Chapter 8
Exact Synthesis of ESOP Forms

Heinz Riener, Rüdiger Ehlers, Bruno de O. Schmitt, and Giovanni De Micheli

8.1 Introduction

In the design of *Very Large-Scale Integration* (VLSI) systems, two-level logic representations are classically used to represent and manipulate Boolean functions. *Exclusive-or Sum-of-Products* (ESOP) is a two-level normal form representation of a Boolean function that consists of one level of multi-input AND gates followed on the next level by one multi-input XOR gate. ESOP forms play an important role in logic synthesis due to their improved compactness for arithmetic or communication circuits with respect to other two-level representations [26] and their excellent testability properties [13]. The inherent reversibility of the XOR operation, moreover, makes ESOP forms particularly suitable in applications such as security [16, 21] or quantum computation [8].

The ESOP representation of a Boolean function is not unique, i.e., the same Boolean function can be expressed as multiple structurally different, but semantically equivalent ESOP forms. In practice, it is important to find a small representation of an ESOP form to reduce the overall costs for realizing it in hardware or implementing it in software. The problem of synthesizing an ESOP form for a given Boolean function is to identify a set of product terms over the Boolean variables of the function such that each minterm in the OFF-set of the function is covered by the product terms an even number of times and each minterm in the ON-set of the Boolean function is covered an odd number of times.

H. Riener (✉) · B. d. O. Schmitt · G. De Micheli
EPFL, Lausanne, Switzerland
e-mail: heinz.riener@epfl.ch

R. Ehlers
University of Bremen, Bremen, Germany

R. Drechsler, M. Soeken (eds.), *Advanced Boolean Techniques*,
https://doi.org/10.1007/978-3-030-20323-8_8

Finding ESOP forms with a small or a minimal number of product terms is hard and numerous exact and heuristic synthesis methods [11, 19, 22, 23, 25, 30] for solving this problem have been proposed. Heuristic methods focus on finding small (but not necessarily minimal) ESOP forms; they are fast, but only examine a subset of the possible search space. Heuristic methods, e.g., the Exorcism approach [19], usually operate in two phases. In the first phase, an ESOP form with a sub-optimal number of product terms is derived from the Boolean function, e.g., by translating each minterm of the Boolean function into one product term or translating the function into special cases of ESOP forms such as Pseudo-Kronecker Expressions [6]. In the second phase, the ESOP form is iteratively optimized and reshaped using cube transformations with the overall goal of merging as many product terms as possible. The cube transformations are applied to each pair of product terms that potentially lead to merging them or with other product terms of the ESOP form. The second phase terminates when, after several iterations, no further size reduction is achieved. Heuristic methods produce small ESOP forms in reasonable time, but suffer from local minima that cannot easily be escaped. In contrast, exact methods find an "exact" ESOP form, i.e., an ESOP form with a minimal number of product terms, but either require to store large tables of pre-computed information [22, 25] or suffer from long runtimes [23]. For instance, the tabular-based methods described by Gaidukov [11] or Papakonstantinou [22] require pre-computed tables of all exact ESOP forms for Boolean functions over $n - 1$ Boolean variables to derive an exact ESOP form for a Boolean function over n Boolean variables. Due to the exponential growth of the number of Boolean functions in the number of Boolean variables, these methods become too time and memory consuming when $n > 6$. Alternative exact synthesis approaches such as a recent formulation of the ESOP synthesis problem using non-linear programming [23] can take several minutes for synthesizing a single exact ESOP form.

Until today, a large gap between the number of product terms optimized with heuristic methods and exact methods remains. Where exact methods hardly can deal with more than 8 Boolean variables and a few product terms, heuristic methods nowadays, e.g., in the quantum domain, have to deal with the optimization of ESOP forms with 10^5 or 10^6 products terms over 16 and more Boolean variables [27]. Our experiments with large-scale ESOP forms showed that heuristic optimization method can often achieve a reduction of $50-80\%$ in the number of ESOP terms with respect to the size of the initial ESOP form. Due to the large combinational search space of the ESOP synthesis problem, lower bounds on the number of required product terms are only known for Boolean functions with a few Boolean variables, such that the capabilities of ESOP optimization techniques remain unclear.

In this paper, we investigate the exact synthesis of ESOP forms using Boolean satisfiability (SAT). SAT-based approaches are very successful on a variety of different verification and synthesis problems. We present an exact synthesis approach for computing ESOP forms with a minimal number of product terms. Starting from a specification in form of a possibly incompletely specified Boolean function, our approach iteratively constructs a Boolean constraint satisfaction problem that is

satisfiable if and only if an ESOP form with k (initially $k = 1$) product terms that implements the specification exists. The problem is then solved utilizing a SAT-solver and, if satisfiable, an ESOP form with k product terms is returned. Otherwise, if unsatisfiable, k is increased and the synthesis process is restarted. The synthesis approach is hardly affected by the number of Boolean variables and particularly fast if the Boolean function can be expressed by using only a few product terms. We argue that such a SAT-based exact synthesis procedure can be a backbone of a new generation of heuristic ESOP optimization methods that, instead of relying on cube transformations applied to a pair of product terms, are capable of optimizing small subsets (windows) of product terms.

The proposed approach is the first ESOP synthesis technique based on Boolean satisfiability. We further present a relaxation of the technique to compute ESOP forms with size close to minimal leveraging the SAT-solver's conflict limit. We have implemented SAT-based exact synthesis for ESOPs and the relaxation of the approach using an off-the-shelf SAT-solver and show in the experiments that SAT-based ESOP synthesis can be readily used to synthesize ESOP forms with up to 8 Boolean variables and up to 100 terms. As benchmarks, we use completely specified Boolean functions that are used as representatives of the NPN4 equivalence classes [12] as well as completely specified Boolean functions that appeared in technology mapping using look-up tables (LUTs) with at most 8 inputs (8-LUT mapping). Moreover, we use a set of randomly generated incompletely specified Boolean functions with up to 8 Boolean variables.

8.2 Background

Exclusive-or Sum-of-Products (ESOP) Let $\mathbb{B} = \{0, 1\}$ and $\mathbb{B}_3 = \{0, 1, -\}$ with the third element "$-$" which denotes don't care. An *ESOP form* in n Boolean variables $x_1, \ldots, x_n$ is a Boolean expression

$$\bigoplus_{j=1}^{k} \left(\bigwedge_{i=1}^{n} x_i^{l_{i,j}} \right), \tag{8.1}$$

where the operators $\oplus$ and $\wedge$ denote standard addition (XOR) and multiplication (AND) in the Galois field with two-elements, respectively, each $l_{i,j} \in \mathbb{B}_3$ is a constant and each expression

$$x_i^{l_{i,j}} = \begin{cases} \bar{x}_i, & \text{if } l_{i,j} = 0 \\ x_i, & \text{if } l_{i,j} = 1 \\ 1, & \text{if } l_{i,j} = - \end{cases} \tag{8.2}$$

for $1 \leq i \leq n$ and $1 \leq j \leq k$. We say that k is the *size* of the ESOP form and call each conjunction $x_1^{l_{1,j}} \cdots x_n^{l_{n,j}}$, $1 \leq j \leq k$, that appears in the ESOP form a *product term*. The Boolean expression in (8.1) is often compactly notated as a list of words

$$l_{1,1} \cdots l_{n,1} \quad l_{1,2} \cdots l_{n,2} \quad \ldots \quad l_{1,k} \cdots l_{n,k}, \tag{8.3}$$

where each word $l_{1,j} \cdots l_{n,j}$ is of fixed length n.

Distance of Product Terms Suppose that

$$u = x_1^{l_{1,p}} \cdots x_n^{l_{n,p}} \quad \text{and} \quad v = x_1^{l_{1,q}} \cdots x_n^{l_{n,q}} \tag{8.4}$$

are two product terms in n Boolean variables. We define the *distance* $d(u, v)$ of u and v as the number of different $l_{i,j}$ for $1 \leq i \leq n$ and $j \in \{p, q\}$, i.e.,

$$d(u, v) = \sum_{i=1}^{n} [l_{i,p} \neq l_{i,q}], \tag{8.5}$$

where [.] denote the Iverson brackets. We say if $d(u, v) = m$, then u and v have distance m or are m-distant.

ESOPs Describing Boolean Functions An ESOP form semantically describes a (single-output) Boolean function $f : \mathbb{B}^n \to \mathbb{B}$, which maps assignments of the Boolean variables $x_1, \ldots, x_n \in \mathbb{B}$ to truth values $f(x_1, \ldots, x_n) \in \mathbb{B}$. Each assignment to all Boolean variables $x_1, \ldots, x_n$ is called a *minterm* and can be interpreted as the decimal number $\sum_{i=1}^{n} x_i 2^{i-1}$ when read as $(x_n \cdots x_1)_2$.

A completely specified Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ over n Boolean variables can be uniquely represented as a truth table, i.e., a word $b_{2^n S} \cdots b_1$ of length 2^n, where $b_j = f(j - 1)$ for $1 \leq j \leq 2^n$. An incompletely specified Boolean function $g : \mathbb{B}^n \to \mathbb{B}_3$ can be represented by two completely specified Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$ and $c : \mathbb{B}^n \to \mathbb{B}$, where $f(x) = [g(x) = 1]$ and $c(x) = [g(x) \neq -]$. We call c the *care function* of g.

Two ESOP forms are semantically *equivalent* if they describe the same Boolean function. An ESOP form with size k is *minimal* if and only if no semantically equivalent ESOP form with fewer product terms exists. Minimal ESOP forms are in general not unique.

8.3 SAT-Based Exact ESOP Synthesis

8.3.1 Exact Synthesis of ESOP Forms

Objective We aim for synthesizing minimal ESOP forms in n Boolean variables when a completely specified Boolean function or incompletely specified Boolean

function is provided as specification. In case of completely specified Boolean functions, this objective can be formally described as follows: given a single-output Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ over n Boolean variables $x_1, \ldots, x_n$, find an integer k and constants $l_{i,j} \in \mathbb{B}_3$ for $1 \leq i \leq n$ and $1 \leq j \leq k$ such that

$$\bigoplus_{j=1}^{k} \left(\bigwedge_{i=1}^{n} x_i^{l_{i,j}} \right) = f(x_1, \ldots, x_n) \text{ for all } x_1, \ldots, x_n \in \mathbb{B}^n \tag{8.6}$$

and k is minimal. The case of incompletely specified Boolean functions can be addressed similarly to (8.6).

Example 8.1 As an introductory example, consider the incompletely specified Boolean function described by the truth table `0x688C802028222222`[1] over 6 Boolean variables with care function `0x6AAEFF3FFEBFEAA6`. A minimal ESOP form, for instance, is

$$\bar{x}_1 x_3 \bar{x}_4 \bar{x}_5 x_6 \oplus \bar{x}_1 x_2 \bar{x}_3 x_5 \bar{x}_6 \oplus \bar{x}_1 \bar{x}_3 \bar{x}_4 \bar{x}_6 \oplus \bar{x}_2 \bar{x}_5 \bar{x}_6 \oplus \bar{x}_1 x_2 x_6, \tag{8.7}$$

which requires 5 product terms and can be equivalently written as

```
0-1001  0-00-0  -0--00  010-10  01---1.
```
(8.8)

In general, minimal ESOPs are not unique. The same Boolean function may also be represented as the ESOP form

```
0-1001  0100-0  -0--00  0-0-10  01---1
```
(8.9)

or

```
0-1001  0-00-0  ----00  011-10  01----.
```
(8.10)

Finding minimal ESOP forms is, due to the large combinational search space, a challenging problem. In [23], a minimal ESOP form for the Boolean function in the previous example was found on average in roughly 668.22 s using integer non-linear programming using different starting points and Matlab as a solving engine. The authors, moreover, point out that decomposition-based ESOP synthesis approaches, e.g., [25], require up to 4 h for synthesizing minimal ESOP forms for incompletely specified Boolean functions over 6 Boolean variables.

[1] We use hexadecimal notation to shorten the string representation of the (binary) truth tables of Boolean functions.

8.3.2 SAT-Based Exact Synthesis Procedure

In this section, we propose a SAT-based exact synthesis approach for ESOP forms. The approach is based on ideas from Knuth [15] (originally proposed by Kamath et al. [14]) and our previous work on learning two-level patches to correct combinational Boolean circuits [24]. Our approach synthesizes an ESOP form for the Boolean function in Example 8.1 in less than a second. We formalize the search problem as a series of Boolean constraint satisfaction problems—one for each possible ESOP size k (starting with $k = 1$) and employ a decision procedure for Boolean satisfiability to decide the satisfiability of the constraints. The constraints are constructed in such a way that they are satisfiable if and only if an ESOP form with k product terms exists and each satisfying assignment corresponds to an ESOP form with k product terms. If the constraints are unsatisfiable, then no ESOP form restricted to k product terms, that is equivalent to the provided Boolean function, exists. By systematically solving the constraint satisfaction problem for increasing values of the size parameter k, a minimal ESOP form is guaranteed to be found.

Formulation of the Constraint Satisfaction Problem Suppose that $f : \mathbb{B}_3^n \to \mathbb{B}$ is a (single-output) Boolean function over n Boolean variables. We formulate the problem of finding an ESOP form equivalent to f with k product terms as a constraint satisfaction problem in propositional logic using $2nk$ Boolean variables, $p = p_{1,1}, \ldots, p_{k,n}$ and $q = q_{1,1}, \ldots, q_{k,n}$, where n is the number of Boolean variables of f, k is the size of the ESOP form, and

$$p_{j,l} = [x_l \text{ in product term } j] \quad \text{and} \quad q_{j,l} = [\bar{x}_l \text{ in product term } j] \tag{8.11}$$

for $1 \leq j \leq k$ and $1 \leq l \leq n$.

For each assignment $x_1 \cdots x_n \in \mathbb{B}_3^n$ of the Boolean function f with the corresponding output value $f(x_1, \ldots, x_n) = b$, we introduce k auxiliary Boolean variables $z = z_1, \ldots, z_k$ and add $k \cdot n + k$ clauses

$$\bigwedge_{j=1}^{k} \bigwedge_{l=1}^{n} \left(\bar{z}_j \vee \mathrm{ITE}(x_i, \bar{q}_{j,l}, \bar{p}_{j,l})\right) \quad \text{and} \quad \bigwedge_{j=1}^{k} \left(z_j \vee \bigvee_{l=1}^{n} \mathrm{ITE}(x_i, q_{j,l}, p_{j,l}) \right), \tag{8.12}$$

which ensure that if and only if $z_j = 1$, then the j-th product term evaluates to 1 for assignment $x_1 \cdots x_n$. The *if-then-else*-operator is defined as

$$\mathrm{ITE}(x_i, v_{j,l}, u_{j,l}) = \begin{cases} v_{j,l}, & \text{if } x_i = 1 \\ u_{j,l}, & \text{if } x_i = 0 \\ false, & \text{otherwise} \end{cases} \tag{8.13}$$

with $v_{j,l} \in \{q_{j,l}, \bar{q}_{j,l}\}$ and $u_{j,l} \in \{p_{j,l}, \bar{p}_{j,l}\}$, respectively. One additional XOR-constraint

$$\left(\bigoplus_{j=1}^{k} z_j \right) = b \tag{8.14}$$

per assignment guarantees that an odd number of z_js evaluates to 1 if $b = 1$ and an even number if $b = 0$.

This constraint satisfaction problem is satisfiable if and only if an ESOP form of size k exists and each satisfying assignment $\hat{p}_{1,1}, \ldots, \hat{p}_{k,n}$ and $\hat{q}_{1,1}, \ldots, \hat{q}_{k,n}$ corresponds to one possible implementation.

Translating XOR-Constraints to CNF All XOR-constraints in the constraint satisfaction problem are, by construction, formulated over disjoint sets of Boolean variables such that techniques like Gaussian elimination are not effective. Instead, we translate each XOR-constraint first into an equivalent XOR-clause by flipping one of the Boolean variables if and only if $b = 0$, i.e.,

$$(z_1 \oplus \cdots \oplus z_k) = b \quad \Longrightarrow \quad \begin{cases} z_1 \oplus \cdots \oplus z_k, & \text{if } b = 1 \\ z_1 \oplus \cdots \oplus \bar{z}_k, & \text{if } b = 0. \end{cases} \tag{8.15}$$

Then, we select two literals l_a, l_b from the XOR-clause and apply the Tseitin transformation to generate four clauses $(\bar{z}_a \vee \bar{z}_b \vee \bar{u})$, $(z_a \vee z_b \vee \bar{u})$, $(z_a \vee \bar{z}_b \vee u)$, $(\bar{z}_a \vee z_b \vee u)$ with the newly introduced Boolean variable u and repeat this process until only one literal is left which is added as a unit clause.

SAT-Based Exact ESOP Synthesis The overall exact synthesis procedure is sketched in Algorithm 3. The function `MakeCSP` constructs the constraint satisfaction problem φ in the Boolean variables p, q, z for a given Boolean function f and size parameter k as described above. The function `SAT` refers to the invocation of a decision procedure for the Boolean satisfiability problem, usually called a *SAT-solver*, and is assumed to decide the satisfiability of φ and, if satisfiable, to also provide a satisfying assignment $\hat{p}$ and $\hat{q}$ for variables p and q. The

Algorithm 3 SAT-based exact ESOP synthesis

input : a (possibly incompletely-specified) Boolean function f
output: a minimal ESOP functionally equivalent to f

```
for k ← 1, 2, ... do
    φ(p, q, z) ← MakeCSP(k, f);
    if p̂, q̂ ⊨ SAT(∃z : φ(p, q, z)) then
        return MakeESOP(p̂, q̂);
    end
end
```

Algorithm 4 SAT-based exact synthesis guided by counterexamples

input : a (possibly incompletely-specified) Boolean function f
output: a minimal ESOP r functionally equivalent to f

```
r ← ϵ;
k ← 1;
φ(p, q, z) ← true;
while m ← NotEquivalent(f,r) do
    φ ← AddConstraints(φ,m);
    if p̂, q̂ ⊨ SAT(∃z : φ(p, q, z)) then
        r ← MakeESOP(p̂,q̂);
    else
        r ← ϵ;
        k ← k + 1;
        φ(p, q, z) ← true;
    end
end
return r;
```

assignment to the intermediate Boolean variables z is for the construction of no further interest and not returned. Finally, the function `MakeESOP` constructs an ESOP form from the assignment $\hat{p}$ and $\hat{q}$ according to the rules described in (8.11). Note that Algorithm 3 always terminates, but may run out of resources (memory or time) if the minimal ESOP requires many product terms. Thus in practice usually an additional termination criterion in form of an upper bound for the size parameter k or maximum number of conflicts examined by the SAT-solver is provided.

Counterexample-Guided Abstraction-Refinement Algorithm 3 synthesizes an ESOP form in one step. Alternatively, counterexample-guided abstraction-refinement can be employed as shown in Algorithm 4. The idea of the abstraction-refinement loop is to iteratively update a candidate ESOP form r (starting from the empty ESOP form ϵ) until it eventually becomes semantically equivalent to the Boolean function f to be synthesized. In each iteration, the constraints of one assignment $x = x_1 \cdots x_n$ for which r and f evaluate differently ($r(x) \neq f(x)$) are added (`AddConstraints`) to the constraint satisfaction problem and r is resynthesized. If φ becomes unsatisfiable, then the constraints cannot be solved within the current restriction to k product terms and k needs to be relaxed. If f and r are equivalent, i.e., no counterexample $x = x_1 \cdots x_n$ is found by `NotEquivalent`, then r is returned as an ESOP form semantically equivalent to f. The main advantage of Algorithms 4 over 3 lies in its ability to abstract from unnecessary constraints which keeps the constraint satisfaction problem as small as possible. The algorithm is fast mainly because modern backtrack search-based SAT-solvers support *incremental solving* [7] and are able to maintain learned information when new constraints are added to a satisfiability problem. The oracle `NotEquivalent` has to be capable of verifying whether a candidate ESOP form r is functionally equivalent to the Boolean function f. For Boolean functions with up to 16 Boolean variables, simulation using explicit representations such as truth

tables can be done very quickly. For Boolean functions with more than 16 Boolean variables, a BDD- or SAT-based procedure can be employed.

8.3.3 Extensions and Variations

Downward vs. Upward Search Algorithm 4 describes an upward search procedure to find a minimal ESOP form starting with 1 term. This approach can be easily modified into a downward search by starting from a maximum number of terms $\hat{k}$ and iteratively decreasing the number of terms by 1 as long as the constraint system is satisfiable. If the constraint system becomes unsatisfiable for a certain number k of terms, the previous $k + 1$ terms correspond to a minimal ESOP form. In practice downward and upward search procedures are useful. An upward search procedure is fast if the expected minimal k is small. Otherwise, proving unsatisfiability with a SAT-solver becomes too time consuming. A downward search procedure is fast if the expected minimal k is close to the initially provided term limit $\hat{k}$.

Conflict Limit For a SAT-solver proving unsatisfiability of a set of constraints, i.e., showing that no assignment exists that satisfies the constraints, often requires labor-intensive analysis. If the search space is sufficiently large, these proofs are often not completed within reasonable time. Most modern SAT-solver provides a conflict limit to allow a user to specify a maximum number of possible solving attempts. If the SAT-solver is unable to find a satisfying assignment within the given conflict limit, the solver reports "unknown" as solution. In this case, the synthesis algorithm can choose to increase or decrease the current k, hoping that the next k is easier to solve because the corresponding constraint system is less or more constrained, respectively. When a conflict limit is employed in Algorithm 4, due to the possible "unknown" solutions, a minimal ESOP form may not be found. However, in case of a downward search, which systematically decreases k, an intermediate "unknown" solution for k_1 can be safely ignored if the constraint system is later proved satisfiable for $k_2 < k_1$, whereas in case of an upward search, an intermediate "unknown" solution for k_1 can be ignored if the constraint system is proved unsatisfiable for a later $k_2 > k_1$.

8.4 ESOP Synthesis for Quantum Computation

ESOP-based logic synthesis and optimization techniques have recently attracted interest due to their application for quantum computing, where ESOP forms are used as an intermediate representation to map Boolean functions into quantum circuits [20]. The appeal of the idea stems from the fact that, in contrast to other mapping approaches, ESOP-based synthesis does not introduce additional garbage

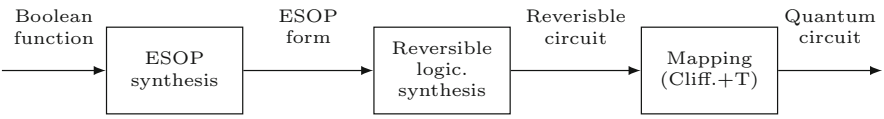

Fig. 8.1 ESOP-based synthesis flow from a Boolean function to a quantum circuit

outputs (often called *ancillæ*) and, consequently, can be realized with fewer qubits—a highly critical resource on today's quantum computers.

In this section, we survey ESOP-based synthesis for quantum circuits. We describe how an ESOP form can be mapped to a quantum circuit and show by example that optimizing ESOP forms has a positive effect on the cost functions for realizing them.

Reversible Logic Synthesis for Quantum Computing Figure 8.1 illustrates an ESOP-based synthesis flow that stepwisely transforms a Boolean function into a quantum circuit leveraging ESOP forms and reversible logic circuits as intermediate representations. The so-called *ESOP-based reversible logic synthesis* [20] has proven effective while keeping the number of extra qubits required for transforming the ESOP form as low as possible. In fact, for translating an ESOP form with n Boolean variables, we present a construction that requires at most $n + 2$ qubits.

We describe reversible logic circuits in terms of reversible gates using a formalism introduced by Toffoli and Fredkin [10]. Given a fixed set $X = x_1, \ldots, x_n$ of Boolean variables, a *(mixed-polarity multiple-controlled) Toffoli gate* is a pair (C, x_t) of *control lines* $C \subset \{x, \bar{x} \mid x \in X\}$ and a *target line* $x_t \in X$ with

$$\{x, \bar{x}\} \not\subset C \text{ for all } x \in X \qquad \text{and} \qquad \{x_t, \bar{x}_t\} \cup C = \emptyset. \tag{8.16}$$

Each Toffoli gate defines a bijective Boolean function $g : \mathbb{B}^n \to \mathbb{B}^n$

$$(x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_{t-1}, x_t \oplus f(x_1, \ldots, x_n), x_{t+1}, \ldots, x_n), \tag{8.17}$$

with *control function* $f : \mathbb{B}^{n-1} \to \mathbb{B}$

$$(x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_n) \mapsto \bigwedge_{c \in C} c. \tag{8.18}$$

The reversible gate flips the Boolean value on the target line if the control function f evaluates to true for the values observed on the control lines.

A *reversible logic circuit* is a cascade of Toffoli gates and the function defined by the reversible logic circuit is the composition function of the individual functions defined by its reversible gates. We use a graphical notation based on Feynman [9] to denote reversible circuits as diagrams. Figure 8.2 illustrates the graphical notation: on the top-left, the figure shows one reversible single-target gate with an arbitrary

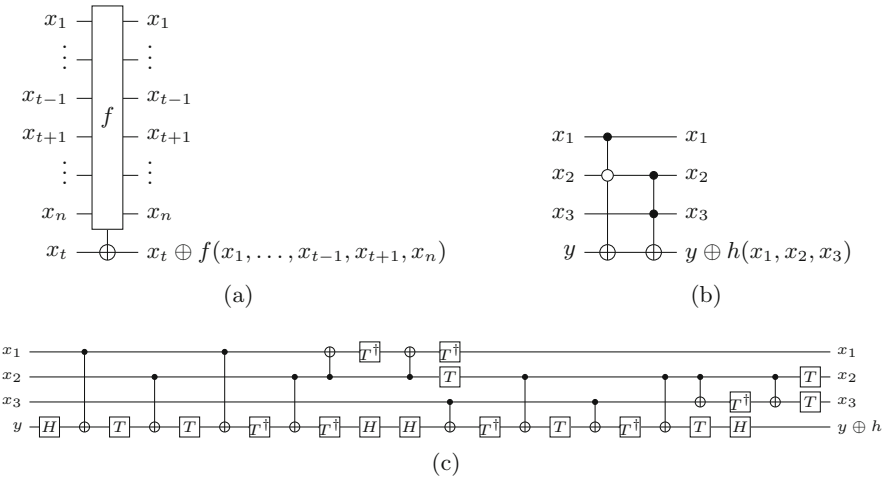

Fig. 8.2 Graphical notation of reversible logic circuits and quantum circuits. (**a**) Toffoli gate. (**b**) Reversible circuit. (**c**) Quantum circuit

control function f; on the top-right, a concrete example of a reversible circuit is given consisting of the cascade of the two Toffoli gates $(\{x_1, \bar{x}_2\}, x_5)$ and $(\{x_2, x_3\}, x_5)$, where the composition function $h(x_1, x_2, x_3) = x_1\bar{x}_2 \oplus x_2x_3$ of the two gates can be observed on line y. In this notation, the line with the $\oplus$ denotes the target line, whereas black and white dots denote positive and negative control lines, respectively. On the bottom, a quantum circuit is shown for the same example. We show this example for completeness, but will not discuss the graphical notation of quantum circuits (see, e.g., Soeken et al. [27] for details).

Mapping ESOP forms into reversible circuits is straightforward. An ESOP form $c_1 \oplus \cdots \oplus c_k$ with k product terms and n Boolean variables is functionally equivalent to a reversible circuit with at most $n + 2$ lines and k Toffoli gates, where the control function of each gate is exactly one c_i for $1 \leq i \leq k$. Since each Toffoli gate is reversible, the concrete order of the Toffoli gates does not matter.

Next, Toffoli gates are mapped into a quantum gate library [18]. In this paper, we focus on the universal fault-tolerant quantum gate library *Clifford+T* [17] and use the number of T-gates as cost function. This simple cost model is based on the assumption that T-gates are far more expensive to realize than all other gates in the Clifford+T library [1]. Based on concrete mappings for Toffoli gates with small number of control lines [18] and a decomposition schemata [3] for larger Toffoli gates, an overapproximation for the number of T-gates necessary to realize an ESOP form $c_1 \oplus \cdots \oplus c_k$ can be computed as

$$T(c_1 \oplus \cdots \oplus c_k) = \sum_{i=1}^{k} T_{\text{cube}}(|c_i|) \tag{8.19}$$

with

$$T_{\text{cube}}(l) = \begin{cases} 0, & l \leq 1 \\ 7, & l = 2 \\ 16, & l = 3 \\ 8(l-1), & l > 3 \wedge n \geq \frac{3l+1}{2} \\ 16(l-1), & \text{else.} \end{cases} \tag{8.20}$$

8.5 Experimental Evaluation

We have implemented Algorithm 4 in *easy*, an open-source toolkit for manipulating ESOP forms[2] using the prominent state-of-the-art SAT-solver Glucose 4.1 [2] as decision procedure for Boolean satisfiability.
We have evaluated the SAT-based synthesis approach in four experiments[3]:

1. NPN4: We synthesized all ESOP forms of minimal size for the representatives of the NPN4 equivalence class.
2. LUT mapping: We synthesized one ESOP form of fixed-size and one of minimal size for each of the Boolean functions that occurred during LUT mapping of Boolean networks.
3. Random: We synthesized one ESOP form of fixed-size and one of minimal size for randomly generated Boolean functions.
4. Reversible logic synthesis: We generate Pseudo-Kronecker Expressions (PKRMs)—a special case of ESOPs—and ESOP using our exact method and analyze the effect of ESOP size minimization on the size of the corresponding quantum circuits. For evaluation, we use the T-metric presented in Sect. 8.4.

All experiments have been conducted on an Intel® Core™ i7-7567U CPU @ 3.50 GHz with 16 GB RAM.

Correctness All computed ESOP forms have been verified against their specifications, i.e., we simulated all ESOP forms for all possible values and compared the results of simulation with the initial truth tables of the provided Boolean functions. Note that it is not possible to verify the minimality of the ESOP forms.

NPN4 We synthesized all ESOP forms of minimum size for all 222 representatives of the NPN4 equivalence classes [12]. Computing one minimal ESOP form for each representatives takes 1.6 s, computing all minimal ESOP forms for each representatives takes 9.2 s. Figure 8.3 shows the histogram of the size of the minimal

[2]Easy, https://github.com/hriener/easy.

[3]The benchmarks and a detailed evaluation of the synthesis results can be found at https://hriener.github.io/misc/2018_easy.html.

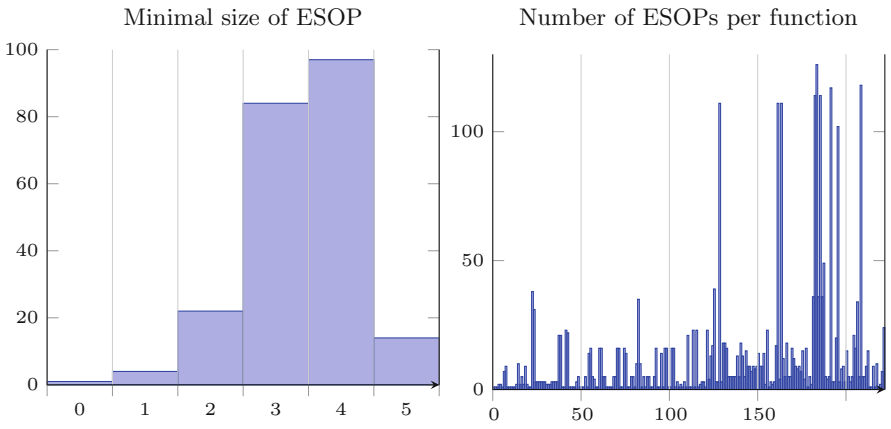

Fig. 8.3 Synthesis of minimal ESOP forms for NPN4

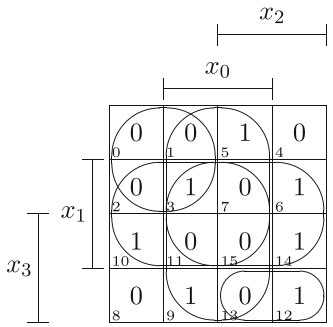

Fig. 8.4 Karnaugh map of `0x166A`

ESOP forms for the representatives (on the left) and the number of ESOP forms of minimal size per representative (on the right). On average a representative has 12 structurally different minimal ESOP forms. Some representatives can have 100 or more ESOP forms of minimal size. The Boolean function `0x166A` (shown in Fig. 8.4) has the most minimal ESOP forms (in total 126) within the NPN4 classes.

LUT Mapping We synthesized one ESOP form for a fixed number of ESOP terms and one ESOP form of minimal size using downward and upward search, respectively, for each Boolean function that occurred in LUT mapping of the EPFL benchmark suite. For LUT mapping, we used the ABC command `if -K 8` [4]. After LUT mapping, we applied `exactmine` [28] to extract all Boolean functions from the benchmarks. We obtained 4001 different Boolean functions with up to 8 Boolean variables and used SAT-based ESOP synthesis to compute ESOP forms. For this experiment, we consider a fixed conflict limit of 10,000. The synthesis results are presented in Table 8.1: the first column (**Terms**) is a user-specified upper limit on the number of terms. The rest of the table is organized in three

Table 8.1 Synthesis of ESOP forms for LUT mapping

	Fixed-size				Downward search				Upward search			
Terms	**R**	**C**	**k**	**T**	**R**	**C**	**k**	**T**	**R**	**C**	**k**	**T**
8	3735	266	5.19	49.65 s	3854	147	3.60	300.44 *s*	3857	3854	3.60	248.07 s
16	3806	195	7.10	50.56 s	3965	36	3.82	695.08 *s*	3965	36	3.82	338.72 s
32	3966	35	8.45	42.67s	4001	0	3.94	1430.41*s*	4001	0	3.94	355.49 s

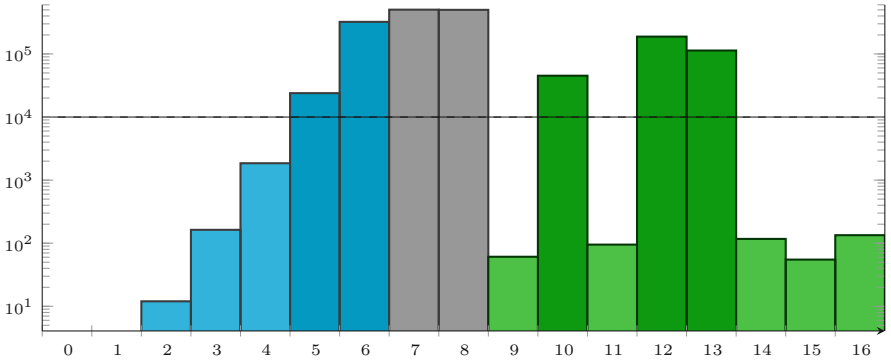

Fig. 8.5 SAT-solver results for different k for `0xF550311031100000`

parts. The first part (**fixed-size**) is dedicated to synthesis of an ESOP form for the given term limit (without minimizing the number of terms). In this case, the SAT-solver's heuristics decides whether unnecessary terms are canceled or kept. The second part (**downward search**) is dedicated to a synthesis procedure that iteratively synthesizes ESOP forms starting from the upper term limit and decreases the number of terms until the constraint system becomes unsatisfiable (as described in Algorithm 4). The satisfying assignment with the smallest number of terms is used for deriving an ESOP form. The last part (**upward search**) is similar to the second part, but starts with 1 term and increases the number if the constraint system is unsatisfiable. The satisfying assignment with the largest number of terms is used to derive an ESOP form. For each part, we list the number of Boolean functions successfully realized (**R**), the number of Boolean functions that could not be synthesized because the SAT-solver's conflict limit (**C**) was exceeded, the average number of terms (**k**) for all realizable Boolean functions, and the total runtime (**T**) for synthesizing all Boolean functions. The runtime includes the time for synthesizing the realizable Boolean function and the time spent in unsuccessful synthesis attempts.

Example 8.2 We illustrate the effect of the conflict limit on upward and downward search with a simple example. Consider the completely specified Boolean function `0xF550311031100000`. We attempt to synthesize an ESOP form of minimal size with at most 16 terms and a conflict limit of 10,000 using the upward and downward search procedures, respectively. Figure 8.5 shows the number of conflicts explored by the SAT-solver in logarithmic scale parameterized by the number of

terms (**k**). The colors encode the decision results: green denotes satisfiable, blue denotes unsatisfiable, and gray denotes unknown. For those k for which the conflict limit of 10,000 was reached, we repeated synthesis with a much higher conflict limit of 500,000 to understand what conflict limit would allow us to conclude the correct result. The results for $k = 7$ and $k = 8$, however, remain unknown, i.e., we do not know whether the constraints are satisfiable, because the conflict limit of 500,000 was also exceeded.

The downward search starts with 16 terms and systematically decreases the number of terms. During the search, the conflict limit is reached with $k = 13$ for the first; the search procedure interprets this as potentially satisfiable, such that the procedure proceeds until finally $k = 4$ is reached. For $k = 4$, the procedure concludes unsatisfiability, terminates, and returns the smallest constructed ESOP form with 9 terms determined during the search process.

The upward search procedure solves the constraint system with increasing number of terms starting with 1. For $k \leq 4$, the SAT-solver proves unsatisfiability of the constraint system. For $5 \leq k \leq 8$, the SAT-solver reaches the conflict limit, which is interpreted as potentially unsatisfiable by our search procedure, such that the search proceeds until $k = 9$. For $k = 9$ terms, the constraint system becomes for the first time satisfiable and the corresponding ESOP form with 9 terms is returned.

Random We synthesized ESOP forms for randomly generated, incompletely specified Boolean functions over 5, 6, 7, and 8 Boolean variables. Each bit in the Boolean function and its care function was chosen by flipping a fair coin. In total, we generated 100 Boolean functions for each number of Boolean variables. Table 8.2 summarizes the results for synthesizing ESOP forms. The first two columns list the number of Boolean variables (**Var.**) and a fixed bound on the number of terms (**Terms**). The rest of the table is organized as Table 8.1. Due to the symmetric design of downward and upward search, they reached exactly the same minimal ESOP forms. Overall downward search is slower due to the fact that unsatisfiability is typically harder to prove and can only be concluded by the SAT-solver for sufficiently small k. Consequently, the downward search procedure on average analyzes many more cases before unsatisfiability is reached. In contrast, upward search keeps searching until satisfiability is reached for the first time, which can occur early in the search process.

Table 8.2 Synthesis of ESOP forms for randomly generated Boolean functions

		Fixed-size				Downward search				Upward search			
Var.	**Terms**	**R**	**C**	**k**	**T**	**R**	**C**	**k**	**T**	**R**	**C**	**k**	**T**
5	16	100	0	8.58	0.11 *s*	100	0	3.34	1.35 *s*	100	0	3.34	0.12 *s*
6	16	99	1	11.32	0.42 *s*	100	0	5.62	24.70 *s*	100	0	5.62	15.99 *s*
7	32	86	14	24.91	3.71 *s*	100	0	17.96	276.70 *s*	100	0	17.96	210.02 *s*
8	96	79	21	54.35	19.64 *s*	100	0	45.41	2156.96 *s*	100	0	45.41	1151.75 *s*

Table 8.3 Synthesis of ESOP forms for Boolean functions from DBS

Benchmark		PKRM			Exact		
		k	**Time [s]**	***T*-gates**	**k**	**Time [s]**	***T*-gates**
1	`0x50455400`	18	0.00	355	4	0.01	128
2	`0x0880`	2	0.00	64	2	0.02	64
3	`0x00f07800`	4	0.00	78	3	0.11	80
4	`0x00070000`	8	0.00	174	2	0.01	80
5	`0x0007f000`	8	0.00	165	3	0.01	96
7	`0x0000ff80`	6	0.00	151	2	0.01	39
8	`0x06170360`	12	0.00	188	5	0.03	135
9	`0x4770ce38`	18	0.00	298	6	0.28	151
9	`0x6a0a4b6e`	18	0.00	339	6	0.25	167
10	`0x4727724a`	18	0.00	335	7	0.33	167
Σ		112	0.00	2147	40	1.06	1107

Reversible Logic Synthesis We synthesized ESOP forms for Boolean functions obtained from *decomposition-based synthesis* (DBS), a recent approach to map permutations into quantum circuits [5, 29]. We extracted the Boolean functions from the DBS approach and synthesized for each Boolean function, a *Pseudo-Kronecker Expressions* (PKRM)—a special case of ESOP forms—using the approach proposed by Drechsler [6], and an exact ESOP form using our proposed method with downward search.

Table 8.3 shows experimental results for 10 Boolean functions; each of them corresponds to one Toffoli gate in the quantum circuit. For both synthesis techniques, the table lists the number of product terms (k), the required runtime (**Time**), and the number of T-gates computed using Eq. (8.19). The example illustrates the positive effect of ESOP optimization for reducing the cost of realizing a quantum circuits. By using our exact ESOP synthesis method, the over-approximated number of T-gates could be reduced by 48.44%, while the additional runtime can be almost neglected.

8.6 Conclusion

We have presented an exact synthesis approach for computing ESOP forms using Boolean satisfiability. The approach needs no pre-computed information, synthesizes one or multiple ESOP forms of minimal size, and can take completely specified or incompletely specified Boolean functions as specifications. We have implemented the approach using an off-the-shelf SAT-solver and have further presented a relaxation that leverages the SAT-solver's conflict limit to find ESOP forms with almost minimal size. We have also presented evidence that the synthesis procedure can deal with small-scale ESOP forms with up to 8 Boolean variables and up to 100 terms. As benchmarks, we have used Boolean functions in the

NPN4 equivalence class, Boolean functions that appeared during 8-LUT mapping, and randomly generated Boolean functions. Moreover, we show how the proposed techniques can be used to reduce the costs for implemented quantum circuits. We envision that the proposed SAT-based synthesis technique can be integrated with large-scale ESOP optimization procedures, e.g., by selecting windows of terms and resynthesizing them.

Acknowledgements This research was supported by H2020-ERC-2014-ADG 669354 CyberCare (200021-146600) 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 Trans. CAD Integr. Circuits Syst. **32**(6), 818–830 (2013)
2. Audemard, G., Simon, L.: On the glucose SAT solver. Int. J. Artif. Intell. Tools **27**(1), 1–25 (2018)
3. Barenco, A., Bennett, C.H., Cleve, R., Divincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Phys. Rev. A: At. Mol. Opt. Phys. **52**(5), 3457–3467 (1995)
4. Brayton, R.K., Mishchenko, A.: ABC: an academic industrial-strength verification tool. In: Proceedings of Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, July 15–19, 2010, pp. 24–40
5. De Vos, A., Van Rentergem, Y.: Young subgroups for reversible computers. Adv. Math. Commun. **2**(2), 183–200 (2008)
6. Drechsler, R.: Preudo-Kronecker expressions for symmetric functions. IEEE Trans. Comput. **48**(9), 987–990 (1999)
7. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5–8, 2003 Selected Revised Papers, pp. 502–518
8. Fazel, K., Thornton, M.A., Rice, J.E.: ESOP-based Toffoli gate cascade generation. In: Pacific Rim Conference on Communications, Computers and Signal Processing (2007)
9. Feynman, R.P.: Quantum mechanical computers. Opt. News **11**, 11–20 (1985)
10. Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theor. Phys. **21**(3–4), 219–253 (1982)
11. Gaidukov, A.: Algorithm to derive minimum ESOP for 6-variable function. In: International Workshop on Boolean Problems, pp. 141–148 (2002)
12. Goto, E., Takahasi, H.: Some theorems useful in threshold logic for enumerating Boolean functions. In: IFIP Congress, pp. 747–752 (1962)
13. Kalay, U., Hall, D.V., Perkowski, M.A.: A minimal universal test set for self-test of EXOR-sum-of-products circuits. IEEE Trans. Comput. **49**(3), 267–276 (2000)
14. Kamath, A.P., Karmarkar, N., Ramakrishnan, K.G., and Resende, M.G.C.: A continuous approach to inductive inference. Math. Program. **57**, 215–238 (1992)
15. Knuth, D.E.: The Art of Computer Programming, vol. 4. Fascicle 6: Satisfiability, 1st edn. Addison-Wesley Professional, Boston (2015)
16. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Proceedings Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, July 7–11, 2008, pp. 486–498

17. Linke, N.M., Maslov, D., Rötteler, M., Debnath, S., Figgatt, C., Landsman, K.A., Wright, K., Monroe, C.R.: Experimental comparison of two quantum computing architectures. Quant. Phys. Comput. Sci. Emer. Technol. (2017). abs/1702.01852
18. Maslov, D.: Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization. Phys. Rev. A **93**(2), 022311 (2016)
19. Mishchenko, A., Perkowski, M.A.: Fast heuristic minimization of exclusive-sums-of-products. In: Reed-Muller Workshop (2001)
20. Mishchenko, A., Perkowski, M.A.: Logic synthesis of reversible wave cascades. In: International Workshop on Logic Synthesis, pp. 197–202 (2002)
21. Mizuki, T., Otagiri, T., Sone, H.: An application of ESOP expressions to secure computations. J. Circuits Syst. Comput. **16**(2), 191–198 (2007)
22. Papakonstantinou, G.K.: A parallel algorithm for minimizing ESOP expressions. J. Circuits Syst. Comput. **23**(1), 1450015 (2014)
23. Papakonstantinou, K.G., Papakonstantinou, G.: A nonlinear integer programming approach for the minimization of Boolean expressions. J. Circuits Syst. Comput. **29**(10), 1850163 (2018)
24. Riener, H., Ehlers, R., Fey, G.: CEGAR-based EF synthesis of Boolean functions with an application to circuit rectification. In: 22nd Asia and South Pacific Design Automation Conference, ASP-DAC 2017, Chiba, January 16–19, 2017, pp. 251–256
25. Sampson, M., Kalathas, M., Voudouris, D., Papakonstantinou, G.K.: Exact ESOP expressions for incompletely specified functions. Integration **45**(2), 197–204 (2012)
26. Sasao, T., Fujita, M. (eds.): Representations of Logic Functions Using EXOR Operators, pp. 29–54. Springer, New York (1996)
27. Soeken, M., Roetteler, M., Wiebe, N., De Micheli, G.: Design automation and design space exploration for quantum computers. In: Design, Automation and Test in Europe, pp. 470–475 (2017)
28. Soeken, M., Riener, H., Haaswijk, W., De Micheli, G.: The EPFL logic synthesis libraries (2018). arXiv e-prints 1805.05121
29. Soeken, M., Mozafari, F., Schmitt, B., De Micheli, G.: Compiling permutations for superconducting QPUs. In: Design Automation Conference (2019)
30. Stergiou, S., Papakonstantinou, G.K.: Exact minimization of ESOP expressions with less than eight product terms. J. Circuits Syst. Comput. **13**(1), 1–15 (2004)