FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 16, April 2004, xx-xx

A Shared-cube Approach to ESOP-based Synthesis of Reversible Logic

N. M. Nayeem and J. E. Rice

Abstract: Reversible logic is being suggested as a possibility for overcoming potential power loss and heat dissipation problems that the computing industry may soon be at a loss to overcome. However, for reversible logic to be a solution we must have techniques for synthesizing function descriptions to reversible circuits. This paper presents an improved ESOP-based reversible logic synthesis approach which leverages situations where cubes are shared by multiple outputs and ensures that the implementation of each cube requires just one Toffoli gate. It has the potential to minimize both gate count and quantum cost, and in fact our experimental results show that this technique can reduce the quantum cost up to 75% compared to results from the existing work.

Keywords: reversible logic, logic synthesis, ESOP, Toffoli gate cascade

1 Introduction

Landauer [1] showed that as information is lost during traditional logic computation heat is generated; this is true regardless of underlying technology. This dissipated heat will cause problems in the near future if Moore's law [2] holds true, and industry publications such as the International Technology Roadmap for Semiconductors (ITRS) are beginning to publically recognize this fact [3]. Reversible logic, on the other hand, has the theoretical potential to dissipate no energy, as no information is lost during computations. According to Bennett [4], it would be theoretically possible for a circuit to dissipate zero energy if it is implemented using reversible gates. Although technologies for the use of reversible logic in everyday computing have yet to be developed, reversible computing has shown itself to be of use in quantum computing [5], low power CMOS design [6], optical computing [7],

Manuscript received FILL IN DATE HERE. An earlier version of this paper was presented at the 2011 RM Workshop, May 25–26, 2011, Tuusula, Finland.

The authors are with the University of Lethbridge, 4401 University Dr. W., Lethbridge, Alberta, Canada (email: noor.nayeem@uleth.ca; j.rice@uleth.ca)

nanotechnology [8], and bioinformatics. In fact, since all quantum computations are reversible there is a clear and direct relationship between reversible computing and quantum computing [9].

In reversible logic fan-out and feedback are not permitted, unlike in traditional logic, thus synthesis of reversible logic must follow different approaches than those that have been developed for traditional logic. There are a number of reversible logic synthesis techniques such as the transformation approach [10], the use of positive polarity Reed-Muller expressions (PPRM) [11], exclusive-or sum-of-products approaches (ESOP) [12, 13], and shared PPRM [14] techniques. Synthesis based on ESOP representations of functions is of interest because of the easy transformation of ESOP terms into a cascade of Toffoli gates, as well as the ability to handle functions with large numbers of inputs.

In this paper, we describe an optimized shared cube synthesis approach which works with the ESOP representation of a function. This approach was first presented in [15]. This approach completes very quickly and is able to handle large sizes of functions, and can work with either reversible or irreversible initial function specifications. The paper begins with some brief background topics in Section 2 and describes previous work in the area in Section 3. Section 4 describes our new approach with a number of examples, followed by which Section 5 provides experimental results and comparisons with the previous work. We conclude in Section 6 with conclusions and a discussion of future work.

2 Background

2.1 Reversible Logic

A function is reversible if it is bijective (*i.e.*, one-to-one and onto) [16]. In other words, a reversible function must have a one-to-one correspondence between its input and output vectors. A reversible gate realizes a reversible function and has the same number of inputs and outputs. A reversible circuit consists of only reversible gates which are interconnected without fanout and feedback [9].

Traditional logic gates such as AND, OR, NAND, NOR and EXOR are not reversible; the exception, however is the NOT gate, which is reversible. The most popular reversible gates are the Toffoli gate and the Fredkin gate. This work uses Toffoli gates exclusively, thus we concentrate on these in our descriptions. An n-bit Toffoli gate maps the input vector [x_1, x_2, ..., x_{n-1}, x_n] to the output vector [x_1, x_2, ..., x_{n-1}, $x_1x_2...x_{n-1} \oplus x_n$] as shown in Figure 1(a). The first (n-1) bits are known as controls and the last bit is the target. The 2-bit Toffoli gate is also known as the controlled-not, or CNOT gate, while a 1-bit Toffoli gate (with no controls) is a NOT gate. In general, the target is affected only if all of the control lines have

the value 1; however negative-control Toffoli gates have recently been proposed. A negative-control Toffoli gate can be defined as mapping the input vector [x_1, x_2, ..., x_{n-1}, x_n] to the output vector [x_1, x_2, ..., x_{n-1}, $\bar{x}_1 x_2 ... x_{n-1} \oplus x_n$] where x_1 is a negative control. All or none of the control lines may be negative controls, and an example of a 3-bit Toffoli gate with a single negative control in its first input is shown in Figure 1(b). The use of negative-control Toffoli gates simplifies a circuit by reducing the number of NOT gates [17, 18].

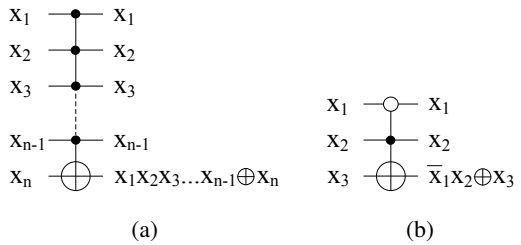


Fig. 1. (a) n-bit Toffoli gate and (b) 3-bit negative-control Toffoli gate.

Gate count is a popular cost metric used to evaluate reversible circuits. Another common metric is quantum cost. The quantum cost refers to the number of elementary (quantum) gates required to implement the circuit, as described in [17] and [18]. As described in [19], a set of basic quantum gates including all one-bit quantum gates and the two-bit exclusive-OR gate can be used to implement any reversible function. Given this fact, they describe implementations for all other (known) reversible gates in terms of these basic gates. Thus a synthesis approach may result in fewer gates, but in some cases these gates may have a higher cost if their implementation requires a higher number of these basic gates. For instance, a Toffoli gate with a large number of controls will have a more complex implementation and thus a higher quantum cost than a Toffoli gate with relatively fewer controls. A common way to calculate the quantum cost of a reversible circuit is to add the quantum costs of each of the individual gates in the circuit. In this work we use the costs given in [20] to compute the quantum cost. According to [17] the negative-control Toffoli gate with at least one positive control has the same quantum cost as a Toffoli gate; however, if all controls are negative, the negative-control Toffoli gate has an extra cost of up to 4. This means that in most instances a negative-control Toffoli gate can replace a NOT gate plus a Toffoli gate with no additional cost.

2.2 Exclusive-or Sum-of-products (ESOP)

In a sum-of-products (SOP) representation of a switching function the terms, also referred to as products or cubes, are created by ANDing one or more literal (a variable in either its complemented or non-complemented form). To create a SOP expression one or more of these terms are then combined using OR gates. The exclusive-or sum-of-products (ESOP) representation is slightly different from a SOP as the OR ($+$) operators are replaced with exclusive-or ($\oplus$) operators. For example, the function $f = xy + yz$ is expressed as a SOP, while the function $f = xy \oplus \bar{x}yz$ is expressed as an ESOP. We note that ESOP forms can represent any Boolean functions, and both $+$ and $\oplus$ are associative operators.

2.3 ESOP-based Reversible Logic Synthesis

Fazel *et al.* [12] proposed a basic ESOP-based reversible logic synthesis approach which uses only Toffoli gates to implement the circuit. A function in the ESOP form is commonly written as a list of cubes, known as a cube-list. A cube represents a term in the ESOP and has the form $x_1x_2\ldots x_p \quad f_1f_2\ldots f_q$, where x_k for $k \in \{1,2,\ldots,p\}$ is an input variable, f_j for $j \in \{1,2,\ldots,q\}$ is an output variable and p and q are the number of input and output variables of the function. An example of this format is shown in Figure 2(a).

This approach requires two input lines corresponding to positive and negative polarities of each input variable x_k and one output line (initialized by 0) for each output variable f_j. Thus the circuit initially has an empty cascade with $2p+q$ lines where p is the number of input variables and q is the number of outputs for the function to be implemented. To generate the reversible cascade implementing the function a Toffoli gate is cascaded for each cube of each output. In other words, for each output variable where $f_j = 1$, each cube is mapped to a Toffoli gate where the controls of the gate are the input lines and the target is the output line f_j. If variable x_k has the positive polarity in the cube, then the control is connected to the input line x_k. If variable x_k has the negative polarity, then the control is connected to the input line $\bar{x}_k$. Thus this approach transforms the cube-list into a cascade of Toffoli gates very quickly. Figure 2 shows an example cube-list and the cascade of gates that is generated by this approach.

Further modifications to this work [12, 13, 21] have suggested methods for reducing the required $2p$ lines to p through the use of inverters and judicious ordering of the cubes to minimize the number of inverters required.

Unlike other methods for reversible logic synthesis such as the sorting technique introduced in [22], this approach does not require that the function to be implemented be reversible. In addition, because of the simplicity of the approach

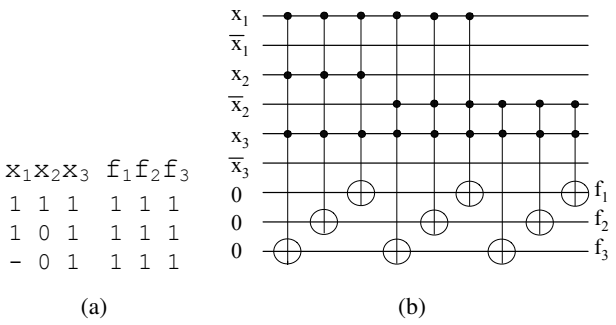


Fig. 2. a) An ESOP cube-list. b) The resulting cascade of gates when generating a circuit from the cube-list in a).

and the fact that the internal storage of the function (the ESOP cube-list) is non-exponential, very large functions can be synthesized, with results in [12] reporting synthesis of functions as large as 143 inputs and 139 outputs (for the same function). While this is not the only approach that can successfully synthesize large functions to a reversible implementation, this approach is one of the fastest and most efficient at managing very large functions.

3 Related Work

While there are a variety of approaches to reversible logic synthesis, including [22, 16, 23, 11] and [24], here we address only techniques based on the ESOP representation.

As detailed in Section 2.3, the basic ESOP-based synthesis proposed in [12] works with the ESOP cube-list and generates a circuit by transforming the ESOP cubes into a cascade of Toffoli gates; also as mentioned above a number of improvements to this approach are suggested in [12, 13] and [21].

Gupta *et al.* [11] present a reversible logic synthesis technique based on a related representation, the positive-polarity Reed-Muller (PPRM) expansion. A tree structure is generated while examining all the possible factors of a term, allowing the construction of a circuit that shares factors. The PPRM representation, while canonical, is a special type of ESOP, with a more rigorous definition, and thus will almost always have more terms than the ESOP representation used in our work. Heuristics are utilized in order to find solutions in a reasonable amount of time. Of interest to a reader new to this area, [11] also provides an overview of other synthesis approaches presented up to the date of publication.

Maslov and Dueck presented an analytical comparison between cascades of Toffoli gates and EXOR PLAs, a structure that can implement ESOP representa-

tions [25]. Their work shows that a model with minimal garbage, referred to as a "reversible cascade with minimal garbage" (RCMG) will in the worst case require a number of gates that is only a constant times larger than the number of gates needed to build a conventional EXOR PLA. Interestingly enough, they also find that there exists functions for which a linear-sized reversible cascade can be constructed, but which have exponential complexity for an EXOR PLA. Their analysis is based entirely on theoretical circuit complexity and thus there are no experimental results here for comparison.

Perkowski *et al.* [24] and [26] have also proposed ESOP-based synthesis approaches. These works use a factorization of each of the ESOPs representing the multiple outputs. In addition a new class of reversible gates is introduced, allowing modification of two qubits but requiring a significantly higher level of complexity. They report achieving good results in terms of gate numbers; for instance, in many cases they required only one gate per product term in the ESOP representation. However the technique still requires the use of some garbage lines, and as the authors themselves state, "[our] cascaded realization of multi-output ESOP generates a large number of garbage outputs and requires a large number of input constants...", both characteristics which we also are attempting to reduce in our work.

Also quite recently a newer technique referred to as the shared-cube synthesis algorithm [27, 28] was proposed. This approach generates one Toffoli gate for each cube irrespective of the number of outputs and adds CNOT gates to transfer the shared functionality to other outputs containing the cube. However, this algorithm may generate multiple Toffoli gates for a single cube if it is shared by more than two outputs. It is to this work that we provide a comparison with our own, as our work is most similar to this approach.

A brief mention of the work in [29] is also relevant; this work introduces additional lines so that even further "sharing" can be taken advantage of. In our work, however, we attempt to maximize sharing of cubes without additional lines, and as well with the least complex gates.

4 Improved Shared Cube Synthesis

Shared cube synthesis works with multi-output functions if the ESOP terms (cubes) are shared by more than one output. For instance, given a multi-output function, $f_1 = ab \oplus cd$ and $f_2 = abc$, shared cube synthesis cannot improve the circuit as there is no shared term between f_1 and f_2. The algorithm presented in [27, 28] takes the best advantage of shared functionality if the ESOP terms are shared by only two outputs. However, if the shared terms exist in more than two outputs as shown in

Figure 2(a), transformation of each term may require more than one Toffoli gate, which is inefficient. The following two examples show that the existing algorithm can be further optimized. The optimized shared cube synthesis described in this paper produces only one Toffoli gate for a cube and hence has the potential to reduce the gate count as well as quantum cost.

Example 1: Given a cube-list of a 3-input 3-output function as shown in Figure 2(a), a Toffoli cascade generated by the algorithm from [27, 28] is shown in Figure 3(a). This cascade requires two Toffoli gates for each of the cubes. An equivalent network depicted in Figure 3(b) generates one Toffoli gate for each cube, and hence minimizes the gate count by 4. Moreover, the quantum cost is reduced from 66 to 34.

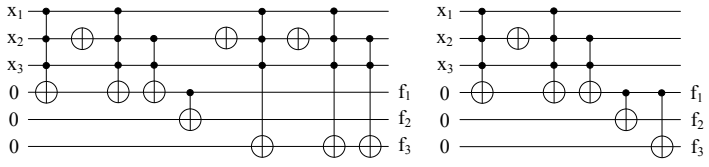


Fig. 3. (a) The Toffoli cascade generated from the cube-list in Figure 2(a) by the algorithm in [27, 28], and (b) an improved Toffoli cascade for the same cube-list.

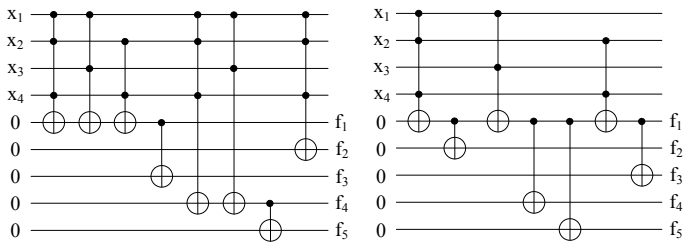
Example 2: Consider the cube-list given in Figure 4(a). The algorithm described in [27, 28] generates a Toffoli network containing three Toffoli gates for the first cube and two Toffoli gates for the second cube, a total of 8 gates as shown in Figure 4(b). However, an efficient synthesis optimizes the network as shown in Figure 4(c). The quantum cost of this network is 27, in contrast to the former approach which costs 56. This example also shows an efficient way to make use of the shared functionality even if the cubes are not shared by all the outputs.

4.1 Our Approach

Like other ESOP-based approaches, our proposed approach also works with the ESOP cube-list of a function. This approach optimizes the synthesis by generating exactly one Toffoli gate for one cube and by minimizing the CNOT gates required to transform the Toffoli gates to other output lines in the circuit. Minimization of CNOTs is performed by grouping together the Toffoli gates before passing these to other outputs via CNOTs. If the output line is not empty and a CNOT is required to remove the effect of other gates on that line before doing the transformation, we apply a technique to find a redundant CNOT gate before inserting a new one. A more detailed explanation of this approach with an example is given below.

$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
1 1 - 1	1 1 1 1 1
1 - 1 -	1 0 1 1 1
- 1 - 1	1 0 1 0 0

(a)



(b) (c)

Fig. 4. (a) An initial cube-list, (b) the Toffoli cascade generated by the algorithm in [27, 28], and (c) an improved Toffoli cascade.

In a cube-list, there may exist some cubes which are not shared by multiple outputs. If the number of 1s in the output part of a cube is one, then only one output contains this cube and no other output shares it. This cube, called an ungrouped cube, will be dealt with individually. Our proposed approach consists of the following two phases:

Phase 1: Generation of sub-lists
Phase 2: Transformation of sub-lists into gate-lists

Generation of sub-lists: This phase takes the original *cube-list* as its input and generates sub-lists as follows:

Step 1: Move ungrouped cubes from the *cube-list* into the *ungrouped-list*.

Step 2: Repeat Step 3 and Step 4 until *cube-list* is empty. Initialize the value of k with a 1 and increase its value by 1 after each iteration.

Step 3: Select a cube from the modified *cube-list* which is shared by the largest number of functions, *i.e.* has the maximum number of 1s in its output part. This cube and every other cube with an identical output part are moved to the *sub-list*$_k$.

Step 4: Select a cube from the *cube-list* which has the maximum number of 1s and which must have 0 at the output position at which the cubes in *sub-list*$_k$ have 0. In other words, the outputs that share this cube must also share all the cubes in *sub-list*$_k$. Afterwards this cube along with the cubes having identical output parts is moved from the *cube-list* to *sub-list*$_k$. This step continues until no such cube exists in the *cube-list*.

Transformation of sub-lists into gate-lists: In this phase, the sub-lists generated by phase 1 are transformed into a cascade of Toffoli gates. The *total-gate-list*, which is initially empty, will contain the final circuit at the end of this phase.

Step 1: For each *sub-list*$_k$, do Step 2 - Step 6.

Step 2: An output line p is selected as the Toffoli target line if the corresponding output contains all cubes in *sub-list*$_k$. If multiple such lines are found, choose one line arbitrarily that has not yet been used as a control or target of any Toffoli gate. If all such lines are occupied by other gates, choose the same line targeted in the last iteration, or any line arbitrarily.

Step 3: The *gate-list*$_k$ is initially empty. For each of the cubes in *sub-list*$_k$ perform steps 4-5 which add gates to the gate-list$_k$.

Step 4: Add a Toffoli gate that has a target on line p. The controls of this Toffoli gate are the input lines for which the input part of the corresponding cube contains zeros and ones. If the input part contains at least one zero, use a negative-control Toffoli gate. After that add a CNOT gate to transfer the gates to other output line(s) only if this cube is the last cube in the list that the output contains.

Step 5: If the line p has already hosted gates before the beginning of Step 2, adding CNOTs in Step 4 transfers those gates to other outputs as well. To remove this unwanted effect, also add CNOTs at the beginning of the *gate-list*$_k$. Note that insertion of this gate may cancel out another CNOT in the *total-gate-list*. If so, remove both of these gates.

Step 6: Append the *gate-list*$_k$ at the end of *total-gate-list*.

Step 7: Generate one Toffoli for each cube in *ungrouped-list* and append the gates to *total-gate-list*.

Example 3: An ESOP *cube-list* of six cubes with four input variables (x_1, x_2, x_3, and x_4) and five output variables (f_1, f_2, f_3, f_4, and f_5) is shown in Figure 5(a). The cubes are labeled C_1 to C_6. Among all the cubes only C_1 is ungrouped since the number of 1s in its output portion is 1 and so it is therefore separated from the *cube-list*. The resulting lists are shown in Figure 5(b). In the modified *cube-list*, C_3 now has the highest number of 1s in its output part. It is thus moved to the *sub-list*$_1$. Note that C_3 is not shared by f_5. Next, from the remaining cubes (C_2, C_4, C_5, and C_6), a cube is selected whose output portion contains the highest number of 1s and which is not shared by output f_5 since f_5 does not contain C_3. Although C_2, C_4, C_5, and C_6 have the same number of 1s in their output parts, C_4 and C_6 are not allowed to move in this iteration since they are shared by f_5. Between the cubes C_2 and C_5, suppose that C_2 has been selected. C_2 along with C_5 is moved to the end of *sub-list*$_1$ since the output patterns of these two cubes are identical. There are no other cubes which can be moved to *sub-list*$_1$. Figure 5(c) shows the cubes in *sub-list*$_1$ and *cube-list*.

The current *cube-list* now consists of C_4 and C_6. Both cubes have the same

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_1:	1	0	-	0	0	0	0	1	0
C_2:	1	1	-	-	1	1	1	0	0
C_3:	1	1	1	-	1	1	1	1	0
C_4:	1	0	0	1	0	1	0	1	1
C_5:	1	-	1	1	1	1	1	0	0
C_6:	1	-	0	-	1	0	1	0	1

Cube-list

(a) An initial cube-list.

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_1:	1	0	-	0	0	0	0	1	0

Ungrouped-list

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_2:	1	1	-	-	1	1	1	0	0
C_3:	1	1	1	-	1	1	1	1	0
C_4:	1	0	0	1	0	1	0	1	1
C_5:	1	-	1	1	1	1	1	0	0
C_6:	1	-	0	-	1	0	1	0	1

Current Cube-list

(b) Separation of ungrouped cubes from the cube-list.

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_3:	1	1	1	-	1	1	1	1	0
C_2:	1	1	-	-	1	1	1	0	0
C_5:	1	-	1	1	1	1	1	0	0

Sub-list$_1$

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_4:	1	0	0	1	0	1	0	1	1
C_6:	1	-	0	-	1	0	1	0	1

Current Cube-list

(c) Generation of sub-list$_1$.

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_4:	1	0	0	1	0	1	0	1	1

Sub-list$_2$

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_6:	1	-	0	-	1	0	1	0	1

Current Cube-list

(d) Generation of sub-list$_2$.

	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5
C_6:	1	-	0	-	1	0	1	0	1

Sub-list$_3$

(e) Generation of sub-list$_3$.

Fig. 5. Cube-list and its sub-lists.

number of output ones. Consider that C_4 is chosen and moved to *sub-list*$_2$. We see that f_1 contains C_6 but not C_4. As a result, C_6 is not allowed to make a group with C_4. Figure 5(d) shows the *sub-list*$_2$. Now only one cube C_6 is remaining in the *cube-list*; thus in the next iteration moving this cube to *sub-list*$_3$ shown in Figure 5(e) completes the phase 1.

In phase 2, we transform three sub-lists and ungrouped-list into a cascade of Toffoli gates. For *sub-list*$_1$ shown in Figure 5(c), outputs f_1, f_2, and f_3 have all the cubes in this list. Moreover, there are no gates on any of these output lines.

Consequently, any of these lines can be used as a target line. Let f_1 be chosen as the target line. A Toffoli gate for C_3 targeting at f_1 is generated. Since f_4 does not share any cube other than C_3 in *sub-list*$_1$, one CNOT is required to transfer C_3 from f_1 to f_4. Next two Toffoli gates are generated for C_2 and C_5. Again, to transfer all the gates from f_1 to f_2 and f_3, two CNOTs are added. The Toffoli cascade for *sub-list*$_1$ is shown in Figure 6(a).

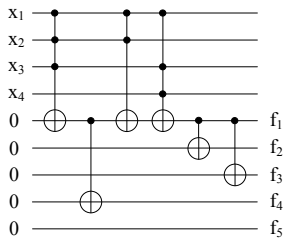
The *sub-list*$_2$, shown in Figure 5(d), has just one cube C_4 which is shared by f_2, f_4, and f_5. From Figure 6(a) we see that f_2 and f_4 are already occupied by other gates. Since the line f_5 is empty, this is chosen as the target line for *sub-list*$_2$. One negative-control Toffoli gate is added for C_4, which is transferred to f_2 and f_4 via CNOTs. Gates generated for this list are appended at the end of the cascade in Figure 6(a), which results in the circuit shown in Figure 6(b).

Next we consider the *sub-list*$_3$ in Figure 5(e), which contains one cube C_6. Outputs f_1, f_3, and f_5 share C_6, and the corresponding output lines are not empty (see Figure 6(b)). Let the target line be f_1. Since f_1 has gates on it, in order to eliminate the effect of these unexpected gates while transferring C_6 to f_3 and f_5, two more CNOTs are needed before generating the Toffoli gate for C_6. However, adding a new CNOT from f_1 to f_3 will cancel out previously the inserted CNOT (from f_1 to f_3) in the circuit shown in Figure 6(b). Therefore, this CNOT is removed rather than adding a new one (from f_1 to f_3). However a CNOT from f_1 to f_5 is required. Afterwards one negative-control Toffoli gate for C_6 and two CNOTs for sharing with f_3 and f_5 are added as shown in Figure 6(c). Finally, the ungrouped cube C_1 shown in Figure 5(b) is transformed directly into a negative-control Toffoli gate. The final cascade is shown in Figure 6(d).

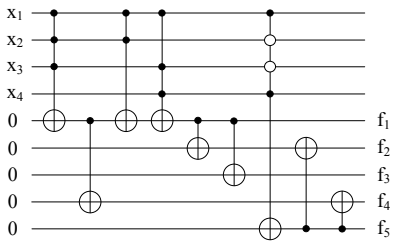
5 Experimental Results

Our proposed approach and the existing approach on shared cube synthesis discussed in [28, 27] have been developed in Java. The existing approach first reported in [28] does not use negative-control Toffoli gates; however, the usage of this type of gates was later suggested in [27] and [30]. Since our approach uses Toffoli gates including negative-control Toffoli gates, for fair comparison, negative-control Toffoli gates have also been incorporated to the circuits generated by the existing approach. The tool EXORCISM-4 [31] is used to generate the ESOP cube-lists for the benchmark circuits. The implemented programs have been run on a 2.4GHz Intel core 2 duo based system with 4GB RAM for 38 benchmark circuits collected from [32]. The execution time is negligible for both programs, and the results of this experiment are compared in Table 1.

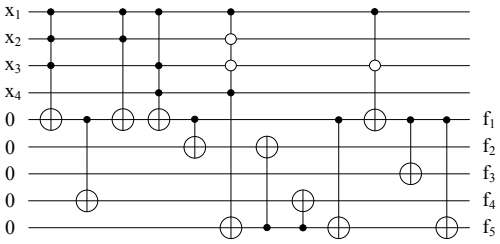
In Table 1 GC and QC stand for gate count and quantum cost, respectively. In



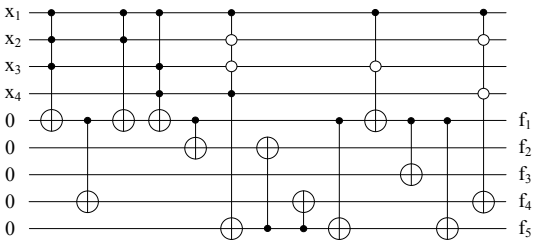
(a) A circuit equivalent to sub-list$_1$.



(b) Appending the circuit for sub-list$_2$ to the circuit of Figure 6(a).



(c) Appending the circuit for sub-list$_3$ to the circuit of Figure 6(b).



(d) A circuit equivalent to the cube-list in Figure 5(a).

Fig. 6. Improved shared cube synthesis process.

the first column, the name of the function is given. Columns two-three and four-five show the gate count and quantum cost of the circuits generated by the existing approach and the proposed approach, respectively. The last two columns indicate the improvements in percentage of the proposed algorithm over the previous one. Negative values indicate that the previous algorithm is better than the proposed one for that function.

It can be seen from the Table 1 that the proposed algorithm reduces the quantum cost for all functions except two functions 9symml and cordic. Circuits apex4, bw, ex5p, and seq are improved by more than 70% in terms of quantum cost. Moreover, a significant improvement is noticed for the functions cm42a, dc1, ham7, hwb8,

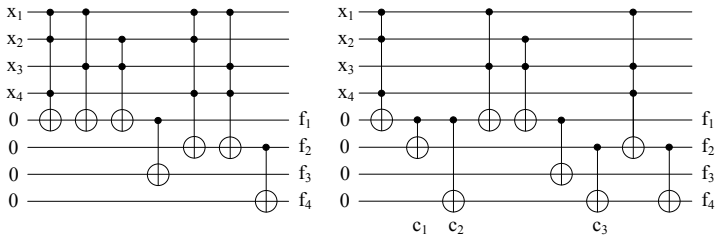
in0, inc, misex1, pdc, and urf3. It is noted that improvements for 9symml and cordic are 0%. This is due to fact that the function 9symml contains only one output; thus there is no shared functionality. For the function cordic, both approaches synthesize similar circuits due to only two outputs in the function, resultings in 0% improvement.

While our method is far better in terms of quantum cost, the opposite trend is found in the improvement column of gate count in Table 1. In our approach, 23 out of 38 circuits require more gates, and an increase of nearly 88% is noticed for apex4, which is the worst case. Since sizes of Toffoli gates used in two approaches are different, we cannot expect an accurate measurement from the gate count comparisons. This is because, while gate count is the simplest way to compare and evaluate different reversible circuits, it simply counts gates but does not take into account the complexity of the gates. As a result, it can compare different circuits only if the functionality (type) of the gates and the number of bits in the gates used in circuits are similar [33]. For example, consider two circuits where the first circuit consists of three 2-bit Toffoli gates and the second circuit consists of two 10-bit Toffoli gates. According to this measure, the second circuit is better. However, a 10-bit Toffoli gate is more complex than a 2-bit Toffoli gate. Since gates have different numbers of bits, this simple measure fails to provide an accurate comparison. This is the underlying cause of the higher gate count numbers for our approach. Since our approach utilizes the shared functionality much better than the previous approach, many large Toffoli gates are replaced by CNOT gates. However, this replacement requires some extra CNOT gates to transfer the gates to other lines and to remove the impact of other gates when the output lines are not empty. This is explained with the following example: given a cube-list in Figure 7(a), circuits generated by the existing shared cube synthesis approach and our proposed approach are shown in Figure 7(b) and Figure 7(c), respectively. The former approach requires five Toffoli gates and two CNOT gates, whereas the latter approach reduces one Toffoli gate but adds three extra CNOT gates which are labeled as c_1, c_2, and c_3. CNOT gates c_1 and c_2 are added to transfer the first Toffoli gate in Figure 7(c) to lines f_2 and f_4. Again, c_3 is added since line f_2 is not empty when the last Toffoli gate is required to transfer from f_2 to f_4. The latter approach reduces the quantum cost from 51 to 41.

The number of garbage outputs is another cost metric often used for evaluation of reversible circuits; however we note that both approaches perform exactly the same in this regard.

$x_1x_2x_3x_4$	$f_1f_2f_3f_4$
1 1 - 1	1 1 1 1
1 - 1 -	1 0 1 0
- 1 1 -	1 0 1 0
1 - 1 1	0 1 0 1

(a)



(b) (c)

Fig. 7. (a) An example cube-list, (b) a Toffoli cascade generated by the approach in [27, 28], and (c) a Toffoli cascade generated by our approach.

6 Conclusion

In this paper we describe an improved shared cube synthesis approach which incorporates an efficient way to group the cubes even when some cubes are not shared by all outputs, resulting in better transformation of cubes into gates. A technique to eliminate redundant CNOT gates has also been added to the synthesis approach to reduce the number of CNOT gates. Experimental results show that our approach can reduce the quantum cost more than 70% for some circuits, as compared to the approach in [27, 28]. We note, however, that we do not have similar improvements when considering the gate count; further examination of this phenomenon suggests that this is due to the reduced complexity of gates that our method trends toward, as compared to the more complex gates used by the previous method.

Previous work on the ESOP-based synthesis approaches has shown that application of template matching [34, 35] can significantly reduce the size of a previously synthesized reversible circuit [21]. Future work includes comparison of previous ESOP-based techniques with this work, as well as the development and incorporation of templates for negative-control Toffoli gates.

Acknowledgment

This research was funded by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Table 1. Experimental results.

Circuit	Previous Approach		Our Approach		Improvement %	
	GC	QC	GC	QC	GC	QC
5xp1	57	901	58	786	-1.75	12.76
9symml	52	10943	52	10943	0	0
alu4	474	43265	454	41127	4.22	4.94
apex4	2988	134330	5622	35840	-88.15	73.32
apex5	593	39245	601	33830	-1.35	13.8
apla	60	2345	72	1683	-20	28.23
bw	194	2379	287	637	-47.94	73.22
cordic	777	187620	777	187620	0	0
C7552	64	1023	89	399	-39.06	61
clip	87	4908	78	3824	10.34	22.09
cm42a	32	266	42	161	-31.25	39.47
cu	25	864	28	781	-12	9.61
dc1	30	213	31	127	-3.33	40.38
dc2	51	1341	51	1084	0	19.16
decod	64	1023	89	399	-39.06	61
dist	110	5079	94	3700	14.55	27.15
dk17	34	1075	34	1014	0	5.67
ex1010	1487	105579	1675	52788	-12.64	50
ex5p	428	13875	646	3547	-50.93	74.44
f2	15	175	14	112	6.67	36
f51m	332	28835	327	28382	1.51	1.57
frg2	1435	127447	1389	112008	3.21	12.11
ham7	28	108	37	67	-32.14	37.96
hwb8	462	14431	480	8195	-3.9	43.21
in0	207	13156	245	7949	-18.36	39.58
inc	62	1425	75	892	-20.97	37.4
misex1	31	586	42	332	-35.48	43.34
misex3c	849	72735	822	49720	3.18	31.64
misex3	848	73867	854	49076	-0.71	33.56
mlp4	82	2744	80	2496	2.44	9.04
pdc	542	55887	649	30962	-19.74	44.6
root	52	2436	48	1811	7.69	25.66
sao2	42	5116	41	3767	2.38	26.37
seq	1189	139826	1287	33991	-8.24	75.69
sqr6	56	708	54	583	3.57	17.66
urf3	1464	89053	1501	53157	-2.53	40.31
wim	21	177	23	139	-9.52	21.47
z4ml	33	492	34	489	-3.03	0.61

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, pp. 183–191, 1961.

[2] G. E. Moore, "Progress in digital integrated electronics," in *Technical Digest 1975 IEEE International Electron Devices Meeting*, 1975, pp. 11–13.

[3] "International Technology Roadmap for Semiconductors (ITRS)," http://public.itrs.net, 2009 executive summary.

[4] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[5] A. N. Al-Rabadi, *Reversible Logic Synthesis: From Fundamentals to Quantum Computing*. Springer-Verlag, 2004.

[6] W. C. Athas and L. J. Svensson, "Reversible logic issues in adiabatic CMOS," in *Proceedings of the Workshop on Physics and Computation (PhysComp)*, Dallas, TX, 1994, pp. 111–118.

[7] P. Picton, "Optoelectronic, multivalued, conservative logic," *International Journal of Optical Computing*, vol. 2, pp. 19–29, 1991.

[8] R. C. Merkle, "Reversible electronic logic using switches," *Nanotechnology*, vol. 4, no. 1, pp. 21–40, 1993.

[9] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[10] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings of the 40th ACM/IEEE Design Automation Conference (DAC)*, 2003, pp. 318–323.

[11] P. Gupta, A. Agrawal, and N. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2317–2330, 2006.

[12] K. Fazel, M. Thornton, and J. E. Rice, "ESOP-based Toffoli gate cascade generation," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, BC, Canada, 22-24 Aug. 2007, pp. 206–209.

[13] J. E. Rice and V. Suen, "Using autocorrelation coefficient-based cost functions in ESOP-based Toffoli gate cascade generation," in *Proceedings of the 23rd Canadian Conference on Electrical and Computer Engineering (CCECE)*, Calgary, Canada, May 2010.

[14] Y. Sanaee, M. Saeedi, and M. S. Zamani, "Shared-PPRM: A memory-efficient representation for Boolean reversible functions," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Washington, DC, USA, 2008, pp. 471–474.

[15] N. M. Nayeem and J. E. Rice, "Improved ESOP-based synthesis of reversible logic," in *Proceedings of the Reed-Muller 2011 Workshop*, Tuusula, Finland, 25–26 May 2011, pp. 57–62.

[16] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.

[17] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 436–444, 2008.

[18] M. Arabzadeh, M. Saeedi, and M. Zamani, "Rule-based optimization of reversible circuits," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2010, pp. 849–854.

[19] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computations," *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, Nov 1995.

[20] D. Maslov, "Reversible logic synthesis benchmarks page," `http://www.cs.uvic.ca/~dmaslov/`.

[21] J. E. Rice and N. Nayeem, "Ordering techniques for ESOP-based Toffoli cascade generation," in *2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim)*, Victoria, BC, Canada, 23–26 Aug. 2011, pp. 274–279.

[22] D. Maslov, G. W. Dueck, and D. M. Miller, "Toffoli network synthesis with templates," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 807–817, 2005.

[23] J. Donald and N. K. Jha, "Reversible logic synthesis with Fredkin and Peres gates," *Journal of Emerging Technology Computing Systems*, vol. 4, no. 1, pp. 1–19, 2008.

[24] M. H. A. Khan and M. A. Perkowski, "Multi-output ESOP synthesis with cascades of new reversible gate family," in *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technology (RM 2003)*, 2003, pp. 144–153.

[25] D. Maslov and G. W. Dueck, "Complexity of reversible Toffoli cascades and EXOR PLAs," in *Proceedings of the 12th International Workshop on Post-Binary ULSI Systems*, Tokyo, 2003, downloaded Mar. 2007 from http://www.cs.unb.ca/profs/gdueckreversible/esop.pdf.

[26] A. Mishchenko and M. Perkowski, "Logic synthesis of reversible wave cascades," in *Proceedings of the International Workshop on Logic Synthesis (IWLS)*, New Orleans, USA, 4–7 June 2002, pp. 197–202.

[27] Y. Sanaee, "Generating Toffoli networks from ESOP expressions," Master's thesis, University of New Brunswick, 2010.

[28] Y. Sanaee and G. W. Dueck, "Generating Toffoli networks from ESOP expressions," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, BC, Canada, 13-18 June 2009, pp. 715–719.

[29] D. M. Miller, R. Wille, and R. Drechsler, "Reducing reversible circuit cost by adding lines," in *Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, Barcelona, Spain, 26–28 May 2010, pp. 217–222.

[30] Y. Sanaee and G. W. Dueck, "ESOP-based Toffoli network generation with transformations," in *Proceedings of 40th IEEE International Symposium on Multiple-Valued Logic*, 2010, pp. 276–281.

[31] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive sum-of-products," in *Proceedings of the 5th International Reed-Muller Workshop*, Starkville, Mississippi, August 2001, pp. 242–250.

[32] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Proceedings of the 38th International Symposium on Multiple Valued Logic*, 2008, pp. 220–225, RevLib is available at `http://www.revlib.org`.

[33] M. Mohammadi and M. Eshghi, "On figures of merit in reversible and quantum logic designs," *Quantum Information Processing*, vol. 8, pp. 297–318, August 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1555567.1555601

[34] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings of the 40th ACM/IEEE Design Automation Conference*. New York, NY, USA: ACM, 2003, pp. 318–323.

[35] D. Maslov, G. W. Dueck, and D. M. Miller, "Techniques for the synthesis of reversible Toffoli networks," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 4, p. 42, 2007.