IET Computers & Digital Techniques

Research Article

IET Journals
The Institution of Engineering and Technology

Improved circuit synthesis approach for exclusive-sum-of-product-based reversible circuits

ISSN 1751-8601
Received on 24th February 2017
Revised 17th December 2017
Accepted on 21st January 2018
E-First on 11th April 2018
doi: 10.1049/iet-cdt.2017.0016
www.ietdl.org

Chandan Bandyopadhyay[1] ✉, Shalini Parekh[2], Hafizur Rahaman[1]

[1]Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur 711103, India
[2]Department of Computer Science Engineering, KU Leuven (Katholieke Universiteit Leuven), Belgium Naamsestraat 63 – box5410, BE-3000 Leuven, Belgium
✉ *E-mail: chandanb@it.iiests.ac.in*

Abstract: In this computing paradigms, the quantum computing has evolved as a promising platform for designing very fast computable circuits. In view of the improved design of such circuits, an efficient synthesis approach for circuits needs to be developed. As a direct synthesis of the quantum circuit is a bit complicated, the concept of reversible circuit appears which internally implements the quantum functionality, and to design better quantum circuit the corresponding reversible circuit have to be optimised. Considering this need, in this work, the authors develop an efficient reversible circuit synthesis scheme that constructs improved circuits by minimising the quantum cost. The entire work is completed in two phases. In the first phase, a circuit design scheme based on the best neighbour is implemented, where a function shares a portion of its own data with a chosen neighbour termed as the best neighbour and builds the shared structure. In the second phase, the designed circuit passes through an optimisation process which further reduces the cost metrics of the circuit. The experiment shows that the optimisation process substantially reduces the cost of the circuits to a great extent. At the end of the work, a comparative study with related works has also been presented.

1 Introduction

In recent years, quantum computing [1, 2] has evolved as one of the prominent computing paradigms with applications in diversified fields like secure communication [3], cryptography [4, 5], power efficient designs [6] and very high speed machine designs [7] and so on. Like Boolean circuits, quantum circuits can also be designed using a set of special gates – called quantum gates [8]. However, such circuits have some major differences with normal traditional circuits in terms of computing speed, fabrication technology and design methodology. Unlike Boolean circuits that rely on two states 0 and 1, quantum circuits can have numerous states known as *superposition* state [9], which basically helps a quantum circuit in making very fast computations.

The works on quantum circuits and related methodologies have advanced significantly in a couple of years. The physical implementation of such circuits has started and quantum circuits fabrication technologies like nuclear magnetic resonance (NMR) [7], Superconducting qubit [10], Ion Trap [11] already have shown their capability in the successful fabrication of quantum integrated circuits (ICs). Even designing quantum computers [12] with lower qubits has been initiated. However, in the meantime, several challenges appear and subsequently a set of constraints have been imposed on designer like the circuit should be nearest neighbour compliant one [13, 14], *Bennett embedding* [15] (that enforces a circuit not to expose intermediate results to output lines) should exist in the designed architecture, the circuit should be optimised one and so on. Apart from these challenges, researchers have continuously studied on various architectural aspects as well to simplify the design procedure.

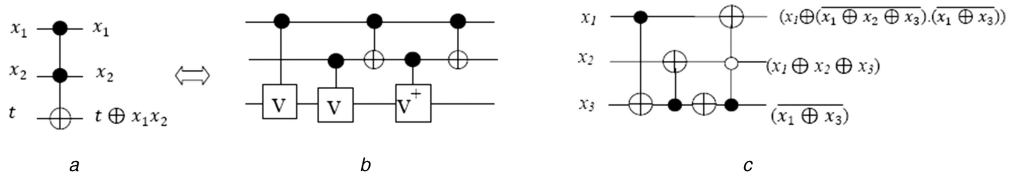
Since the design of quantum circuits is a complex task, many methods employ an approach that does not directly realise the given quantum functionality but employs a multiple-step approach. As all the quantum operations are inherently reversible, the quantum functionality of Boolean components is first realised as a reversible circuit, rather than a quantum circuit. Then, the resulting reversible circuit is mapped to an equivalent quantum representation. This significantly reduces synthesis complexity.

Furthermore, in a way to generalise the design procedure, a set of reversible gates like Toffoli [16], Feynman [17], Fredkin [18] are introduced. Though this development makes the design procedure very generic, at the same time constructing cost efficient circuits found much to interest and developing efficient synthesis schemes seem very important.

Though a huge variety of synthesis approaches are already available, the more efficient synthesis technique needs to be developed. The reversible circuit synthesis approaches can be categorised into two classes depending on the level of scalability and the type of algorithm employed:

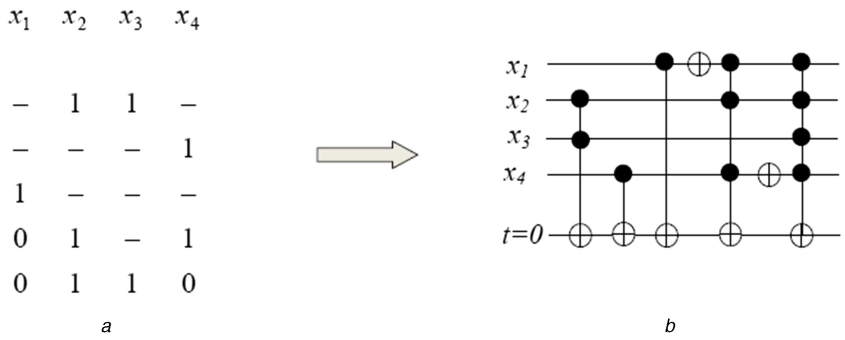
i. *Optimal methods drove synthesis schemes:* Synthesis schemes based on optimal algorithms which generate minimal circuits [18, 19].However, such algorithms have a low scalability level and cannot generate optimal results beyond the six-variable functions. This limitation is due to the rapid increase in time and space complexities [20] in algorithms when they are applied beyond a certain limit.
ii. *Sub-optimal methods driven synthesis schemes:* Synthesis approaches belonging to this category produce sub-optimal result sets. Such schemes can be scaled to very large size functions but cannot ensure optimality in solution space. Methods like genetic algorithms, A* [21], ant colony optimisation [22], simulated annealing [23], binary decision diagrams (BDDs) [24] and exclusive-sum-of-products (ESOPs) [25] based synthesis approaches are the examples of this class.

Among these sub-optimal solutions, BDD and ESOP are very widely known techniques in synthesis platform which are known for their capability in processing very large size benchmarks (beyond 100 variables). Both the techniques have some merits and demerits. For example, BDD-based circuits are much more cost optimised than ESOP-based designs, but such designs require huge ancille lines to form circuits. On the contrary, though ESOP-based designs are not much cost effective as BDD based ones, such designs uses a minimum number of ancille lines while forming circuits. Comparatively, ESOP-based designs are considered more efficient than BDDs due to two reasons: (i) they do not require



a b c

Fig. 1 *Preliminaries on reversible family*
(a) Two-control Toffoli gate, ***(b)*** NCV realisation of (a), ***(c)*** Example of reversible circuit



a b

Fig. 2 *Deriving ESOP representation from cubelist*
(a) Cube list for function $f(x_1, x_2, x_3, x_4) = x_2x_3 \oplus x_4 \oplus x_1 \oplus \bar{x}_1x_2x_4 \oplus \bar{x}_1x_2x_3\bar{x}_4$, ***(b)*** Equivalent ESOP representation

additional circuit lines to implement functions, (ii) ESOP-based synthesis generates fast sub-optimal circuits with a linear time complexity which BDDs cannot.

In recent years, several synthesis algorithms based on ESOP representation have been introduced [26–30] and improvements over existing designs are being reported very often. Finding ESOP-based designs very promising, here in our work, we have come up with a new algorithm that exploits the best neighbour property to form improved ESOP realisations. In our design process, after finding the best neighbour, a function shares a portion of its own data and after that, it executes an optimisation process to further reduce the cost values of the circuit. This scheme shows considerable improvements in cost parameters compared to related designs.

The remaining of the paper is organised as follows. Section 2 presents the preliminaries. The proposed design technique is stated in Section 3. Section 4 presents experimental results and discussions. Finally, Section 5 concludes the work.

2 Preliminaries

This section presents a brief introduction to reversible quantum logic, ESOP representation, different cost metrics for evaluating circuit efficiency and a brief discussion on various works related to ESOP-based designs.

2.1 Basics on reversible and ESOP representation

Definition 1: A circuit can be called a 'Reversible Circuit' if it satisfies the three basic criteria – (i) there exist one to one mapping between input to output vectors and vice versa, (ii) it should be fan-out free (iii) and finally the circuit should consist of reversible gates only.
Gates like NOT, CNOT and Toffoli are the well-known gates that are mostly used in designing reversible circuits. Each reversible gate has a set of control and target lines that implement a certain function.

Example 1: Design of a reversible gate (two-control Toffoli) is shown in Fig. 1*a*. As we said earlier that behind each reversible gate there exists a set of quantum gates, which collectively implements a logic, here also, the internal design of the said gate is consisting of a cluster of quantum gates (quantum realisation is depicted in Fig. 1*b*) that collectively implement the function $t \oplus x_1x_2$. After cascading several similar reversible gates a reversible circuit can be formed and the design of such a reversible circuit is shown in Fig. 1*c*.

Definition 2: An ESOP is a Boolean expression in which product terms are separated by XOR operator. Basically, it is a variation of the sum of products (SOPs) form in the Boolean function representation.

For example, consider the Boolean expression $f = xy + yz$ (which is in SOP form), the equivalent ESOP expression would be $f = xy \bigoplus yz \bigoplus xyz$. While representing an n-bit single output function in ESOP format, it requires total $(n + 1)$ circuit lines, where the first n lines are control lines and the last $(n + 1)$th line is the target line or the functional output line.

ESOP circuit is mainly designed from cube list which stores the gate level descriptions of that circuit. The term cube list can be defined as follows:

Definition 3: A cubelist is expressed in the 2D matrix format ($m \times n$), whose each row (denoted by variable m) represents a cube (C_i) that contains specified control information of a gate and variable n denotes the number of control lines present in that circuit.

Example 2: The ESOP representation of the function $f(x_1, x_2, x_3, x_4) = x_2x_3 \oplus x_4 \oplus x_1 \oplus \bar{x}_1x_2x_4 \oplus \bar{x}_1x_2x_3\bar{x}_4$ is given in Fig. 2*b*.However, prior to designing the circuit, its corresponding cubelist is retrieved (given in Fig. 2*a*) using some cubelist generation algorithm. From the equivalent cubelist of the input function, it can be seen that there are a total five cubes in the gate level descriptions, where some control nodes have positive polarity some of negative. While forming the circuit the negative controls have been replaced with NOT gates and corresponding to each cube a gate is designed in resultant realisation finally leading to a complete ESOP circuit of Fig. 2*b*.

Definition 4: For a multiple output function, each output may shares some common functions in-between the different functional outputs and such functional outputs are called its neighbours. Among these neighbours with which the selected function shares the maximum amount of sub-functions is termed as the best neighbour of that selected function.

2.2 Associated cost metrics for reversible circuit

In a way to evaluate the efficacy of designed circuits, two cost parameters – quantum cost [26] and gate count are known to be widely accepted metrics. Apart from these two parameters, in recent years several cost parameters like T-count, T-depth [27, 28], nearest neighbour cost [13, 14] have been introduced. As in our designs, we have mainly computed the first two cost values against

the benchmark functions, here we are only introducing the said terms.

Quantum cost (QC): The minimum number of quantum operations required to implement a quantum circuit is known as quantum cost.

Gate count: The number of gates present in a circuit is known as gate count.

The details regarding cost values are reported in [26]. As per the above definition, the quantum cost and gate count values for the circuit given in Example 2 (Fig. 2*b*) are 51 and 7, respectively.

2.3 Brief review of ESOP related works

As in our designs, we have worked with ESOP circuits, here we are presenting some reviews on that synthesis scheme.

The ESOP-based circuit synthesis approach to represent Boolean function was first introduced in [29]. Initially, it has been shown that to synthesis an n input m output function, the technique requires a total $(2n + m)$ lines but finally, they optimised the initial representation and formulated an improved scheme where the same input function is represented in ESOP form using $(n + m)$ lines only.

Later on, the scientists have worked with various design models on ESOP circuits to further improve the cost parameters in circuits.

An improved ESOP-based circuit design using co-relations or structural symmetries between reversible gates has been proposed in [29]. In this work, an autocorrelation coefficient-based cost function has been introduced in identifying the positions of Toffoli gates.

Application of templates in optimising ESOP circuits has been introduced in [30], where a greedy technique is implied in the template matching scheme to reorder the cubes and to minimise the usage of NOT gates in the circuit.

Several ESOP-based techniques [like the incorporation of negative control gates, reordering of cubes] have been proposed to optimise the cost metrics in later times. However, as all these techniques were not reducing the cost to a great extent and a new technique generating more improved ESOP circuit has been developed in [31] by sharing the control lines in between them. However, this approach does not work effectively when functions do not bear common structural similarities in between them.

Further, this sharing concept has been improved in [32], where cube clustering and functional output sharing both are applied in a way to get cost efficient representations.

A better way for ESOP-based designs [33] by finding the best pairs and then mapped them to a weighted matching problem is reported in a very recent year, where considerable improvements in quantum cost over ESOP benchmarks are achieved.

3 Proposed technique

Our developed synthesis scheme has two objectives, first, to design cost efficient representations and second, to make the synthesis scheme applicable for large-scale benchmarks too. The proposed design procedure is completed in two phases. In the first phase, a synthesis algorithm generates shared functionality-based circuits from primary inputs. However, these shared circuits are not optimised and to further improve it, the circuits pass through a second phase where a set of optimised templates replace sub-structures in the circuits by their structures to eliminate the possible redundancies from the designs.

3.1 Phase 1: Formation of shared-functionality-based ESOP designs

The main objective of this phase is to generate shared-functionality-based designs from initial structures. However, how to share the functionalities in between output lines to get efficient representations depend on the technique or heuristic deployed. Here, we have come up with a strategy that uses some heuristics to determine to whom a function should share its functionality in a way to form shared designs. This strategy involves four steps which we are explaining here in details.

Input: *.pla* file of Benchmark function
Output: Shared ESOP circuit

begin
  ESOP_cubelist = EXORCISM-4(*.pla*);
  do
    count = Calculate_Number_of_Common_Cubes(f_i, f_k) ;
    $index_{ik}$ = Info(f_i, f_k) ;
    TTList = Store($index_{ik}$, *count*) ;
  while(all the combinations on outputs f_1 ... f_m are covered)
  Sorted_ TTList = Sort_in Desending_Order(*TTList*) ;
  Draw_Circuit_for(*Sorted_ TTList*$_0$) ;
  i = 0;
  do
    if (Select_Pair(*Sorted_ TTList*$_i$)∩ Select_Pair(*Sorted_ TTList*$_k$) = ϕ) **then**
      Draw_Circuit_for(*Sorted_ TTList*$_k$) ;
    end
  while(all outputs f_1 ... f_m are checked)
end

Fig. 3 *Algorithm 1: Generation of shared ESOP structure from specification files*

Step 1: First, process a source file (in *.pla* format) through tool EXORCISM-4 [34] and obtain a (*.esop*) file in which specification of an ESOP circuit is written in cubelist format. Now, if each of the cubes from this cubelist is mapped to specific gates over a set of the control and target lines then an ESOP circuit can be obtained. Here, we termed this circuit as the primary ESOP circuit.

However, this circuit is not an optimised one as there are several areas where the circuit can be improved further. For example, by sharing common sub-expressions between output lines, separating the circuit in sub-circuits and optimising each module, sharing data by adding some ancille lines. Here in a way to build cost efficient designs, we perform the first type of optimisation (as mentioned earlier) by sharing common sub-expressions between output lines in phase 1 and the second type of optimisation in phase 2. Now we state how this sharing of common sub-expressions is possible using our developed strategy which is explained in next three steps.

Step 2: Before sharing output lines based on common sub-expressions, we need the information metrics like what sub-expressions are common and which combinations can be selected so that possible maximum sub-expressions can be shared. In a way to find that ground, here we have come up with a heuristic-based approach.

First, we read the (*.esop*) file that has been obtained in the previous step and retrieves the cubelist from it. Next, we generate all possible $\binom{n}{2}$ distinct combinations (where n denotes the number of outputs in that function) of pairs in output lines that reveal what and how much information is shared between any pair of output lines in that specific circuit.

Step 3: Now, we prepare a list and count the maximum number of cubes that are common in the paired outputs and sort it in descending order. While scanning the list in a top-down manner, including those output lines in pairs that share a maximum number of cubes in between them to find the best matches. However, the choice of pairs should be such that no pairs should contain the same outputs that are shared by any of the previous pairs, i.e. all the pairs should be mutually exclusive in terms of sharing of functional outputs.

Step 4: This selection process produces a total of $n/2$ pairs from n output lines. Now, the selected pairs can share its common sub-expressions with its chosen partner. The entire process that leads to the formation of shared-ESOP circuits is stated in Algorithm 1 (see Fig. 3).

Here, we explain the stated procedure with examples. Here, the procedure is described using two different benchmark functions where the first benchmark has even number of outputs and the second function has an odd number of functional outputs. The reason behind taking such functions is that we want to show how different combinations of pairs are selected to share common sub-functions against different benchmarks with unlike *inputs/outputs*.

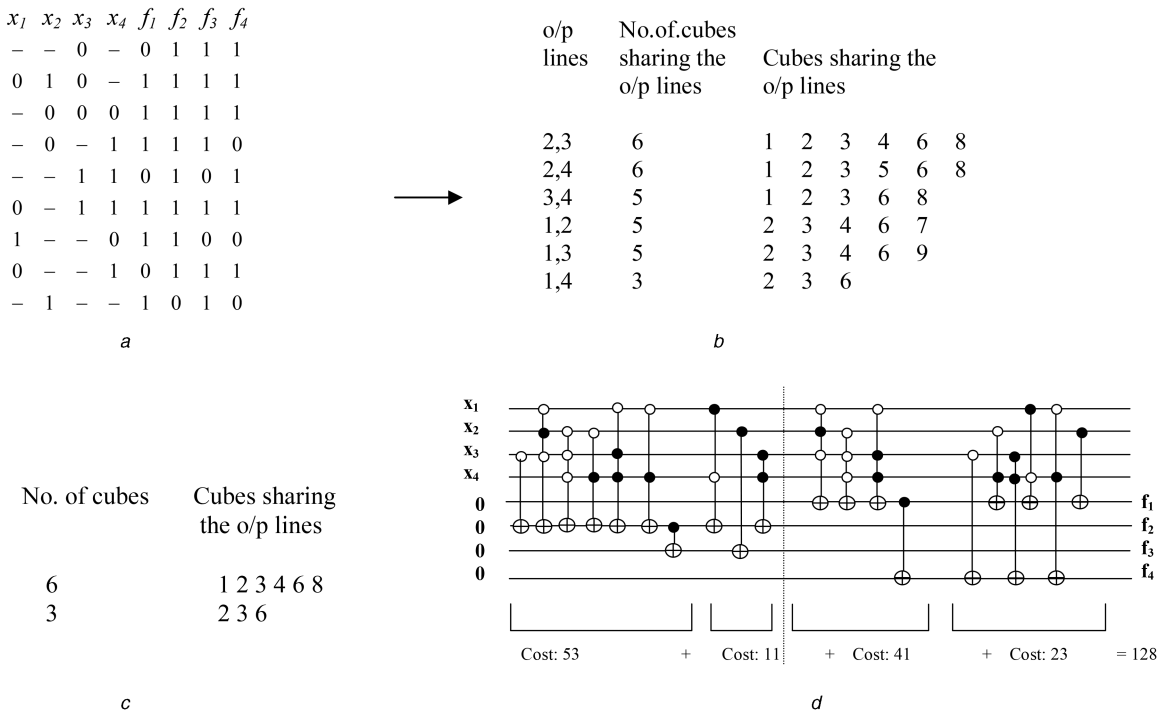


Fig. 4 *Expanding Toffoli network based on hamming value*

(a) Input Cubelist for function *aj-e11_81*, ***(b)*** List of possible cube pairs and their sharing information, ***(c)*** Final sharing information, ***(d)*** Improved ESOP circuit for function *aj-e11_81*

Example 3: Consider a benchmark function 'aj-e11_81', which has four input and four output lines. First, we process this file through the EXORCISM-4 tool and obtain its equivalent cubelist in Fig. 4*a*. Now, we execute Step 2 of the stated scheme by taking the generated cubelist as input. After completion of that execution, a sorted list (as shown in Fig. 4*b*) containing a combination of pairs of output functions is computed. As it can be seen that the list contains a total of $\binom{4}{2}$ distinct pairs {(2, 3), (2, 4), (3, 4), (1, 2), (1, 3), (1, 4)}.

Now, we have started scanning the information metric of Fig. 4*b* in a top-down manner. We found the first output pair (2, 3) that bears the sharing information for second and third functional outputs and it is six cubes (cube1, cube2, cube3, cube4, cube6 and cube8) in between them. So, we choose the initially selected pair (2, 3) and place its associated cubes in the final sharing table as a first entry.

The next pair is (2, 4) and this pair cannot be moved to final sharing table as the second functional output is also present in the pair (2, 3) which is previously selected. If we would select it, then the mutual exclusion property in pair set would be violated. Similarly, we cannot select the next three output pairs. However, the sharing of cubes in between (1, 4) is possible as both the selected pairs are mutually exclusive, i.e. $(2, 3) \cap (1, 4) = \phi$. So, we move the pair (1, 4) to final sharing table.

As all the output functions are enlisted, we stop this process and the final sharing information is formed in Fig. 4*c*. Next, we design the circuit in Fig. 4*d* as per the sharing information obtained in final sharing table of Fig. 4*c*. The circuit of Fig. 4*d* has a total cost of 128, but in spite of generating the final sharing table, if we proceed with the cubelist of Fig. 4*a*, it would cost 218. So, it can be seen that application of the developed scheme helps to build improved structures.

In the next example, we are considering a function containing odd numbers of outputs as in the previous example, it was easy to share an even number of outputs in-between by forming pairs.

Example 4: Here, we have taken another benchmark function 'pcler8_190' which has 16 input and 5 output lines. The equivalent ESOP cubelist of this function is shown in Fig. 5*a*. At first, the sharing metrics of cubes are computed using the same strategy and are given in Figs. 5*b* and *c*, respectively. Now, it can be seen that only two pairs (3, 4) and (1, 2) share functional data with its partner. However, as the ninth cube in the cubelist has no defined bits so this cube cannot be shared with any of the outputs.

The final shared circuit is given in Fig. 5*d*. Here also if a simple ESOP circuit is drawn directly from the cube list source file, then it would cost 368 but in our design (Fig. 5*d*), the cost has improved to 320.

3.2 Phase 2: Circuit optimisation towards obtaining improved ESOP representation

In this phase, we process the circuits obtained in the previous phase through a set of optimised templates and a transformation rule in which sub-structures in the circuits are replaced with suitable templates to get more efficient realisations. This phase involves two steps: in the first step, an optimisation process runs and in the second step, the control lines are shared for making the designs more compact.

Step 1: The designed templates are shown in Fig. 6 and these templates find its structure, which matches the input circuit and replaces with suitable templates to make the circuit an improved one. Along with these set of templates, we introduce one more transformation rule that also helps to expand circuits for getting optimised in the future process.

Circuit transformation rule: For a gate netlist N, if there exist two successive gates G_1 and G_2 over same target line T in such a way that both the gates maintain a hamming distance of k, then they can be replaced with k number of gates whose cumulative cost is lower than the cumulative cost of gates G_1 and G_2.

For example, the gates of Fig. 7*a* have a hamming distance of 3, and they can be replaced with the equivalent structure of Fig. 7*b*. Hence the cost is reduced to 17 from the original cost of 26. Similarly, the gates of Fig. 7*c* maintain a hamming distance 4 in between themselves, which can be replaced with lower cost-based Toffoli cluster as given in Fig. 7*d*.

Step 2: In phase 1, the circuit shares its functional data between target lines and then the circuit passes through an optimisation process as mentioned in *step 1* of phase 2. Now, there may exist certain parts of the circuit which can be optimised further if sharing of data in control lines is possible.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	f_1	f_2	f_3	f_4	f_5
–	–	1	1	1	1	–	–	–	–	–	–	–	–	–	–	1	0	0	0	0
–	–	1	1	1	1	–	–	–	0	–	0	–	–	–	–	0	0	1	1	0
–	–	1	1	1	1	–	–	–	0	–	–	–	–	–	–	0	1	0	0	0
–	–	1	1	1	1	–	–	–	0	–	0	1	–	–	–	0	0	0	1	0
–	–	–	–	1	1	–	–	–	0	–	0	1	–	–	–	1	0	0	0	0
–	–	–	1	–	–	–	–	1	–	1	–	–	–	1	1	0	0	0	0	1
–	–	–	–	1	1	–	–	–	–	–	–	–	0	–	–	0	0	0	1	0
–	–	–	–	0	1	–	1	–	–	–	–	–	–	–	–	0	0	0	1	0
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	1	1	1	1	0
–	–	–	–	1	1	–	–	–	–	–	–	0	–	–	–	0	0	1	0	0
–	–	–	–	0	1	1	–	–	–	–	–	–	–	–	–	0	0	1	0	0
–	–	–	–	1	1	–	–	–	–	–	0	–	–	–	–	0	1	0	0	0
–	1	–	–	0	1	–	–	–	–	–	–	–	–	–	–	0	1	0	0	0
1	–	–	–	0	1	–	–	–	–	–	–	–	–	–	–	1	0	0	0	0

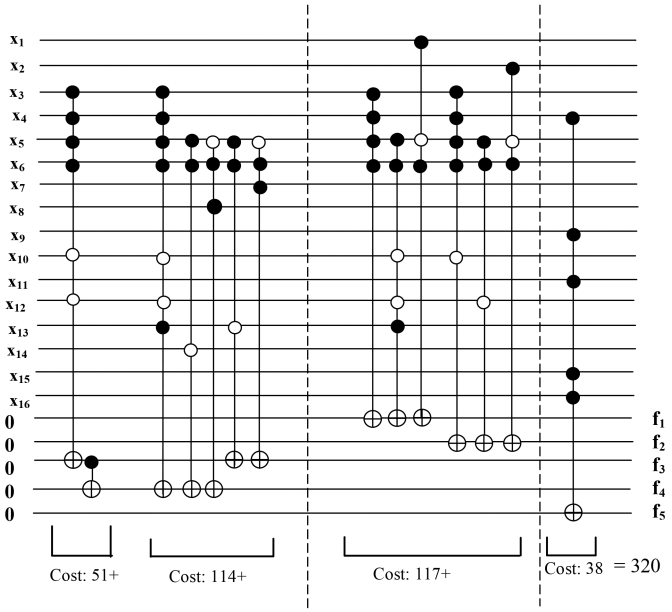
a

o/p lines	No. of cubes sharing the o/p lines	Cubes sharing the o/p lines
3,4	2	2 9
1,2	1	9
1,3	1	9
1,4	1	9
2,3	1	9
2,4	1	9
1,5	0	
2,5	0	
3,5	0	
4,5	0	

b

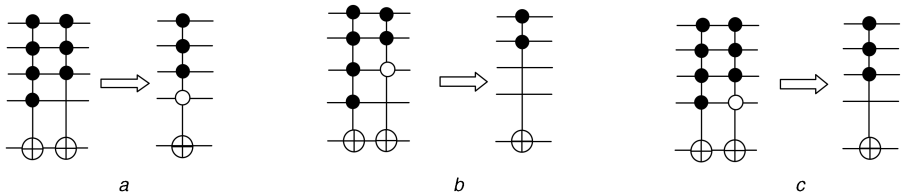
o/p lines	No.of cubes	Cubes sharing the o/p lines
3,4	2	2 9
1,2	1	9

c



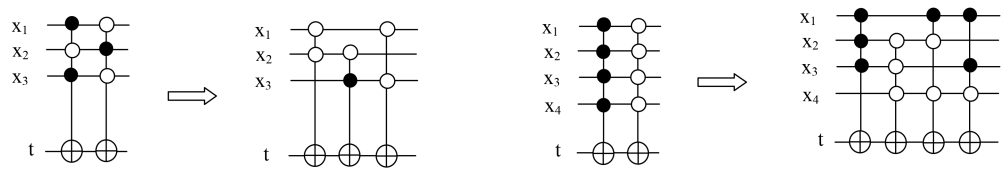
d

Fig. 5 *Employing algorithm1 over benchmark function pcler8_190*
(a) Input cubelist for function *pcler8_190*, ***(b)*** List of possible cube pairs and their sharing information, ***(c)*** Final sharing information, ***(d)*** Improved ESOP circuit for function *pcler8_190*



a *b* *c*

Fig. 6 *The derived templates*
(a) Template 1, ***(b)*** Template 2, ***(c)*** Template 3



a *b* *c* *d*

Fig. 7 *Expanding Toffoli network based on hamming value*
(a) Two gates with hamming distance of 3, ***(b)*** Transformed circuit from (a), ***(c)*** Two gates with 2 hamming distance apart, ***(d)*** Transformed circuit from (c)

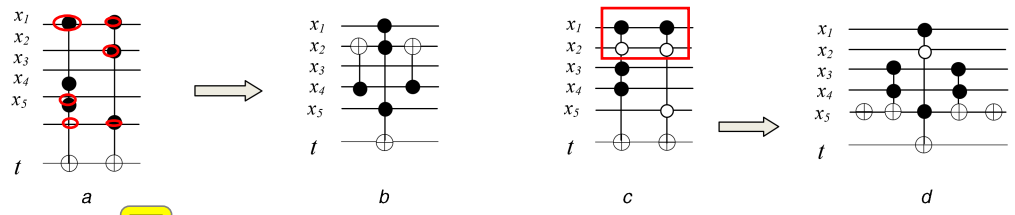


Fig. 8 *Control node sharing templ*
(a) Initial design incurring total QC = 26, ***(b)*** QC comes down to 15 after sharing the control lines, ***(c)*** ESOP circuit incurring QC = 42 before sharing, ***(d)*** Cost reduced to 25 after sharing the control lines

Input: Shared ESOP circuit: CKT_{shared}
Output: Improved ESOP representation: $CKT_{improved}$

```
begin
    T1= template 1;
    T2= template 2;
    T3= template 3;
    Flag=1;
    while(Flag==1)
        do
            Match_Template(Ti, CKT_shared);
            CKT_updated = Replace_with_Template(Ti, CKT_shared);
        while(no further matches are found in CKT_updated)
        S=Invoke(circuit_transformation_rule);
        if (S==False)
            Flag=0;
        end if
    end while
    CKT_improved = Share_cube_Process (CKT_updated)
end
```

Fig. 9 *Algorithm2: Template matching process*

In a way to find a solution, here we have come up with a different design process that helps to share functional values in-between control lines. Two separate designs for two different conditions (two Toffoli gates with equal no. of defined control nodes and two gates with unequal nodes) have been shown in Figs. 8*a* and *c*. Interestingly, it can be seen that after sharing the control node values, the cost of the design of Fig. 8*a* is reduced to 15 from 26. Similarly, the design cost of Fig. 8*c* has come down to 25 from earlier value of 42.

In the entire optimisation process, we have used several ways to reduce the cost of the designs by improving their data sharing approach. Now, the whole optimisation algorithm has been summarised in Algorithm 2 (see Fig. 9).

In a way to explain how the optimisation process works, we are taking the post-synthesised circuits of Figs. 4*d* and 5*d* which we have obtained after executing the phase 1 over two different benchmark functions '*aj-e11_81*' and '*pcler8_190*'.

Once the circuit of Fig. 4*d* passes through the first optimisation process of phase 2, the cost of the circuit is reduced to 117 (see Fig. 10*a*) and later when the second optimisation process executes over Fig. 10*a*, the cost further comes down 104 (see Fig. 10*b*).

Similarly, when the circuit of Fig. 5*d* passes through both the optimisation stages of phase 2, then also cost improvement in the circuit is observed. Before the optimisation, the cost of the circuit was 320 and now, the design is improved further leading to a more compact representation incurring a cost of 275 (see Fig. 10*c*).

So, from this development, it can be concluded that phase 2 helps to build more improved and optimised circuits for efficient reversible logic synthesis.

4 Experimental results and analysis

We have tested the proposed approach over a wide range of benchmarks taken from [26, 35] and the obtained results are displayed in two tables. The first table (Table 1) presents a comparison between our approach and existing ESOP techniques. In that table, we have compared the quantum cost and execution time incurred for each of the benchmarks with the quantum cost of techniques [30, 36], and [25]. From that comparison, it can be seen that our approach performs fairly well than the reported ones. In the second table (Table 2), another set of comparison with non-ESOP related works (Reed–Muller-based work [37] and transformation-based work [38]) has been given and there also a steady cost improvement is observed.

In terms of scalability, the developed synthesis scheme can deal with large benchmarks as well but there are some issues related to our design approach which are discussed below.

Issue 1: In the first phase of our design, we consider the pairing of output lines while making the cost improvement in the circuits but this improvement some time is not possible if the input function is consisting of single output only. In that scenario, the circuit will be constructed directly from the cubelist. However, for the functions having high dependencies in output lines show considerable improvement in quantum cost if our design approach is followed.

Issue 2: Another point is that in the first phase of our design approach, we also prepare a table containing possible functional output pairs and their sharing information at the end of the approach (before designing the circuit). In that part, we start scanning to select best possible pairs. However, if two pairs are there that share a same number of cubes with its partner, then anyone can be selected. For example, in Fig. 4, we may select a cube pair (2, 4) and (1, 3) rather selecting (2, 3) and (1, 4). As we have set it randomly in our synthesis process, presently we are working on this to fix the issue so that definite selection criteria can be employed and the best combination of pairs can be selected.

Issue 3: In terms of scalability, the proposed scheme can handle large circuits but not beyond benchmarks with 30 variable functions and this restriction is imposed as we had to generate a '*sharing table*' by taking $\binom{n}{2}$ distinct combinations from output lines that needs some hard calculations.

5 Conclusions

This work has proposed an approach for generating improved ESOP circuits for logic functions by first pairing functional outputs and then optimising probable areas based on the structural similarities. This scheme is best suited for such circuits where there exist high inter-dependencies between functional outputs. Interestingly, the optimisation algorithm that is included in the second phase of our design scheme can also effectively be used for post-synthesis optimisations as well. Experimentally, it has been seen that the proposed scheme not only gives better results in comparison with other ESOP related works but also perform better than the non-ESOP related techniques. In terms of scalability, the proposed design approach can deal with large benchmarks as well. Further improvements in the design will be investigated in future.

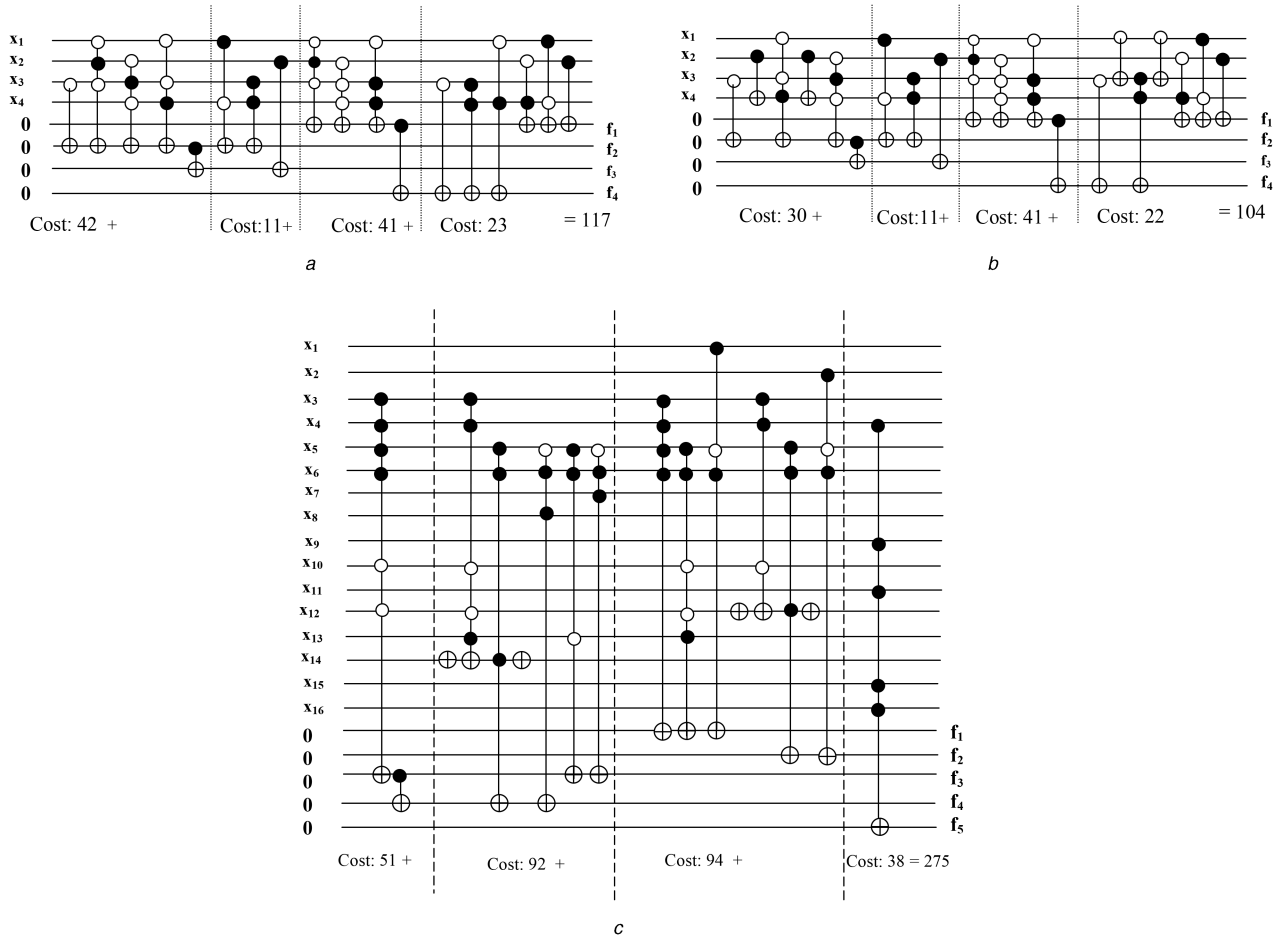


Fig. 10 *Application of algorithm2 over benchmark function pcler8_190*
(a) Circuit obtained after executing Step 1 of phase 2 over Fig. 4*d*, ***(b)*** Circuit obtained after executing step 2 of phase 2 over (a), ***(c)*** Improved ESOP circuit for function *pcler8_190*

Table 1 Comparisons with ESOP related works

Benchmark specification		QC from [30], method I	QC from [30], method II	QC from [36], method III	QC from [25], method IV	Our technique		
Function names	In/out					QC after phase 1	QC after phase 2	ET, s
3_17_6	3/3	—	46	45	**39**	39	35	<1
4gt12_24	4/1	43	43	43	**43**	42	37	<1
4gt11_23	4/1	5	5	5	**5**	5	5	<1
4gt10_22	4/1	35	35	35	—	34	34	<1
4mod7	4/3	167	169	169	**143**	147	131	<1
f2	4/4	246	—	—	**116**	210	162	<1
4_49_7	4/4	201	222	222	**119**	143	103	<1
aj-e11_81	4/4	201	217	221	**116**	128	104	<1
wim	4/7	218	—	—	**150**	206	168	<1
dc1_142	4/7	454	—	—	**201**	244	172	<1
Ex2	5/1	160	—	—	**118**	143	114	<1
Ex3	5/1	97	—	—	**73**	76	51	<1
C17	5/2	105	81	81	—	83	82	<1
cm82a	5/3	143	—	—	—	128	98	<1
rd53_68	5/3	269	—	—	**136**	246	194	<1
squar5_206	5/8	465	—	—	—	350	331	<1
C7552_119	5/16	—	2015	2015	**1123**	1535	967	<1
con1	7/2	206	—	—	**179**	163	163	<1

Benchmark specification		QC from [30], method I	QC from [30], method II	QC from [36], method III	QC from [25], method IV	Our technique		
Function names	In/out					QC after phase 1	QC after phase 2	ET, s
rd73_69	7/3	1150	—		**820**	1120	711	<1
Z4_224	7/4	642	420	412	**260**	551	334	<1
Z4ml_225	7/4	642	—	—	**260**	551	334	<1
sqrt8	8/4	616	—	—	**—**	568	320	<1
misex1_178	8/7	1012	—	—	**743**	714	624	<1
dk27_146	9/9	252	—	—	**245**	245	221	<1
max46	9/1	4968	—	—	**3239**	4524	2588	<1
add6	12/7	5757	6764	6751	—	5518	3491	<1
alu1_94	12/8	239	216	216	—	198	172	<1
t481	16/1	236	—	—	**192**	229	183	<1
pcler8	16/5	340	—	—	—	319	275	<1
mux	21/1	815	—	—	—	800	800	<1
Cordic	23/2	348,779	—	—	**98,456**	187,582	69,723	<1

'—' (dash): signifies 'non-availability of results'.

GC: gate count.

QC: quantum cost.

ET: execution time.

Table 2 Comparisons with non-ESOP related works

Benchmark specification		RMRLS [37]			RMS [38]			Proposed design			
Function name	In/out	QC	GC	ET	QC	GC	ET	QC after phase 1	QC after phase 2	GC	ET, s
decod24_10	2/4	55	11	497	19	7	<0.01	19	15	6	<0.01
mini-alu_84	4/2	173	21	495.61	248	36	<0.01	85	85	6	<0.01
rd53_68	5/3	—	—	>500.00	2646	221	0.14	246	194	18	<0.01
mod5adder_66	6/6	529	37	494.46	151	35	0.06	477	301	30	<0.01
rd73_69	7/3	—	—	>500.	20,779	1344	1.93	1120	711	44	<0.01
rd84_70	8/4	—	—	>500.	8738	124	9.92	3554	2388	29	<0.01
cycle10_2_61	12/12	1435	26	491.87	1837	41	26.17	1788	1444	42	<0.01
plus63mod4096	12/12	—	—	>500.	4873	24	17.74	956	955	27	0.01
plus127mod8192	13/13	—	—	>500.	9131	25	57.16	1462	1461	27	0.01
plus63mod8192	13/13	—	—	>500.	9183	28	57.19	1356	1227	31	<0.01

6 References

[1] Shor, P. W.: 'Algorithms for quantum computation: discrete logarithms and factoring', *Found. Comput. Sci.*, 1994, pp. 124–134

[2] Grover, L. K.: '*A fast quantum mechanical algorithm for database search*'. Theory of Computing, 1996, pp. 212–219

[3] Nielsen, M., Chuang, I.: '*Quantum computation and quantum information*' (Cambridge Univ. Press, Cambridge, 2000)

[4] Laforest, M., Simon, D., Boileau, J.C., *et al.*: 'Using error correction to determine the noise model', *Phys. Rev. A*, 2007, **75**, (1), pp. 133–137

[5] http://www.nature.com/nature/journal/v519/n7541/full/nature14270.html

[6] Landauer, R.: 'Irreversibility and heat generation in the computing process', *IBM J. Res. Dev.*, 1961, **5**, p. 183

[7] Veldhorst, M., *et al.*: 'A two qubit logic gate in silicon', *Nature*, 2014, p. 410

[8] Barenco, A., Bennett, C.H., Cleve, R., *et al.*: 'Elementary gates for quantum computation', *APS Phys. Rev.*, 1995, pp. 3457–3467

[9] Haffner, H., Hänsel, W., Roos, C.F., *et al.*: 'Scalable multiparticle entanglement of trapped ions', *Nature*, 2005, pp. 643–646

[10] Ghosh, J., Galiautdinov, A., Zhou, Z., *et al.*: 'High-fidelity CZ gate for resonator based superconducting quantum computers', *Phys. Rev. A*, 2013, **87**, p. 022309

[11] Barends, R., Kelly, J., Megrant, A., *et al.*: 'Logic gates at the surface code threshold: superconducting qubits poised for fault-tolerant quantum computing', *Nature*, 2014, pp. 500–503

[12] Kane, B.: 'A silicon-based nuclear spin quantum computer', *Nature*, 1998, pp. 133–137

[13] Saeedi, M., Wille, R., Drechsler, R.: 'Synthesis of quantum circuits for linear nearest neighbor architectures', *Quantum Inf. Process.*, 2011, **10**, (3), pp. 73–89

[14] Wille, R., Lye, A., Drechsler, R.: 'Exact reordering of circuit lines for nearest neighbor quantum architectures', *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 2014, **33**, (12), pp. 1818–1831

[15] Soeken, M., Roetteler, M., Wiebe, N., *et al.*: '*Logic synthesis for quantum computing*' (DAC, Austin, 2017)

[16] Toffoli, T.: 'Reversible computing'. Tech. Memo-MIT/LCS/TM-151, MIT Lab for Computer Science, 1980, pp. 632–644

[17] Feynman, R.: 'Quantum mechanical computers'. Foundations of Physics, 1986, vol. 16, pp. 507–531, (Originally appeared in Optics News, February 1985)

[18] Golubitsky, O., Falconer, S. M., Maslov, D.: 'Synthesis of the optimal 4-bit reversible circuits'. Proc. of the 47th Design Automation Conf., 2010, pp. 653–656

[19] Grobe, D., Wille, R., Dueck, G. W., *et al.*: 'Exact multiple-control Toffoli network synthesis with sat techniques', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, **28**, (5), pp. 703–715

[20] Chattopadhyay, A., Chandak, C., Chakraborty, K.: 'Complexity analysis of reversible logic synthesis', arXiv preprint arXiv:1402.0491, 2014

[21] Datta, K., Rathi, G., Sengupta, I., *et al.*: 'Synthesis of reversible circuits using heuristic search method'. 2012 25th Int. Conf. VLSI Design (VLSID), 2012, pp. 328–333

[22] Li, M., Zheng, Y., Hsiao, M. S., *et al.*: 'Reversible logic synthesis through ant colony optimization'. Design, Automation & Test in Europe Conf. & Exhibition (DATE), 2010, pp. 307–310

[23] Abdessaied, N., Soeken, M., Dueck, G. W., *et al.*: 'Reversible circuit rewriting with simulated annealing'. IFIP/IEEE Int. Conf. Very Large Scale Integration (VLSI-SoC), 2015, 2015, pp. 286–291

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

[25] Bandyopadhyay, C., Rahaman, H., Drechesler, R.: 'A cube pairing approach for synthesis of ESOP-based reversible circuit'. IEEE 44th Int. Symp. on Multiple-Valued Logic (ISMVL), 2014, Germany (DOI: 10.1109/ISMVL.2014.27), pp. 109–114

[26] Maslov, D.: 'Reversible logic synthesis benchmark page'. http://www.cs.uvic.ca/dmaslov/, 2002

[27] Miller, D., Soeken, M., Drechsler, R.: 'Mapping NCV circuits to optimized Clifford+T circuits'. Reversible Computation, 2014

[28] Amy, M., Maslov, D., Mosca, M.: 'Polynomial-time T-depth optimization of Clifford+T circuits via Matroid partitioning'. 6th Int. Conf. on Reversible Computation, Japan, 2014

[29] Fazel, K., Thornton, M., Rice, J. E.: 'ESOP based Toffoli gate cascade generation'. PACRIM, 2007, pp. 206–209

[30] Rice, J. E., Nayeem, N. M.: 'Ordering techniques for ESOP-based Toffoli cascade generation'. PacRim 2011, August 2011, pp. 274–279

[31] Nayeem, N. M., Rice, J. E.: 'A shared-cube approach to ESOP-based synthesis of reversible logic'. Facta Universitatis Series: Electronics and Energetics, 2011, vol. 24, pp. 385–402
[32] Datta, K., Rathi, G., Sengupta, I., *et al.*: 'An improved reversible circuit synthesis approach using clustering of ESOP cubes'. 18th Reed-Muller Workshop, 2013
[33] Jegier, J., Kerntopf, P.: 'Application of the maximum weighted matching to quantum cost reduction in reversible circuits'. MIXDES, 2017, vol. 24, pp. 224–228
[34] Mishchenko, A., Perkowski, M.: 'Fast heuristic minimization of exclusive-sums-of-products'. 6th Reed-Muller Workshop, 2001, pp. 242–250
[35] Wille, R., Grosse, D., Teuver, L., *et al.*: 'Revlib: an online resources for reversible functions and reversible circuits'. ISMVL, 2008
[36] Rice, J. E., Suen, V.: 'Using autocorrelation coefficient-based cost functions in ESOP-based Toffoli gate cascade generation'. CCECE, May 2010, pp. 1–6
[37] Gupta, P., Agrawal, A., Jha, N. K.: 'An algorithm for synthesis of reversible logic circuits', *IEEE Trans. CAD*, 2006, **25**, (11), pp. 2317–2330
[38] Maslov, D., Dueck, G. W., Miller, D. M.: 'Techniques for the synthesis of reversible Toffoli networks', *ACM Trans. Des. Autom. Electron. Syst.*, 2007, **12**, (4), p. 42