Reversible Circuit Synthesis of Symmetric Functions Using a Simple Regular Structure

Arighna Deb[1], Debesh K. Das[1], Hafizur Rahaman[2], Bhargab B. Bhattacharya[3], Robert Wille[4], and Rolf Drechsler[4]

[1] Computer Science and Engineering, Jadavpur University, Kolkata, India
arighna87@rediffmail.com, debeshd@hotmail.com
[2] Information Technology, Bengal Engg. and Sci. University, Howrah, India
rahaman_h@hotmail.com
[3] Nanotechnology Research Triangle, Indian Statistical Institute, Kolkata, India
bhargab@isical.ac.in
[4] Institute of Computer Science, University of Bremen, Bremen, Germany
Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
rwille@informatik.uni-bremen.de, drechsler@uni-bremen.de

Abstract. In this paper, we introduce a new method to realize symmetric functions with reversible circuits. In contrast to earlier methods, our solution deploys a simple and regular cascade structure composed of low-cost gates which enables significant reductions with respect to quantum costs. However, the number of garbage outputs increases slightly. To overcome this, we next propose an optimized design by reusing the garbage outputs. The resulting design thus offers a powerful approach towards reversible synthesis of symmetric Boolean functions.

Keywords: Quantum computation, Reversible logic, Symmetric functions.

1 Introduction

Reversible computing has become one of the major research areas in the recent times. Reversible logic has found applications in quantum computing [1, 2], low power design [3, 4], optical computing [5], DNA computing [6], as well as in nanotechnology [7]. These promising applications mandate new solutions for design automation of the emerging classes of circuits and systems.

Among the various research problems related to the field of reversible circuit design, logic synthesis has received significant attention. A number of reversible synthesis methods has been proposed for this purpose [8–16]. Usually, they aim for reducing the quantum costs, i.e. the number of elementary operations to be conducted in a quantum device, as well as the number of garbage outputs, i.e. output connections that are sometimes required to ensure reversibility but are not utilized to represent the desired function.

In this paper, we address the problem of synthesizing symmetric Boolean functions using reversible logic. These special types of functions have many applications to cryptology and to the design of secured systems, control and communications circuits. Accordingly, synthesis methods for such functions have been

G.W. Dueck and D.M. Miller (Eds.): RC 2013, LNCS 7948, pp. 182–195, 2013.
© Springer-Verlag Berlin Heidelberg 2013

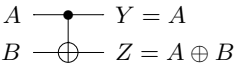


Fig. 1. Feynman (CNOT) Gate

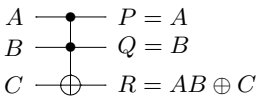


Fig. 2. Toffoli Gate

studied extensively [17–21]. Realizations of symmetric functions by reversible logic gates have been described in [9,22,23]. Picton used Fredkin gates to realize digital summation threshold logic (DSTL) devices [23]. An efficient realization of arbitrary symmetric functions using a Reversible Programmable Gate Array (RPGA) has been proposed in [9,22].

We propose a new approach for designing symmetric functions using an array of Peres gates. Our solution uses simpler reversible gates compared to previously introduced designs [9,13] and is inspired by a regular structure proposed in [19]. This yields a significant reduction in the quantum cost. However, the number of garbage outputs increases slightly. This is eventually addressed by proposing an optimization of the regular structure that enables reuse of garbage outputs and hence, leading to a reduction of them. The benefits of the proposed design is demonstrated by comparing ours with the solutions obtained by previously proposed techniques [9,13].

The rest of the paper is organized as follows. In Section 2, we provide the basics of reversible functions, reversible gates, and symmetric functions. Section 3 introduces the proposed regular structure as well as its optimization. Based on that, Section 4 describes how general symmetric functions can be realized with this structure. Finally, the resulting design is compared to previous work in Section 5 and the paper is concluded in Section 6.

2 Preliminaries

2.1 Reversible Logic Functions

A function f is said to be reversible if and only if $f : \mathbb{B}^n \rightarrow \mathbb{B}^n$ over variables $X = (x_1, x_2, \cdots, x_n)$ maps each input to a unique output and if f has the same number of input and output variables. It implies that there are 2^n input rows and 2^n output rows in the truth table of f and the output rows are the permutation of the input rows. We use the notation $(n \times n)$ to represent an n-input reversible function f.

2.2 Reversible Logic Gates

A reversible circuit is a fan-out free cascade of reversible gates. The common reversible gates include the Feynman gate, the Fredkin gate, the Toffoli gate, and the Peres gate.

Feynman Gate:- A (2×2) Feynman gate (FG), also known as controlled-NOT gate or simply CNOT gate, is shown in Fig. 1. It has two inputs, known as the

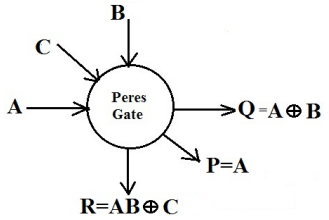


Fig. 3. Peres Gate

control input (A) and the target input (B), respectively. The logical relationship between inputs and outputs can be written as: $Y = A$, $Z = A \oplus B$.

Toffoli Gate:- A multiple control Toffoli gate (TG) t_m has the form $t_m(C, t)$, where $C = \{x_{i_1}, x_{i_2}, \cdots, x_{i_m}\} \subset X$ is the set of control lines and $t = \{x_j\}$ with $C \cap t = \emptyset$ is the target line. The value of t is inverted if and only if all control lines are set to 1. For $m = 0$ and $m = 1$, the gates are called NOT and CNOT, respectively. Fig. 2 illustrates the Toffoli gate with three inputs (A, B, C) and three outputs (P, Q, R), where (A,B) are control inputs that are unaffected by the action of the Toffoli gate. The third input is a target input (C) that is inverted if both, A and B, are 1 and otherwise remains unchanged. Thus, we get $P = A$, $Q = B$, $R = C \oplus AB$.

Peres Gate:- Fig. 3 shows a 3×3 Peres gate (PG). This gate performs the following operation: $P = A$, $Q = A \oplus B$, $R = C \oplus AB$, where the outputs are denoted as (P,Q,R) and inputs are denoted as (A,B,C).

Besides that, the following definitions related to reversible circuits are important in this work.

Control input and target input:- A reversible gate consists of two sets of inputs: control set and target set. If at least one control line is set to 0, then nothing happens to the target lines. If instead all control lines are set to 1, then the gate function is applied to the target line.

Constant input:- A constant input of a reversible function is a fixed input value (either 0 or 1).

Garbage outputs:- They refer to the outputs that are not assigned a certain function value. Garbage outputs are very much essential without which reversibility cannot be achieved for irreversible functions. For example, an AND operation of the two inputs A and B can only be achieved using the structure in Fig. 2 with C=0. In this example, the unused outputs P and Q are garbage outputs.

Quantum Cost (QC):- For its operation, a reversible gate offers a quantum cost given by the number of elementary quantum operations, which are performed by elementary quantum gates called as controlled-NOT (CNOT) gate, controlled-V gate, controlled-V+ gate, etc.; each having quantum cost of unity. The quantum costs of different reversible gates are shown in Table 1.

Table 1. Quantum cost

Reversible Gate	Quantum Cost
CNOT gate	1
TOF(a,b;c)	5
TOF(a,b,c;d)	14
PERES gate	4
Fredkin gate	5

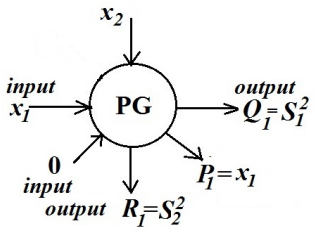


Fig. 4. Design for 2-inputs

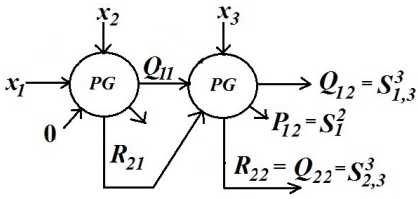


Fig. 5. Design for 3-inputs

2.3 Symmetric Functions

A switching function over n variables is a function $f(x_1, x_2, \cdots, x_n) : Q^n \rightarrow Q$, where Q denotes the set that consists of two values $\{0, 1\}$. A switching function $f(x_1, x_2, \cdots, x_n)$ is totally symmetric if it is unchanged by any permutation of its variables $(x_1, x_2, \cdots, x_n)$.

For a symmetric function, it is sufficient to specify the number of inputs that are to be set to logic 1 for the function to be 1. An n-variable symmetric function is represented as $S^n(A)$, where A is a set of integers $(a_i, \cdots, a_j, \cdots, a_k)$ and $\forall a_i, a_j, 1 \leq a_i, a_j \leq n, a_i \neq a_j$. This is denoted by $S^n_{a_i,\cdots,a_j,\cdots,a_k}$. For n variables, $2^{n+1}-2$ different symmetric functions (excluding constant functions 0 and 1) can be constructed. If the set A contains only consecutive integers $(a_l, a_{l+1}, \cdots, a_q)$ with $a_l < a_q$, the symmetric function is called consecutive symmetric function and denoted by $S^n_{a_l - a_q}$. A totally symmetric function $S^n(A)$ can be expressed as a union of maximal consecutive symmetric functions, such that $S^n(A) = S^n(A_1) + S^n(A_2) + \cdots + S^n(A_m)$, with m being the minimum and such that $\forall i, j, 1 \leq i, j \leq m, A_i \cap A_j = \emptyset$, whenever $i \neq j$.

Example 1. $S^{15}_{4,5,6,7,12,13,14,15}$ can be written as the summation of two consecutive symmetric functions S^{15}_{4-7} and S^{15}_{12-15}.

3 Synthesis of Symmetric Boolean Functions

In this section, we present our approach to the synthesis of symmetric Boolean functions as reversible circuits. First, we introduce the proposed regular structure followed by possible optimization. This builds the basis of a generic synthesis scheme for general symmetric functions, which is outlined in the next section.

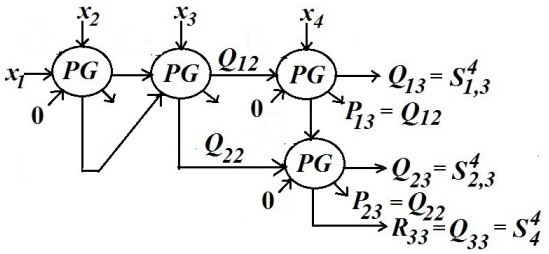


Fig. 6. Design for 4-inputs

3.1 The Proposed Regular Structure

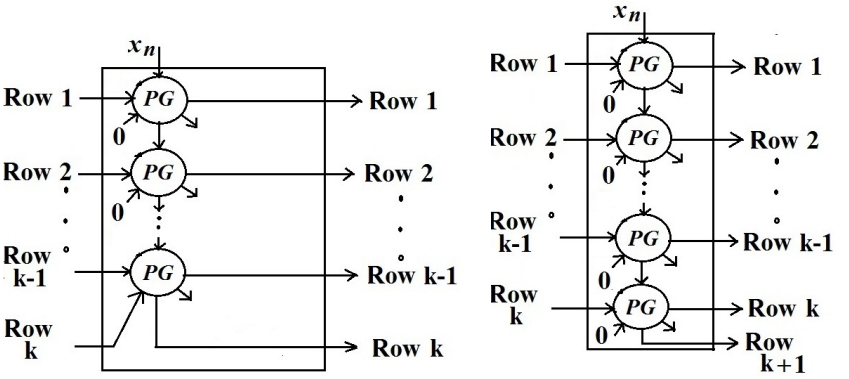
Our design consists of an array of Peres gates. The reversible gates in the design are thereby arranged as a matrix, i.e. in the form of rows and columns. In the following, this is illustrated for certain values of n, i.e., for different input sizes.

Consider the design for $n = 2$ inputs, i.e. for x_1 and x_2. The design is composed of a single (3×3) Peres gate. Throughout the design, the input line C of the Peres gate is assigned a value 0. Hence, a structure as depicted in Fig. 4 results. When C is set to 0, the Peres gate produces the following outputs: $P_1 = x_1$, $Q_1 = x_1 \oplus x_2$, and $R_1 = x_1 x_2$. Thus, this design produces two symmetric Boolean functions, namely $Q_1 = S_1^2$ and $R_1 = S_2^2$. The output P_1 is a garbage output.

Consider the design for $n = 3$ inputs, i.e., for x_1,x_2, and x_3. In this case, the design deploys two (3×3) Peres gates. There are two rows and two columns. In the first row, we have two Peres gates, whereas the second row does not contain any gate. The design is shown in Fig. 5. In this case, the output Q_{11} from the first Peres gate is given as one of the inputs to the second Peres gate in the first row. The other two inputs of the second Peres gate are x_3 and R_{21} (output of the 1^{st} Peres gate). Therefore, the outputs that are obtained from the 2^{nd} Peres gate are: $P_{12} = Q_{11}, Q_{12} = Q_{11} \oplus x_3, R_{22} = Q_{11} x_3 \oplus R_{21}$. Here, also P_{12} is the garbage output. The output R_{22} appears in the second row as Q_{22}. This structure already realizes two other symmetric functions, namely $Q_{12} = S_{1,3}^3$ and $Q_{22} = S_{2,3}^3$.

Consider the design for $n = 4$ inputs, i.e., for x_1,x_2, x_3, and x_4. Then, the top row contains three Peres gates, whereas the second row contains a single Peres gate. Fig. 6 shows this design. At the first row, the output Q_{12} of the 2^{nd} Peres gate is given as input to the 3^{rd} Peres gate along with inputs x_4 and 0. This produces outputs $P_{13} = Q_{12}$, $Q_{13} = (Q_{12} \oplus x_4)$, and $R_{23} = Q_{12} x_4$. The output R_{23} from the top row and the output Q_{22} from the previous column appear as inputs to the Peres gate at the second row, which generates $Q_{23} = Q_{22} \oplus R_{23}$. This equals to $S_{1,3}^4$. The third row has the output $R_{33} = Q_{33} = Q_{22} R_{23}$ realizing S_4^4.

It may be observed that, for $n = 2$ $(n = 3)$, there are two output lines producing S_1^2 and S_2^2 ($S_{1,3}^3$, $S_{2,3}^3$). For $n = 4$, there are three output lines producing $S_{1,3}^4$, $S_{2,3}^4$, S_4^4. Hence, a regular structure results where each output line corresponding to a row in the design produces a certain symmetric Boolean function.



(a) circuit at $(n-1)^{th}$ level for $2^{k-1}<n<2^k$ (b) circuit at $(n-1)^{th}$ level for $n = 2^k$

Fig. 7. The circuit structure at $(n-1)^{th}$ level

This can be generalized as follows: Consider the structure to be designed for n inputs, i.e. for $x_1, \ldots, x_n$. Here, we can think of different columns, where columns may be termed as levels and respectively introduce a new input x_i. Hence, the inputs x_1 and x_2 are considered in the 1^{st} level, the input x_3 in the 2^{nd} level, the input x_4 in the 3^{rd} level, and so on. Notice that the network for $n = 4$, subsumes the complete structure for $n = 3$. If we have the circuit for any $n = i$, then the circuit for $n = i + 1$ can be obtained by appending one more level. Thus, there are $(n-1)$ columns or levels in the array. Therefore, for a circuit structure of n input lines, there are k rows and $(n-1)$ columns with $k = \lfloor \log_2 n \rfloor + 1$. The circuit structure to be appended after the $(n-2)^{th}$ level is shown in Fig. 7(a) for $2^{k-1}<n<2^k$ and in Fig. 7(b) for $n = 2^k$. Let the inputs to the $(n-1)^{th}$ level be represented as $Y^1_{n-1}, Y^2_{n-1}, \cdots, Y^k_{n-1}$. Then, the outputs after the $(n-1)^{th}$ level can be recursively determined using following relation:

$$Y^i_n = Y^i_{n-1} \oplus y^i_n \text{ for } 1 \leq i \leq k \tag{1}$$

where

$$Y^i_0 = 0 \tag{2}$$

$$\begin{aligned} y^i_n &= x_n \text{ for } i = 1 \\ &= Y^{i-1}_{n-1} y^{i-1}_n \text{ for } i>1 \end{aligned} \tag{3}$$

It can be observed that a new row is added to the design for every $n = 2^k$ input variables, where $(k = 2, 3, 4, ...)$. Thus, for any n, the number of rows is $\lfloor \log_2 n \rfloor + 1$. For $2^{k-1}<n<2^k$, the k^{th} row does not contain any gate. The output R from the Peres gate of the $(k-1)^{th}$ row is given as the target line to the Peres gate in the same row and next column. This results in a cascade of Peres gates with quantum costs of $2m + 2$ [24], where m is the number of Peres gate in the cascade. For $n = 2^k$, the k^{th} row contains CNOT gates except the last one where

a Peres gate is used. In this case, there are $(k+1)$ outputs. The output in the $(k+1)^{th}$ row appears from the output R of the Peres gate of the preceding row. Hence, there is no gate in the $(k+1)^{th}$ row. For n-input variables, the entire design contains only Peres gates. For example, the circuit structure for $n = 8$ is shown in Fig. 8.

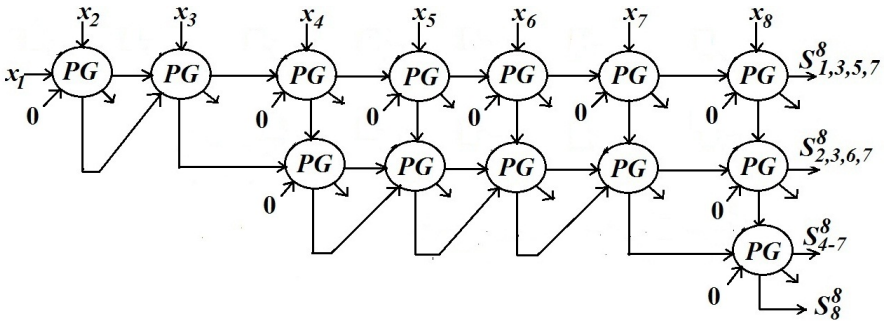


Fig. 8. Circuit for 8-inputs

Following the structure outlined above, $k = (\lfloor \log_2 n \rfloor + 1)$ symmetric functions are produced. The $i^{th}(1 \leq i \leq k)$ output line represents thereby the symmetric function $S^n_{a_{i_1},a_{i_2},\cdots,a_{i_q}}$, where each a_{i_j} is an integer whose binary representation has a 1 in the i^{th} bit. Apart from these symmetric functions produced at k output lines, the Boolean functions realized by the garbage outputs are also symmetric with a fewer number of literals. For $n = 8$, the corresponding inputs, outputs, and garbage outputs at the different levels of the network are listed in Table 2.

Table 2. Inputs and outputs at each level

Levels	Inputs	Outputs	Garbage Outputs
1	x_1, x_2	S^2_1, S^2_2	x_1
2	x_3	$S^3_{1,3}, S^3_{2,3}$	S^2_1, S^2_2
3	x_4	$S^4_{1,3}, S^4_{2,3}, S^4_4$	$S^3_{1,3}, S^3_{2,3}$
4	x_5	$S^5_{1,3,5}, S^5_{2,3}, S^5_{4,5}$	$S^4_{1,3}, S^4_{2,3}, S^4_4$
5	x_6	$S^6_{1,3,5}, S^6_{2,3,6}, S^6_{4-6}$	$S^5_{1,3,5}, S^5_{2,3}, S^5_{4,5}$
6	x_7	$S^7_{1,3,5,7}, S^7_{2,3,6,7}, S^7_{4-7}$	$S^6_{1,3,5}, S^6_{2,3,6}, S^6_{4-6}$
7	x_8	$S^8_{1,3,5,7}, S^8_{2,3,6,7}, S^8_{4-7}, S^8_8$	$S^7_{1,3,5,7}, S^7_{2,3,6,7}, S^7_{4-7}$

For an n-input function, the total number of Peres gates is therefore given by

$$N_{PG} = (n\lceil \log_2 n \rceil - n - 2^{\lceil \log_2 n \rceil} + \lfloor \log_2 n \rfloor + 2) \tag{4}$$

The design requires $n\lceil \log_2 n \rceil - 2n - 2^{\lceil \log_2 n \rceil} + 2\lfloor \log_2 n \rfloor + 3$ constant inputs (fixed to 0). It may be observed that the total number of garbage in the design

is equal to the sum of the total number of Peres gates. Therefore, the design produces

$$N_{garbage} = N_{PG} \tag{5}$$

garbage lines for any n.

3.2 Further Optimization of the Proposed Structure

The proposed regular structure generates a large number of garbage lines. The design can further be improved in this respect. Notice that once the garbage lines are used as control lines, they no longer play any role in the circuit. Therefore, the structure can be improved by reusing the garbage lines as target or control lines in the rest of the circuit.

The resulting reversible circuit for $n = 4$ inputs is redrawn in Fig. 9 (the original realization is depicted in Fig 6). The circuit remains the same for up to $n = 3$ inputs, i.e., the 1^{st} row produces the output $Q_{12} = S^3_{1,3}$ and the 2^{nd} row produces the output $R_{22} = S^3_{2,3}$. For $n = 4$, now a (3×3) Toffoli gate with two control lines Q_{12} and x_4 is added to the design. This produces the output $T_{23} = Q_{12}x_4 = S^3_{1,3}x_4$. Since the design adds a new row to the structure at $n = 4$, the 2^{nd} row will have a Peres gate producing outputs at the 2^{nd} and the 3^{rd} rows. This Peres gate takes R_{22} and T_{23} as control inputs and produces $R_{33} = R_{22}T_{23} = S^4_4$ at the 3^{rd} row with the target input line set to 0. It also produces the output $Q_{23} = T_{23} \oplus R_{22} = S^4_{2,3}$ at the 2^{nd} row. Now, a Peres gate is introduced in the 1^{st} row, which works on the same set of control and target lines. This means that the Toffoli gate with two control lines present in the Peres gate is an exact replica of the (3×3) Toffoli gate added previously in the 1^{st} row. This makes the garbage output T_{23} to become zero. Later, this line can be reused again in the circuit.

In general, for $2^{k-1} \leq n < 2^k$ $(k \neq 1, 2)$, the optimized design inserts a single Toffoli gate before every Peres gate present in the $(k-1)^{th}$ row. Initially, a Toffoli gate in the $(k-1)^{th}$ row and the j^{th} $(1 \leq j \leq n-1)$ column produces an output which appears as one of the inputs to the Peres gate in the k^{th} row and the j^{th} column. Once this output line is used, and if it is not required any more, it becomes a garbage output. The presence of a Peres gate in the $(k-1)^{th}$ row results in a structure of Toffoli gates followed by another Toffoli gate and a CNOT gate (since Peres gate is equivalent to a Toffoli gate followed by CNOT gate) in the same row. The two back-to-back Toffoli gates work on the same set of control lines and target line. Since the Peres gate preceded by the Toffoli gate in the same row works on the same set of control and target lines, this makes the garbage output to become zero. Therefore, this line can now be reused as a target line to other gates in the next level in the structure as shown in Fig. 10 for $n = 5$ inputs. Note that any row where a Peres gate and a Toffoli gate share the same target line and one of the control lines, a so called Peres-Toffoli double gate can be applied. The quantum cost of such a gate is 7 [24].

This optimization technique results in $(2\lfloor \log_2 n \rfloor - 1)$ constant input lines for an n-input design, which is less than that of the original structure. On the

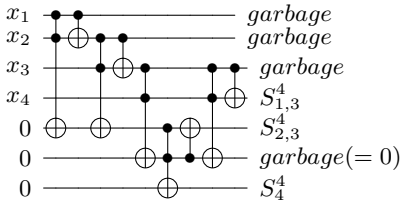


Fig. 9. Optimized design for 4-inputs

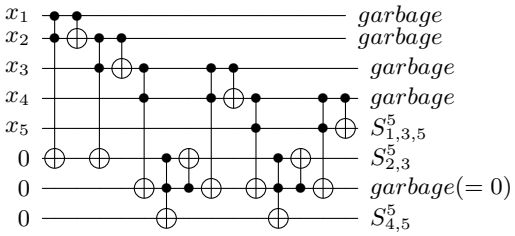


Fig. 10. Optimized design for 5-inputs

contrary, the optimized design requires some additional Toffoli gates along with the Peres gates of the main structure. The total number of Toffoli gates in the design is given by

$$N_{toffoli} = n\lfloor \log_2 n \rfloor - n - 2^{\lfloor \log_2 n \rfloor + 1} + \lfloor \log_2 n \rfloor + 3 \tag{6}$$

while the total number of Peres gates remains the same as in the original design.

The total number of garbage lines required for the optimized design is given as

$$N_{reduced-garbage} = n + \lfloor \log_2 n \rfloor - 2. \tag{7}$$

Comparing this with the result shown in Equation (4), we observe that the number of garbage lines in the optimized structure is less than that of the original structure.

4 Reversible Synthesis of General Symmetric Functions

For any n inputs, the proposed structure produces $\lfloor \log_2 n \rfloor + 1$ number of symmetric functions. Two symmetric functions S^n_A and S^n_B are true for any weight w of input vectors, $w = (1, 2, \cdots n)$, if $A \cap B = \{w : w \in A \text{ and } w \in B\}$, where A and B are a set of integers containing the Hamming weights of the input vectors. The aim here is to separate these common weights between any two symmetric functions and represent all the symmetric functions in terms of individual weight of its input vector. To do this, the output lines of the regular structure are fed to a network consisting of a number of blocks called *extraction-elimination (EE)* modules.

4.1 Extraction-Elimination (EE) Module

As the name implies, this module performs two operations: the first one is an "extraction", which extracts the common weight of an input vector from two symmetric functions for which the functions are true. The second one is an "elimination", which eliminates the common weight from those two symmetric

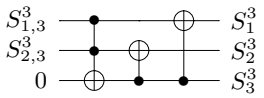


Fig. 11. *Extraction-Elimination* module: $S^3_{1,3}, S^3_{2,3}$ are inputs along with target line set to 0 and S^3_1, S^3_2, S^3_3 are outputs

functions. This module produces three symmetric functions of single weight. The extraction operation is implemented using a Toffoli gate whose target line is set to 0. The two elimination operations (one for each symmetric function) are performed using two CNOT gates. The quantum cost of this module is 7. The complete module is shown in Fig. 11. It is a garbage-free circuit where each output line is essential.

4.2 Realization of General Symmetric Functions

The *EE module* is used to decompose $k = \lfloor \log_2 n \rfloor + 1$ symmetric functions of multiple weights realized by the regular structure described in Section 3 into n symmetric functions of single weight of its input vector, where $k < n$. This is done using the following procedure:

1) First, the regular structure as described in Section 3 is constructed for n inputs. The structure produces $k = \lfloor \log_2 n \rfloor + 1$ outputs, each of which is a symmetric function of n inputs. For any given regular structure, we represent integers 1 to n with its binary equivalent, i.e., for any n, the bit positions are $(2^{\lfloor \log_2 n \rfloor} \cdots 2^3 2^2 2^1 2^0)$, where $2^{\lfloor \log_2 n \rfloor}$ is the most significant bit of the number n. Each bit position of the decimal number n indicates an output line of the regular structure. Hence, there are $\lfloor \log_2 n \rfloor + 1$ outputs in the regular structure. The total number of $1'$s present in any bit position 2^m, where $m = 0, 1, 2, \cdots \lfloor \log_2 n \rfloor$ indicates the corresponding output line of the regular structure realizing a symmetric function. If a 1 is present at the bit position 2^m, which is an *MSB*, then the regular structure has at most $m + 1$ output lines. It is noticed that the integers 1 to n denote the weights of the input vector for which the functions are true. Once all the integers are represented in their equivalent binary forms, the process of identification follows.

2) During this process, we identify all the 2^m bit positions that are 1 for binary equivalents of all the consecutive integers 1 to n. This helps in indicating all the corresponding output lines realizing symmetric functions, i.e., they are true for that integer (weight of the input vector). Two cases related to the identification of bit positions are considered. The first case implies if any one of the 2^m bit positions is 1 and rest of the bit positions are 0. Then, the symmetric function in the corresponding output line is true only for that integer (weight of the input vector). In the second case, if more than one bit positions are 1, then the corresponding output lines realizing the functions, are true. Whenever the second case is encountered, the output lines of the regular structure are identified

Table 3. Binary representations of five consecutive numbers

Decimal number	its binary equivalent
1	0001
2	0010
3	0011
4	0100
5	0101

from the bit positions of the binary number equal to 1. Now an *EE* module is applied to the two lines indicated by the two bit positions. This results in the extraction of the corresponding integer value. The extracted integer is copied at the target line of a (3×3) Toffoli gate by setting the line to 0. This integer is a weight of the input vector for which the functions in those two lines are set to 1. Following the extraction operation, the elimination operations are performed on these two lines by two CNOT gates, one for each line. This results in three symmetric functions with no common input weight. This process continues until all the integers are considered.

Following this procedure, we require $(n - \lfloor \log_2 n \rfloor - 1)$ number of *EE* modules for an n-input structure to convert $\lfloor \log_2 n \rfloor + 1$ symmetric functions of multiple weights to n symmetric functions of single weight.

Example 2. Consider the regular structure for $n = 5$ inputs. There are three output lines producing outputs $f_1 = S^5_{1,3,5}, f_2 = S^5_{2,3}$ and $f_3 = S^5_{4,5}$ on line $1, 2,$ and 3 respectively. The possible weights of the input vector and their binary equivalents are shown in Table 3. From the table, it can be observed that the first binary number has a single 1 at its bit position 1. This represents weight 1 for which f_1 will be true. Similarly, the second binary number has a single 1 at its bit position 2, thus representing weight 2 for which f_2 is true. In the third binary number, we have two 1's - one in bit position 1 and another in bit position 2, indicating weight 3 for which the functions in line 1 (f_1) and line 2 (f_2) are true. Therefore, we append an *EE* module to line 1 and 2 which produces three outputs- $S^5_{1,5}$ at line 1, S^5_2 at line 2, and S^5_3 at line 4. In the fourth binary number, there is a single 1 at bit position 3 meaning that the output line 3 is true for weight 4. In the last binary number, we observe that there are two 1's in bit positions 1 and 3. Thus, line 1 and line 3 are now applied to the *EE* module which produces outputs S^5_1 at line 1, S^5_4 at line 3, and S^5_5 at line 5. Therefore,

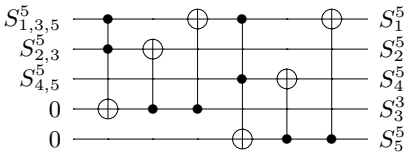


Fig. 12. *EE* modules appended at the end of the regular structure for 5 inputs

using two *EE* modules five symmetric functions of single weight are produced as shown in Fig. 12.

5 Comparison to Previous Work

We have compared the cost metrics of the proposed regular and the optimized structure with those reported in previous work [9, 13]. The comparison is made on the basis of quantum cost and the number of garbage lines. The results are reported in Table 4. We observe that the realizations of benchmark functions obtained by the first technique have less quantum costs as compared to those reported in previous work [9, 13]. The number of garbage bits in this design is larger in comparison to those from [13], but fewer than those of [9]. However, these garbage outputs also implement symmetric functions with a fewer number of literals and, thus, can be utilized to synthesize other symmetric functions. Furthermore, by slightly increasing the quantum costs, the number of garbage lines can further be reduced using the proposed optimization technique.

Table 4. Comparison of quantum cost

Function			Quantum cost				Garbage			
Name	In	Out	[9]	[13]	Sect. 3.1	Sect. 3.2	[9]	[13]	Sect. 3.1	Sect. 3.2
$rd53$	5	3	145	36	20	28	15	5	6	5
$rd73$	7	3	303	64	32	46	30	7	10	7
$rd84$	8	4	403	98	44	66	39	11	13	9
$9sym$	9	1	505	94	59	88	37	11	19	14

6 Conclusion

In this paper, we have proposed a synthesis scheme for realizing symmetric Boolean functions with reversible logic. Compared to earlier synthesis methods, our solution relies on a simple and regular cascade structure. The garbage outputs of our design can also be used to realize symmetric Boolean functions with a fewer number of literals. We have evaluated the proposed design on some well known benchmark symmetric functions. Our simulation results reveal that the proposed design significantly reduces the quantum cost, but may require additional ancillary lines thereby increasing the number of garbage outputs. To reduce these garbage lines further, we have also proposed a modified structure in which these garbage lines can be properly reused while implementing the output functions. Both of these design approaches admit a hierarchical structure and can thus be built in an iterative fashion. This regular structure thus obtained can be fed to a network of extraction-elimination *(EE)* modules to synthesize symmetric functions of single weights from those having multiple weights. The *EE* network is an entirely garbage-free network.

Acknowledgement. This work was partly supported by CSIR grant (ref.-22(0590)/12/$EMR-II$) and UGC MRP grant (ref.-41 − 620/2012(SR)).

References

1. Knill, E., Laflamme, R., Milburn, G.J.: A scheme for efficient quantum computation with linear optics. Nature, 46–52 (2001)
2. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge Univ. Press (2000)
3. Wille, R., Drechsler, R., Oswald, C., Garcia-Ortiz, A.: Automatic design of low-power encoders using reversible circuit synthesis. In: DATE, pp. 1036–1041 (2012)
4. Desoete, B., Vos, A.D.: A reversible carry-look-ahead adder using control gates. INTEGRATION, the VLSI Jour. 33(1-2), 89–104 (2002)
5. Cuykendall, R., Andersen, D.R.: Reversible optical computing circuits. Optical Letters 12(7), 542–544 (1987)
6. Thapliyal, H., Srinivas, M.B.: The need of DNA computing: reversible designs of adders and multipliers using fredkin gate. In: Proc. SPIE, Optomechatronic Micro/Nano Devices and Components (2005)
7. Merkle, R.C.: Reversible electronic logic using switches. Nanotechnology 4, 21–40 (1993)
8. Agarwal, A., Jha, N.K.: Synthesis of reversible logic. In: DATE, pp. 21384–21385 (2004)
9. Perkowski, M., Kerntopf, P., Buller, A., Chrzanowska-Jeske, M., Mishchenko, A., Song, X., Al-Rabadi, A., Jozwiak, L., Coppola, A., Massey, B.: Regularity and symmetry as a base for efficient realization of reversible logic circuits. In: IWLS, pp. 245–252 (2001)
10. Mishchenko, A., Perkowski, M.: Logic synthesis of reversible wave cascades. In: IWLS, pp. 197–202 (2002)
11. Gupta, P., Agrawal, A., Jha, N.: An algorithm for synthesis of reversible logic circuits. IEEE TCAD 25(11), 2317–2330 (2006)
12. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE TCAD 22(6), 723–729 (2003)
13. Maslov, D.: Efficient reversible and quantum implementations of symmetric Boolean functions. IEEE Proc. of the Circuits, Devices and Systems 153(5), 467–472 (2006)
14. Große, D., Wille, R., Dueck, G.W., Drechsler, R.: Exact multiple control Toffoli network synthesis with SAT techniques. IEEE TCAD 28(5), 703–715 (2009)
15. Miller, D.M., Maslov, D., Dueck, G.W.: A transformation based algorithm for reversible logic synthesis. In: Design Automation Conf., pp. 318–323 (2003)
16. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic for large functions. In: Design Automation Conf., pp. 270–275 (2009)
17. Rovetta, C., Mouffron, M.: De Bruijan sequences and complexity of symmetric functions. Cryptography and Communications Journal 3(4), 207–225 (2011)
18. Yanushekvich, S.N., Butler, J.T., Dueck, G.W., Shmerko, V.P.: Experiments on FPRM expressions for partially symmetric logic functions. In: IEEE International Symposium on Multiple Valued Logic, pp. 141–146 (2000)
19. Lauradoux, C., Videau, M.: Matriochka symmetric Boolean functions. In: IEEE ISIT, pp. 1631–1635 (2008)

20. Keren, O., Levin, I., Stankovic, S.R.: Use of gray decoding for implementation of symmetric functions. In: International Conference on VLSI, pp. 25–30 (2007)
21. Rahaman, H., Das, D.K., Bhattacharya, B.B.: Implementing symmetric functions with hierarchical modules for stuck-at and path-delay fault testability. Journal of Electronic Testing: Theory and Applications 22(2), 125–142 (2006)
22. Perkowski, M., Kerntopf, P., Buller, A., Chrzanowska-Jeske, M., Mishchenko, A., Song, X., Al-Rabadi, A., Jozwiak, L., Coppola, A., Massey, B.: Regular realization of symmetric functions using reversible logic. In: EUROMICRO Symp. on Digital Systems Design, pp. 245–252 (2001)
23. Picton, P.: Modified Fredkin gates in logic design. Microelectronics Journal 25, 437–441 (1994)
24. Moraga, C., Hadjam, F.Z.: On double gates for reversible computing circuits. In: Proc. Intl. Workshop on Boolean Problems (2012)