_7	7	4859	0.0	3039	1.0	3039	1.0
prime4	4	102	0.0	95	0.0	106	0.0
prime5	5	367	0.0	271	28.5	289	0.1
prime6	6	1054	0.0	786	0.8	786	0.7
prime7	7	3600	0.0	2283	1.0	2283	0.9
prime8	8	8302	0.0	4420	1.1	4420	1.0

avg. reduction *EXACT(lit) = 28.23%*
avg. reduction *EXACT(unit) = 22.66%*

For the quantum compilation results, we interfaced *easy* with *caterpillar*[4] and *tweedledum*[5]. The first library is dedicated to quantum compilation, hierarchical methods, and quantum memory management, whereas the second library implements state-of-the-art synthesis methods, e.g., Young subgroup decomposition based synthesis.

[4] `https://github.com/gmeuli/caterpillar`
[5] `https://github.com/boschmitt/tweedledum`

Table 3. Synthesis of EPFL arithmetic benchmark on

			PKRM		Opt.				PKRM		Opt.	
k		Q	T	t[s]	T	t[s]		Q	T	t[s]	T	t[s]
4	adder	511	5398	0.0	5356	0.4	bar	1415	76816	0.2	56320	1.8
5		448	16061	0.1	15151	0.5		1031	95576	0.3	63694	2.9
6		448	16271	0.1	15279	0.6		647	52750	0.2	50944	1.8
7		427	37259	0.1	36110	0.7		647	52750	0.3	50944	1.9
8		427	37963	0.1	36654	0.7		647	52750	0.3	50944	1.9
9		416	84076	0.2	72338	0.8		647	52750	0.3	50944	1.9
10		416	85509	0.2	72985	0.9		647	52750	0.3	50944	1.9
4	div	26467	757193	5.8	635999	12.4	hyp	64630	2448872	25.3	2208000	37.5
5		24474	851035	6.8	690622	15.1		56568	2647894	26.1	2156087	40.5
6		24083	876636	8.0	709586	19.0		50118	2860466	28.2	2145634	46.6
7		23944	939887	9.6	742327	23.8		48399	3501767	31.0	2817812	51.8
8		23808	1034583	11.2	773058	26.6		47581	4540244	36.9	3546120	66.7
9		23711	1204407	13.0	831482	30.5		46992	5379295	43.0	4158260	79.1
10		23633	1710038	15.4	875766	34.3		46933	6238649	50.0	4596940	94.4
4	log	10420	458335	2.4	380787	12.8	max	1484	54422	0.2	42684	5.4
5		9661	623957	3.2	492501	24.1		1346	76507	0.2	60597	6.4
6		8156	1033225	4.3	768429	49.4		1256	104109	0.3	79853	6.4
7		8141	1507690	5.1	883462	103.7		1149	148355	0.4	102310	6.0
8		4658	2196359	6.2	1228593	48.1		1067	209851	0.6	140106	6.9
9		4456	3393095	8.4	1912337	65.8		977	323027	0.8	200270	5.9
10		3697	5786642	10.8	3268408	74.8		929	355341	1.1	230118	5.6
4	mult	8194	359422	1.8	270268	6.2	sin	1962	71409	0.4	64103	14.5
5		8100	479930	2.2	368062	8.8		1818	82386	0.5	71471	19.4
6		6706	1034190	2.8	579420	11.6		1608	115107	0.7	92659	25.7
7		7050	1448336	3.7	847558	15.2		1553	137989	0.9	104092	27.3
8		5101	1371054	3.7	818914	16.6		1449	249964	1.2	157332	32.5
9		5165	2115333	5.2	1410009	18.3		915	794521	1.5	362082	33.2
10		4006	3657831	8.0	2417393	23.9		878	1241237	2.2	542136	37.2
4	sqrt	8686	317522	1.7	255275	6.4	square	6909	354552	1.5	240636	11.5
5		8351	344049	2.3	265948	6.8		6092	553311	1.9	308262	18.0
6		8332	391900	2.9	285310	7.8		4195	299574	1.8	206683	16.1
7		8152	448518	3.4	301246	8.4		4213	368160	2.0	261272	22.8
8		7986	709282	4.4	358215	9.5		3764	477446	2.3	341092	24.3
9		7976	720144	5.3	359296	10.6		3724	658343	2.9	445505	32.5
10		7966	1413589	7.0	540586	12.4		3792	876884	3.7	532124	41.4

min avg. improvement k=4 : 17.86 %
max avg. improvement k=10 : 36.32 %
avg. improvement: 26.36 %

8 Conclusion

In this work, we integrate ESOP synthesis methods into quantum compilation flows in order to improve the quality of the produced quantum circuits. We target fault-tolerant quantum computing and aim at minimizing the number of expensive T gates. We consider two different compilation flows for Boolean functions that make use of ESOP-based reversible synthesis.

For both frameworks this integration leads to promising results, which show maximum T-count reductions of 28.23% in *DBS* and 36.32% in *LHRS* with respect to *PKRM*. In conclusion, advanced ESOP synthesis methods, both exact and heuristic, can be applied inside the quantum compilation flows that use

ESOP-based reversible synthesis, to generate better circuits for fault-tolerant quantum computing.

Acknowledgments This research was supported by the European COST Action IC 1405 'Reversible Computation', by the EPFL Open Science Fund and the Institutional Strategy of the University of Bremen, funded by the German Excellence Initiative.

References

1. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **32**(6), 818–830 (2013)
2. Brayton, R., Mishchenko, A.: ABC: An academic industrial-strength verification tool. In: International Conference on Computer Aided Verification. pp. 24–40. Springer (2010)
3. De Vos, A., Van Rentergem, Y.: Young subgroups for reversible computers. Advances in Mathematics of Communications **2**(2), 183–200 (2008)
4. Drechsler, R.: Pseudo-kronecker expressions for symmetric functions. IEEE Transactions on Computers **48**(9), 987–990 (1999)
5. Drechsler, R., Finder, A., Wille, R.: Improving ESOP-based synthesis of reversible logic using evolutionary algorithms. In: European Conference on the Applications of Evolutionary Computation. pp. 151–161. Springer (2011)
6. Fazel, K., Thornton, M., Rice, J.E.: ESOP-based Toffoli gate cascade generation. In: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing. pp. 206–209 (2007)
7. Haener, T., Soeken, M., Roetteler, M., Svore, K.M.: Quantum circuits for floating-point arithmetic. In: International Conference on Reversible Computation. pp. 162–174. Springer (2018)
8. Ignatiev, A., Morgado, A., Marques-Silva, J.: PySAT: A Python toolkit for prototyping with SAT oracles. In: Theory and Applications of Satisfiability Testing. pp. 428–437 (2018)
9. JavadiAbhari, A., Patil, S., Kudrow, D., Heckey, J., Lvov, A., Chong, F.T., Martonosi, M.: ScaffCC: A framework for compilation and analysis of quantum computing programs. In: Proceedings of the 11th ACM Conference on Computing Frontiers. p. 1. ACM (2014)
10. Li, C.M., Manyà, F.: MaxSAT, hard and soft constraints. In: Handbook of Satisfiability, pp. 613–631 (2009)
11. Maslov, D.: Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization. Physical Review A **93**(2), 022311 (2016)
12. Miller, D.M., Wille, R., Drechsler, R.: Reducing reversible circuit cost by adding lines. In: 2010 40th IEEE International Symposium on Multiple-Valued Logic. pp. 217–222. IEEE (2010)
13. Mishchenko, A., Chatterjee, S., Brayton, R.K.: Improvements to technology mapping for LUT-based FPGAs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **26**(2), 240–253 (2007)
14. Mishchenko, A., Perkowski, M.: Fast heuristic minimization of exclusive-sums-of-products. Proc. Int. Workshop Appl. ReedMuller Expansion Circuit Des. pp. 242–250 (2001)

15. Mizuki, T., Otagiri, T., Sone, H.: An application of ESOP expressions to secure computations. Journal of Circuits, Systems, and Computers **16**(02), 191–198 (2007)
16. Papakonstantinou, K., Papakonstantinou, G.: A nonlinear integer programming approach for the minimization of boolean expressions. Journal of Circuits, Systems and Computers **27**(10), 1850163 (2018)
17. Perkowski, M., Chrzanowska-Jeske, M.: An exact algorithm to minimize mixed-radix exclusive sums of products for incompletely specified Boolean functions. In: ISCAS. pp. 1652–1655 (1990)
18. Rawski, M.: Application of functional decomposition in synthesis of reversible circuits. In: International Conference on Reversible Computation. pp. 285–290. Springer (2015)
19. Riener, H., Ehlers, R., Schmitt, B., De Micheli, G.: Exact synthesis of ESOP forms. CoRR **abs/1807.11103** (2018), `http://arxiv.org/abs/1807.11103`
20. Sasao, T.: EXMIN2: A simplification algorithm for exclusive-or-sum-of-products expressions for multiple-valued-input two-valued-output functions. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **12**(5), 621–632 (1993)
21. Sasao, T.: Representations of logic functions using EXOR operators. In: Representations of discrete functions, pp. 29–54. Springer (1996)
22. Soeken, M., Haener, T., Roetteler, M.: Programming quantum computers using design automation. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018. pp. 137–146. IEEE (2018)
23. Soeken, M., Mozafari, F., Schmitt, B., De Micheli, G.: Compiling permutations for superconducting QPUs. In: DATE. p. To appear (2019)
24. Soeken, M., Riener, H., Haaswijk, W., Micheli, G.D.: The EPFL logic synthesis libraries. CoRR **abs/1805.05121** (2018), `http://arxiv.org/abs/1805.05121`
25. Soeken, M., Roetteler, M., Wiebe, N., De Micheli, G.: Logic synthesis for quantum computing. CoRR **abs/1706.02721** (2017), `http://arxiv.org/abs/1706.02721`
26. Soeken, M., Roetteler, M., Wiebe, N., De Micheli, G.: LUT-based hierarchical reversible logic synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2018)
27. Stergiou, S., Daskalakis, K., Papakonstantinou, G.: A fast and efficient heuristic ESOP minimization algorithm. In: Proceedings of the 14th ACM Great Lakes symposium on VLSI. pp. 78–81. ACM (2004)
28. Wille, R., Soeken, M., Otterstedt, C., Drechsler, R.: Improving the mapping of reversible circuits to quantum circuits using multiple target lines. In: 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC). pp. 145–150. IEEE (2013)
29. Zhegalkin, I.: The technique of calculation of statementsin symbolic logic. In: Mathe. Sbornik. vol. 34, pp. 9–28 (1927), (in Russian)