Constructive Reversible Logic Synthesis for Boolean Functions with Special Properties

Anupam Chattopadhyay[1], Soumajit Majumder[1],
Chander Chandak[2], and Nahian Chowdhury[1]

[1] MPSoC Architectures Research Group, RWTH Aachen University, Germany
anupam.chattopadhyay@umic.rwth-aachen.de
[2] IIT Kharagpur, India

Abstract. Reversible computation is gaining increasing relevance in the context of several post-CMOS technologies, the most prominent of those being quantum computing. The problem of implementing a given Boolean function using a set of elementary reversible logic gates is known as reversible logic synthesis. Though several generic reversible logic synthesis methods have been proposed so far, yet the scalability and implementation efficiency of these methods pose a difficult challenge. Compared to these generic synthesis methods, few reversible logic synthesis approaches for restricted classes of Boolean functions demonstrated better implementation efficiency and scalability. In this paper, we propose a novel constructive reversible logic synthesis technique for Boolean functions with special properties. The proposed techniques are scalable, fast and outperforms state-of-the-art generic reversible synthesis methods in terms of quantum cost, gate count and the number of lines.

1 Introduction

From thermodynamic principles of computing, Landauer [11] pointed out that for every bit of information lost, $kT \cdot \ln 2$ Joules of heat is generated in an irreversible computation, which is recently verified experimentally [3]. Bennett [2] proposed that the computation can be done in reversible manner to achieve theoretically zero power dissipation by building upon Landauer's observations. This concept helped to form the field of reversible computation, which also dictates that the physical reversibility must be accompanied at higher abstraction by logical reversibility. This is a cornerstone for serveral post-CMOS technologies including quantum computing. Reversible logic synthesis accepts an (in)completely specified reversible Boolean function as input and generates a logical representation of the function, where reversible logic gates are used.

Boolean functions serve as prime building block of symmetric-key cryptosystems and error-correcting codes, for which several properties are highly desirable such as nonlinearity, symmetry, correlation immunity and balancedness. The main motivation behind this work is to show that the combinatorial construction of the reversible Boolean functions with specific properties can be done in order to improve the circuit efficiency compared to automated logic synthesis method. In this paper, we focus to two properties namely, *symmetry* and *nonlinearity* of Boolean functions.

S. Yamashita and S. Minato (Eds.): RC 2014, LNCS 8507, pp. 95–110, 2014.

2 Preliminaries

A Boolean function f is of the form $f : \{0,1\}^n \rightarrow \{0,1\}$ (or equivalently $f : \mathbb{V}_2^n \rightarrow \mathbb{V}_2$). The output of the Boolean function f can be represented as a string s of ones and zeros. It can also be represented as a multivariate polynomial over $GF(2)$. This polynomial can be expressed as a exclusive disjunction (EXOR) of a constant a_0 and one or more conjunctions of the function argument. This is called the Exclusive Sum-Of-Product (ESOP) representation. A less general representation of the ESOP form is known as the Algebraic Normal Form (ANF). The general ANF for a function $f(x_1, .., x_n)$ over n-variables can be written as,

$$\begin{aligned} f(x_1, .., x_n) =& a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_i x_i \oplus \cdots \oplus a_n x_n \\ & \oplus \cdots \oplus a_{1,2,\ldots,n} x_1 x_2 \cdots x_n \end{aligned} \tag{1}$$

Reversible and Irreversible Boolean Functions. An n-variable vectorial Boolean function is *reversible* if all its output patterns map uniquely to an input pattern and vice-versa. It can be expressed as an n-input, n-output bijection or alternatively, as a Boolean permutation function over the truth value set $\{0, 1, \ldots 2^{n-1}\}$. An *irreversible* Boolean function $f_{irr} : \{0,1\}^n \rightarrow \{0,1\}^m$ with $n \neq m$ can also be made reversible with the help of extra input lines (ancilla) and/or output lines (garbage lines) such that, $input + ancilla = output + garbage$.

Nonlinearity. The nonlinearity of a Boolean function f on n-variables, denoted by N_f is the minimal Hamming Distance between f and all the affine functions on $\mathbb{V}_2^n$.

The class of Boolean functions having the highest nonlinearity are known as *Bent functions*. They are defined only on $\mathbb{V}_2^{2k}$, i.e. Boolean functions of even number of variables and their nonlinearity is given by $2^{2k-1} - 2^{k-1}$. In contrast, the maximum nonlinearity attainable for a Boolean function $\mathbb{V}_2^{2k+1}$ with odd number of variables still remains an open problem.

Symmetry. A Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is called symmetric if its output is invariant under any permutation of its input bits. Equivalently we can say that the value of $f(x)$ is constant for all x's having the same weight.

Direct Sum. The direct sum of two strings x and y, of lengths n and m respectively, denoted by $x \aleph y$ is given by $x \aleph y = (x \otimes y^c) \oplus (x^c \otimes y)$, where $x \otimes y = (x_0 \ AND \ y) ... (x_{n-1} \ AND \ y)$ denotes the Kronecker product of two strings producing a string of length nm. y^c denotes complement of y.

Reversible Logic Synthesis. Reversible Boolean logic synthesis is achieved with the help of reversible logic gates. The gates are characterized by their implementation cost in quantum technologies, which is dubbed as Quantum Cost (QC). We use the standard QC values from [13] along with the latest improvements reported in [30] for QC computation. Few prominent reversible logic gates are as following.

- **CNOT gate**: CNOT(a, b) = $(a, a \oplus b)$.
- **CCNOT gate (Toffoli gate)**: CCNOT(a, b, c)=$(a, b, ab \oplus c)$. This gate can be generalized with Tof_n gate, where first $n-1$ variables are used as control lines. NOT and CNOT gates are denoted as Tof_1 and Tof_2 respectively.
- **Controlled Swap gate (Fredkin gate)**: Fred(a, b, c) = $(a, \overline{a}b \oplus ac, \overline{a}c \oplus ab)$. This is generalized with $Fred_n$ gate $(n > 1)$, where first $n-2$ variables are used as control lines.
- **Peres gate**: Per(a, b, c) = $(a, a \oplus b, ab \oplus c)$. This gate can be generalized with Per_n gate $(n > 2)$ [30], where first $n-1$ variables are used as control lines.

2.1 Related Work and Motivation

A Boolean function should possess certain properties for its use in cryptographic applications such as symmetry, balancedness and high nonlinearity. Matsui in [17] showed that Boolean functions of low nonlinearity can be approximated and hence can be consequently attacked using linear cryptanalysis attacks, which makes high nonlinearity a desirable property of cryptographically strong Boolean function. To this effect, researchers came up with multiple construction methods for highly nonlinear Boolean functions with large number of variables. These constructive methods could be adopted for reversible circuit construction, which has not been attempted before this work. This constructive approach not only provides a scalable reversible logic synthesis method for highly nonlinear Boolean functions but also, demonstrates increased efficiency of implementation compared to generic reversible logic synthesis techniques. Symmetry of Boolean functions, while a desirable property for cryptographic applications, has also been shown to be important for general reversible logic synthesis [23].

Existing reversible logic synthesis methods can be broadly classified in two categories - generic [26] and property-specific. In the area of property-specific reversible logic synthesis, Beth and Rötteler [4] suggested synthesis approach for linear reversible circuits using Gaussian Elimination and LU-Decomposition to yield circuits with $O(n^2)$ gates. In [21], an improved algorithm with better speed and asymptotically optimal performance for synthesis of linear reversible circuits is proposed. Younes [34] proposed a factorization algorithm for synthesis of homogeneous Boolean functions. For Symmetric Boolean functions, a synthesis technique is proposed at [15]. This is improved further at [8], where a cascade of Peres gates is utilized to obtain reversible circuits with improved QC.

It has been noted at [15] that the constructive reversible logic synthesis procedures for Boolean functions with special properties are scalable and can outperform, in many cases, the generic synthesis techniques. This forms the key motivation of this work. Besides, it has been shown in a recent work that several Quantum algorithms do require efficient reversible circuits for specific classes of Boolean functions [7]. In this work, we make two contributions. First, we propose a constructive reversible logic synthesis technique for highly nonlinear Boolean functions. Second, we propose a constructive reversible logic synthesis technique for symmetric Boolean functions. For both the cases, we report improved results compared to the current literature.

In contrast to the state-of-the-art synthesis techniques [28,26] for Boolean functions with special propertes, we explore deeper and draw from the classical Boolean function

construction techniques from the literature. *The constructive approach presented in this paper have multiple advantages, e.g., scalability, low synthesis runtime and significant implementation efficiency, as we demonstrate via benchmarking with state-of-the-art generic and property-specific reversible synthesis flows.*

3 Synthesis of Highly Nonlinear Functions

In this section, several construction techniques for highly nonlinear Boolean functions and Bent functions are discussed. Those are followed by their reversible circuit synthesis approaches, corresponding theoretical results on the upper bounds of gate count (GC), QC and the total number of lines (L) and comparison with state-of-the-art generic synthesis techniques.

3.1 Construction Method I: [27]

Here, we follow the concatenation-based construction of n-variable, m-resilient Boolean functions. (following Theorem 4,[27]). The idea is to utilize Boolean functions with smaller number of variables to construct a highly nonlinear Boolean function of large variable count. Before the concatenation, *direct sum* function is used. An optimized reversible circuit construction for the direct sum function (denoted as $\aleph$) is developed for the same. We illustrate with the help of a construction of a 14-variable Boolean function with 2nd order resiliency. The values of k and r, which are defined as two parameters for the construction in [27] are chosen as 6. Following the theorem, we concatenate 4 Boolean functions f_i's on $\mathbb{V}_2^{12}$ as $f_i = g_i \aleph \lambda_i$, where g_i is maximum nonlinear function on 6-variables. We choose 4 Bent functions on $\mathbb{V}_2^6$ which are as following.

$$\begin{aligned} g_1 &= x_1x_2 \oplus x_3x_4 \oplus x_5x_6, \\ g_2 &= x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_2, \\ g_3 &= x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_2 \oplus x_3, \\ g_4 &= x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_2 \oplus x_3 \oplus x_4 \end{aligned} \tag{2}$$

The λ_i's belong to $UL_k(m+1)$ where, $UL_k(m+1) = L_k(m+1) \cup \cdots L_k(k)$ (i.e., $L_k(3) \cup \cdots L_k(6)$) The individual sets $L_k(j)$ denote the set of all k-variable linear Boolean functions which are non-degenerate on exactly j-variables. The choices for the λ_i's are,

$$\begin{aligned} \lambda_1 &= L_6(3) = x_7 \oplus x_8 \oplus x_9, \\ \lambda_2 &= L_6(4) = x_7 \oplus x_8 \oplus x_9 \oplus x_{10}, \\ \lambda_3 &= L_6(5) = x_7 \oplus x_8 \oplus x_9 \oplus x_{10} \oplus x_{11}, \\ \lambda_4 &= L_6(6) = x_7 \oplus x_8 \oplus x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12} \end{aligned} \tag{3}$$

For reversible circuit implementation, constructions of g_i and λ_is are straightforward. It is noted that, the reversible logic implementation of *direct sum* is nothing but the $\oplus$ operation of the two functions g_i and λ_i. This allows efficient implementation of

this method also via ESOP-based approach. However, the constructive approach considerably reduced the QC of the individual functions, resulting in overall improvement (Table 1). In the table, the shaded cells represent equal or improved performance in comparison with state-of-the-art synthesis methods.

Comparison with the state-of-the-art synthesis methods: We compare the proposed synthesis technique with state-of-the-art reversible logic synthesis methods. The functions are represented by the choice of the parameters - n, k, r and m. The nonlinearity of the functions are denoted by N_f and the maximum achievable nonlinearity (in the case of Bent functions) by $nlmax$. It can be observed that, with the proper choice of r and k, the method can easily scale to large Boolean functions. On the other hand, the choice of a large k and/or r, requires one to first synthesize a large Boolean function.

Table 1. Benchmarking Construction Method I

Function	$N_f/nlmax$	BDD[32]			ESOP[20,10]			MMD[18]			This work		
		Lines	Gates	QC	Lines	Gates	QC	Lines	Gates	QC	Lines	Gates	QC
$(14, 6, 6, 1)$	7836/8028	26	67	179	15	17	157	14	40	886	18	20	88
$(14, 6, 6, 2)$	7836/8028	31	86	238	15	16	148	14	39	660	18	21	91
$(16, 6, 8, 1)$	31856/32368	31	80	208	17	22	262	16	168	5670	20	23	87
$(16, 8, 6, 2)$	31344/32368	36	93	241	17	16	136	16	172	5850	20	27	91

3.2 Construction Method II: Recursive Construction[22]

A construction method presented in [22] generates large Boolean functions of high nonlinearity and resilience recursively like the previous one. A 10-variable, 4-resilient Boolean function of degree 4 and of nonlinearity 480 is constructed using a 7-variable, 2-resilient Boolean function of degree 4 and nonlinearity 56 as described in Theorem 7 [22]. The 7-variable function is first presented below. Note that this function was again found using a constructive method based on 6-variable functions.

$$f(x_1, \ldots, x_7) = (1 \oplus x_7)(1 \oplus x_6)h_1() \oplus (1 \oplus x_7)x_6h_2() \oplus (1 \oplus x_6)x_7h_3() \oplus x_6x_7h_4(), \tag{4}$$

where

$$\begin{aligned}
h_1(x_1, \ldots, x_5) &= x_1 \oplus x_2 \oplus x_1x_4 \oplus x_3x_4 \oplus x_2x_5 \oplus x_3x_5 \oplus x_4x_5 \oplus x_1x_4x_5 \oplus x_2x_4x_5 \oplus x_3x_4x_5 \\
h_2(x_1, \ldots, x_5) &= 1 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_3x_4 \oplus x_2x_5 \oplus x_3x_5 \oplus x_4x_5 \oplus x_1x_4x_5 \oplus x_2x_4x_5 \oplus x_3x_4x_5 \\
h_3(x_1, \ldots, x_5) &= x_3 \oplus x_1x_3 \oplus x_1x_2x_3 \oplus x_1x_4 \oplus x_2x_4 \oplus x_1x_2x_4 \oplus x_3x_4 \oplus x_1x_3x_4 \oplus x_5 \oplus x_1x_5 \\
&\quad \oplus x_1x_2x_5 \oplus x_1x_3x_5 \\
h_4(x_1, \ldots, x_5) &= 1 \oplus x_2 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_4 \oplus x_1x_4 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_1x_5 \oplus x_2x_5 \oplus x_1x_2x_5 \\
&\quad \oplus x_3x_5 \oplus x_1x_3x_5
\end{aligned} \tag{5}$$

In [22], an n-variable Boolean function F_d is defined to be in *desired form* if it follows the construction

$$F_d = (1 \oplus x_n)f_1 \oplus x_nf_2, \tag{6}$$

where f_1 and f_2 are $(n-1)$-variable functions with their degree being 1 less than F_d. The aforementioned 7-variable function is in *desired form* since, those are constructed recursively using 6-variable functions. Based on the 7-variable function $f(x_1, \ldots, x_7)$, a 10-variable, 4-resilient Boolean function of degree 4 and of nonlinearity 480 is constructed as following. Let $F = x_{n+2} \oplus x_{n+1} \oplus f$ and $G = (1 \oplus x_{n+2} \oplus x_{n+1})f_1 \oplus (x_{n+2} \oplus x_{n+1})f_2 \oplus x_{n+2} \oplus x_n$. Then the target 10-variable function F_1 of specified properties is constructed as $F_1 = (1 \oplus x_{n+3})F \oplus x_{n+3}G$. The construction of the 10-variable function from the 7-variable function can be easily achieved with Toffoli and Fredkin gates, as shown in the Figure 1. The detailed implementation of the constituent functions h_1, h_2, h_3 and h_4 are not shown. These functions were synthesized using an ESOP-based flow including common cube sharing.

The proposed synthesis technique is compared with the existing state-of-the-art synthesis techniques in Table 2. It is interesting to note that, even though the recursive construction allows direct implementation via simple reversible gates, none of the synthesis methods had matching QC or gate count that could be achieved from our constructive technique. The additional 6 lines, compared to the MMD method is contributed due to the fact that the constituent functions were synthesized using ESOP, thereby requiring 4 lines for the constituent functions h_1, h_2, h_3 and h_4. Furthermore, 2 were required for 6-variable functions f_1 and f_2, which were used for constructing $f(x_1, \ldots, x_7)$ as well as for constructing F_1. The improvement in constructive method is due to the following facts

- The *desired form* of a function directly translates to controlled swap gate
- Recursive construction method is based on basic CNOT gates

Evidently, these properties are not utilized by generic synthesis methods.

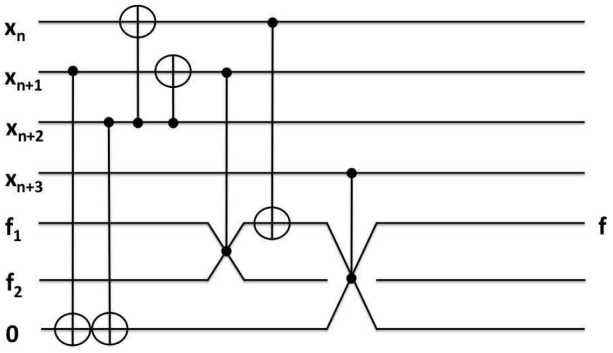

Fig. 1. Construction Method II:10-variable, 4-resilient Boolean function

Table 2. Comparison of Construction Method II with existing synthesis methods

Synthesis Method	Lines	Gates	QC
BDD[32]	42	147	447
ESOP[20,10]	26	89	309
MMD[18]	10	654	55632
This work	16	44	230

3.3 Construction Method III: [9]

An early construction method for Bent functions on $\mathbb{V}_2^{2k}$ was originally proposed in [9]. This construction, known as Maiorana-McFarland construction, has been further generalized in [5]. Numerous developments on Bent function construction followed the basic Maiorana-McFarland technique, such as in [12], authors obtained Bent functions with high resiliency. In this work, we focus on the basic construction as outlined in [9] and [35]. The idea of the construction is to concatenate the linear functions on $\mathbb{V}_2^k$, thereby generating Bent functions on $\mathbb{V}_2^{2k}$.

$$f(y,x) = \pi(y) \cdot x \oplus g(y), x, y \in \mathbb{V}_2^k \tag{7}$$

where f is the resultant Bent function, π represents a permutation on $\mathbb{V}_2^k$ and g is any Boolean function on $\mathbb{V}_2^k$. There are $2^{2^k}(2^k!)$ such Bent functions, where the possible permutations are covered by the factor $(2^k!)$. For our study, we restricted π to all possible linear functions, and assumed g to be 0, thereby generating $2^k(2^k!)$ Bent functions. The general implementation is as shown in the Fig. 2.

Mapping to Reversible Circuits and Implementation Cost Determination: The aforementioned construction can be realized by applying psuedo-optimal linear reversible circuit synthesis [21] followed by a set of Fredkin gates. This construction has high L, GC and QC due to the multiplexer type functionality where k lines act as control lines and select one from all possible 2^k linear functions on $\mathbb{V}_2^k$. This construction method suffers from scalability issues since, with increasing variable count, the number of linear functions increases exponentially.

Comparison with state-of-the-art synthesis methods: The benchmarking results are presented in Table 3. Note that due to the size constraint, the complete functions are not presented for the studied 8-variable Boolean functions. Instead, only the permutation of the 16 linear functions for the sub-space $\mathbb{V}_2^4$ are given, where $1 \rightarrow 0, 2 \rightarrow x_1, 3 \rightarrow x_2, \cdots, 16 \rightarrow x_1 \oplus x_2 \oplus x_3 \oplus x_4$. The total number of lines required remain upper-bounded by 20, which is due to 4 control inputs, 4 inputs for the linear functions, which are reused as part of the total 16 linear functions. Thereby, the upper bound of lines can be generalized as $k + 2^k$ for a bent function construction on $\mathbb{V}_{2k}^k$. In this case, however, the line counts could be further reduced by applying algebraic optimization based on the ESOP formulation (see subsection 3.5). In the same manner, it is possible to determine the generalized costs for the linear function generator part. However, the identification of minimum swap count for a given permutation is non-trivial. Thankfully, the construction method as shown in [35] includes the swaps. Except for the count of lines, the constructive method outperformed ESOP and BDD-based methods both in gate count and QC for most of the permutations. We did not benchmark against MMD as it typically reports even higher gate count and QC compared to ESOP and BDD-based methods.

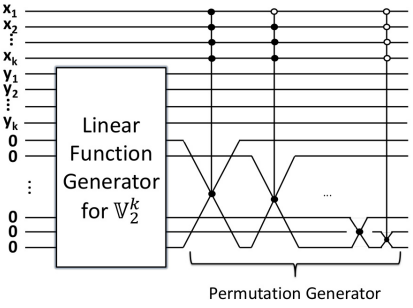

Fig. 2. Construction Method III: Bent Function

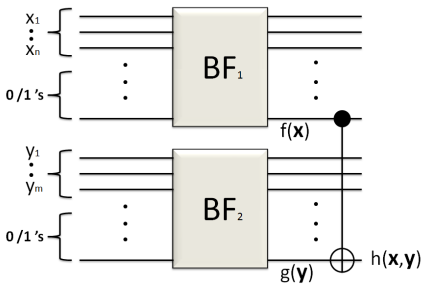

Fig. 3. Construction Method IV: Bent Function

Table 3. Benchmarking Construction Method III

Permutation of Linear Functions	Variable	BDD[32]			ESOP[20,10]			This work		
		Lines	Gates	QC	Lines	Gates	QC	Lines	Gates	QC
$\{11, 15, 10, 16, 4, 5, 2, 13, 8, 14, 6, 12, 9, 3, 7, 1\}$	8	22	71	219	9	32	488	18	33	97
$\{7, 14, 5, 13, 1, 16, 6, 3, 10, 12, 9, 2, 4, 11, 8, 15\}$	8	22	68	196	9	35	479	14	30	90
$\{15, 9, 14, 7, 5, 1, 16, 12, 6, 2, 11, 10, 4, 8, 3, 13\}$	8	22	62	174	9	25	353	13	23	71
$\{6, 14, 9, 4, 7, 13, 11, 1, 8, 15, 12, 5, 10, 3, 16, 2\}$	8	22	70	202	9	33	509	19	34	86
$\{11, 13, 16, 15, 8, 7, 2, 14, 10, 5, 4, 9, 3, 12, 1, 6\}$	8	20	62	174	9	36	516	16	35	99
$\{4, 7, 10, 9, 15, 16, 3, 8, 12, 11, 13, 6, 1, 5, 14, 2\}$	8	24	65	169	9	29	461	15	31	87
$\{9, 4, 2, 15, 6, 10, 7, 11, 12, 3, 16, 14, 5, 1, 13, 8\}$	8	22	72	208	9	34	506	15	29	81
$\{16, 9, 10, 7, 15, 4, 5, 12, 2, 1, 6, 13, 11, 8, 14, 3\}$	8	23	68	200	9	33	461	15	30	86
$\{16, 4, 15, 8, 9, 1, 14, 5, 10, 13, 2, 3, 12, 6, 7, 11\}$	8	22	73	201	9	34	490	18	34	98
$\{2, 7, 5, 6, 12, 3, 1, 4, 16, 10, 13, 11, 9, 14, 8, 15\}$	8	22	68	196	9	37	597	16	30	86

3.4 Construction Method IV: [33]

The authors in [33] proposed two theorems for construction of new Bent functions using existing Bent functions. This is particularly interesting for Boolean functions with odd number of variables, where direct constructions methods cannot be used [25].

Theorem 1: Let f and g be Boolean functions on $\mathbb{V}_2^m$ and $\mathbb{V}_2^n$ respectively. Then the Boolean function $h : \mathbb{V}_2^{m+n} \rightarrow \mathbb{V}_2$ defined by $h(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) \oplus g(\mathbf{y})$ is bent iff f and g are bent.

Theorem 2: If f is a Bent function on $\mathbb{V}_2^n$, then $f \oplus l$ is a Bent function for any affine function l on $\mathbb{V}_2^n$.

Mapping to Reversible Circuits and Implementation Cost Determination: The basic idea of this construction according to theorem 1 is given by Fig. 3. The second construction method can be achieved similarly by performing a CNOT operation between the Bent function and the linear function. The QC of the resulting circuit using this method of construction is simply the sum of QCs of the constituent functions added

with 1, which is due to the CNOT gate. The total number of lines for the resulting circuit is $L(f_1) + L(f_2) + m + n$, where f_1 denotes a Bent function on n-variables and f_2 is a Bent function on m-variables for Theorem 1. Here, an ESOP-based implementation of the constituent Bent functions is assumed. f_2 is a linear function on n-variables for Theorem 2 (hence, $m = 0$ for the second construction).

The QC, GC and L for this construction are enlisted in Table 4.

Table 4. Implementation Costs for Construction Method IV

Gate Count	$GC(f_1) + GC(f_2) + 1$
Quantum Cost	$QC(f_1) + QC(f_2) + 1$
Lines	$L(f_1) + L(f_2) + m + n$

Comparison with state-of-the-art synthesis methods: This simple construction of Bent function is compared with BDD-based and ESOP-based methods, when the final Boolean function is subjected to synthesis. We observed an improved performance in most of the cases. The constructive method is scalable to large number of variables, in contrast to the generic methods. An 1-hour timeout set to the benchmarked synthesis methods failed to return a valid circuit in one case (indicated by '-').

3.5 Post-synthesis Optimization

Aforementioned construction techniques show strong algebraic structure and hence there is a wide scope for optimizing the synthesis by using *common cube sharing*. Common cube sharing is a well-studied problem in classical logic synthesis as it helps in minimization of cost and size by identifying the sub-circuits which form the basis for larger functional blocks. This optimization is applied on the Boolean functions obtained following the construction techniques. The implementation costs of these functions, as presented in the following Table 5, is computed after application of the cube sharing algorithm [20]. Note that, such optimizations are present also for the ESOP-based synthesis flows that we compared against and hence, do not provide any undue advantage to the proposed constructive synthesis flow.

Table 5. Benchmarking Construction Method IV

Function		Variable	BDD[32]			ESOP[20,10]			This work		
			Lines	Gates	QC	Lines	Gates	QC	Lines	Gates	QC
$f_{\mathbf{x}}$	$= x_1x_2$	2	3	1	5	3	1	5	3	1	5
$g_{\mathbf{y}}$	$= y_1y_2 \oplus y_3y_4 \oplus y_5y_6$	6	9	11	31	7	3	15	7	3	15
$h_{\mathbf{x},\mathbf{y}}$	$= f_{\mathbf{x}} \oplus g_{\mathbf{y}}$	8	12	17	45	9	4	20	10	5	21
$f_{\mathbf{x}}$	$= x_1x_2 \oplus x_3x_4$	4	6	7	19	5	2	10	5	2	10
$g_{\mathbf{y}}$	$= y_1y_2 \oplus y_3y_4 \oplus y_5y_6 \oplus y_7y_8 \oplus y_9y_{10} \oplus y_{11}y_{12}$	12	19	30	78	13	12	88	13	6	30
$h_{\mathbf{x},\mathbf{y}}$	$= f_{\mathbf{x}} \oplus g_{\mathbf{y}}$	16	22	40	92	17	24	112	18	9	41
$f_{\mathbf{x}}$	$= x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_7x_8 \oplus x_9x_{10} \oplus x_{11}x_{12} \oplus x_{13}x_{14} \oplus x_{15}x_{16}$	16	22	40	92	17	26	214	17	12	44
$g_{\mathbf{y}}$	$= y_1y_2 \oplus y_3y_4 \oplus y_5y_6 \oplus y_1 \oplus y_2 \oplus y_6$	6	9	12	28	7	6	18	7	6	18
$h_{\mathbf{x},\mathbf{y}}$	$= f_{\mathbf{x}} \oplus g_{\mathbf{y}}$	22	-	-	-	-	-	-	24	19	53

4 Synthesis of Symmetric Functions

In contrast to nonlinear Boolean functions, constructive approach for synthesizing symmetric Boolean functions have been studied in the past [15,8], possibly due to their usage in efficient synthesis of general reversible Boolean functions [23]. Before proceeding further, we present some recent results on generalized Peres gates as well as show how cascaded multi-control Peres gates can be realized with lower Quantum costs.

4.1 Quantum Cost of Cascaded, Generalized Peres Gates

Generalization of Peres gates is introduced in [30] with the following definition,

$$Per_n(x_0, x_1, \cdots, x_n) = (x_0, x_0 \oplus x_1, x_0x_1 \oplus x_2, \cdots, x_0x_1 \cdots x_{n-1} \oplus x_n),\ n \geq 2 \tag{8}$$

Such gates are implemented in an optimized manner with controlled k^{th}-root-of-NOT gates. In the following Fig. 4, Per_3 and its corresponding reversible circuit realization is shown. Here, controlled-V, controlled-V^+ represents controlled k^{th}-root-of-NOT for $k = 2$ and controlled-W, controlled-W^+ represents the same for $k = 4$. In the lower part of the same Fig. 4 an alternative reversible circuit is presented, which locally re-orders the controlled k^{th}-root-of-NOT gates. Since the control lines are exclusive, the resultant Boolean function remains the same.

In lemma 1 of [30], the number of elementary gates for Per_n is proved to be n^2, which is same as its QC since the elementary gates have a QC of 1. We explore this further considering cascaded, generalized Peres gates. The re-ordering of controlled

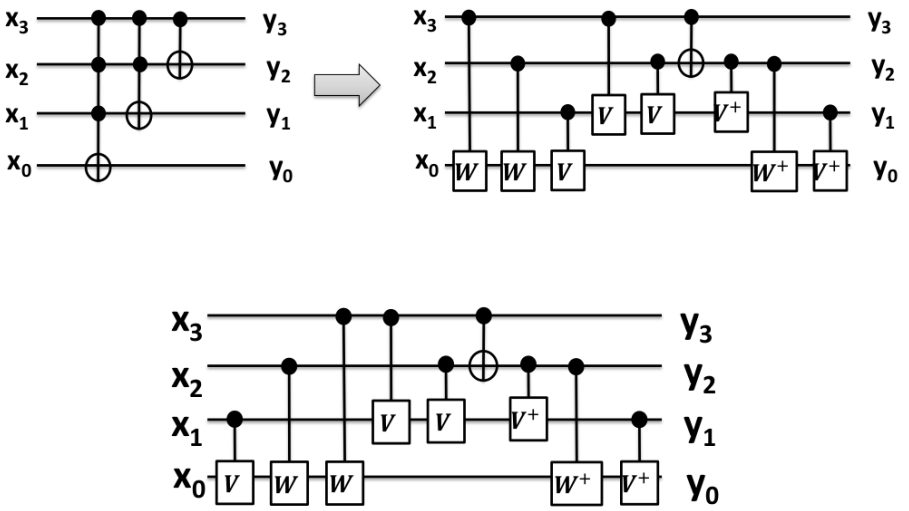

Fig. 4. Generalized Peres Gate Implementation

k^{th}-root-of-NOT gates provides an opportunity to reduce a few adjacent gates. This has been explored in the context of basic Peres gate [19] and for multi-control Toffoli gates [29].

Lemma 1. *Cascading* 2 Per_n *gates require* $n^2 + n$ *gates.*

Proof. From lemma 1 of [30], 2 cascaded Per_n gates require $2n^2$ elementary gates. An n-controlled Peres gate, i.e., Per_n consists of three parts. The part in middle implements a Per_{n-1} gate. The first and the last part of the circuit consists of n and $n-1$ controlled Quantum gates respectively. For two adjacent Per_n gates, by re-ordering, $(n-1)$ controlled Quantum gates can be cancelled out with their corresponding inverses. This leads to a reduction of $2(n-1)$ elementary gates. Every Per_n contains a Per_{n-1} inside it, which, in the same manner allows a reduction of $2(n-2)$ gates. This continues till there is a Per_2 gate, for which a reduction of $2(n-(n-1))$ gates are possible. Hence, by summing the reductions, we obtain the elementary gate count as following.
$2n^2 - \sum_{i=1}^{n-1} 2(n-i)$
$= 2n^2 - n(n-1)$
$= n^2 + n$ □

Corollary 1. *Cascading* t Per_n *gates require* $n^2 + (t-1)n$ *gates.*

Proof. For each pair of Per_n gates, a reduction of $n(n-1)$ is obtained. For t number of cascaded Per_n gates, the total reduction from a basic tn^2 gate count is $(t-1)n(n-1)$. Hence, the final gate count is,
$tn^2 - (t-1)n(n-1)$
$= tn^2 - (t-1)n^2 + (t-1)n$
$= n^2 + (t-1)n$ □

Evidently, the above results also lead to the QC values as we are only considering gates with 1 QC. An application of the above lemma is shown graphically for 2 cascaded Per_3 gates in Fig. 5. It is clear that by cascading as much as possible Peres gates with same control lines, one can obtain a significant reduction in gate count and QC. In the constructive synthesis of symmetric Boolean functions, this property is exploited.

4.2 Constructive Synthesis for Symmetric Functions

Since the symmetric Boolean functions have unique output for a given Hamming weight of the input n-variable Boolean vector $\mathbb{V}_2^n$, we propose an approach based on two phases. First, the Hamming weight computation in $(\lfloor \log_2 n \rfloor + 1)$ lines followed by an evaluation of the function on those lines. Note that, due to the sharing of one target Hamming weight line with one input line, $(\lfloor \log_2 n \rfloor)$ ancilla lines are needed. The circuit complexity largely depends on the Hamming weight computation, which is done using a ripple-carry adder approach as shown in Fig. 6. This approach is earlier used for Hidden Weighted Bit (HWB) functions in [13]. It can be observed that the Hamming weight computation circuitry is nothing but a series of cascaded, generalized Peres gates. In case of $rd73$, the Hamming weight computation is done with 2 cascaded Per_2

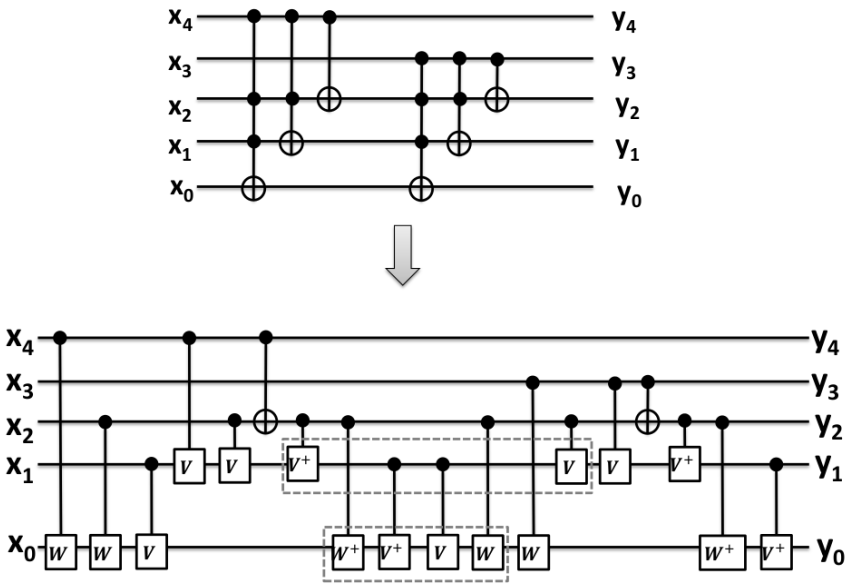

Fig. 5. Cascaded Generalized Peres Gate Implementation

gates and 4 cascaded Per_3 gates, resulting into a total QC of 24. Please note, that to reduce garbage count, an input line is re-used for storing the LSB of the Hamming weight value. The Hamming weight values directly result into the output. Hence, no further gate is required. In providing the gate count, we report the number of mixed-control Tof_n gates for ease of comparison with previous results.

Table 6. Benchmarking with Property-specific Synthesis Techniques

Function	I/O	[8] (Low Garbage)		[8] (Low QC)		[15]			This work		
		Garbage	QC	Garbage	QC	Garbage	Gates	QC	Garbage	Gates	QC
$2of5$	5/1	-	-	-	-	6	12	32	6	9	15
$rd53$	5/3	5	28	6	20	5	12	36	4*	10	18
$rd73$	7/3	7	46	10	32	7	20	64	6*	16	24
$rd84$	8/4	9	66	13	44	11	28	98	7	20	27
$6sym$	6/1	-	-	-	-	9	20	62	8	16	32
$9sym$	9/1	14	88	19	59	11	28	94	10	22	30

The computation of Hamming weight is further optimized by taking the desired output into consideration. For example, the benchmark function $2of5$ produces an output of 1 when the Hamming weight is 2 or 010. However, 110 is not a valid Hamming weight for 5-variable circuit. Therefore, the Hamming weight computation can optimize the carry propagation circuitry and an additional line used for the most significant

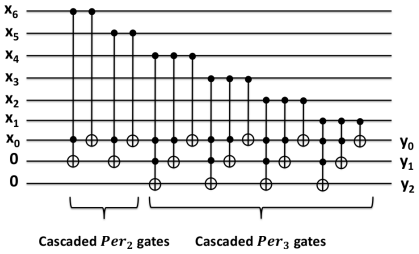

Fig. 6. Generation of Hamming Weight for $rd73$

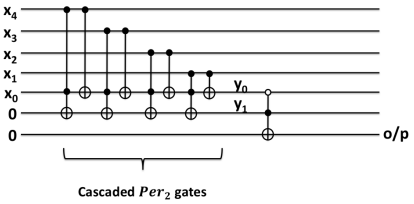

Fig. 7. Reversible Ciruit: $2of5$

bit. Similarly for $rd84$, the Per_4 gate can be avoided by putting an inverted-control Toffoli gate at the end, which sets the output line indicating Hamming weight of 1000 to true if none of the less Hamming weight values are true. This allows cascading of 5 Per_3 gates. For $9sym$, computing the final two Hamming weights 1000 and 1001 can be avoided, since those do not influence the output. Neither of the Hamming weight values influencing the output, i.e. 3, 4, 5 and 6, has any overlapping bit-pattern in the 3 least significant bits. The complete reversible circuit for $2of5$ is shown in Fig. 7.

Comparison with state-of-the-art synthesis methods: This construction technique easily outperforms both the previous property-specific synthesis flows for symmetric functions in *all the efficiency metrics* as shown in Table 6. In [8], a cascade of 2-control Peres gates are used followed by an extraction-elimination module. In contrast, we do not require any follow-up module and obtain the individual Hamming weight values directly. The values with * in the garbage count are shown to be minimal [13]. Further, it is likely that any approach based on adder circuit and generalized Peres gates may benefit from the results presented in this paper.

4.3 Upper Bounds of Symmetric Functions

In this subsection, we establish novel upper bounds for the gate count and QC for symmetric Boolean functions based on the proposed constructive approach. To the best of our knowledge, no such upper bound for symmetric Boolean functions exist.

Based on a recent result [30], we have an expression of the QC of Tof_n gate as following.

$$2(n-1)^2 - 2(n-1) + 1 \tag{9}$$

While this result is for positive-control Tof_n gates, we use it also for mixed-polarity Tof_n gates. It has been showed in the case of mixed-polarity Tof_n gates, a realization with equivalent QC can be obtained [16]. The only case, where the QC for mixed-polarity is higher, is when all the control inputs are negated. In that case, the QC is

$$2(n-1)^2 - 2(n-1) + 3 \tag{10}$$

Using the proposed constructive approach, we explore the following upper bound calculation based on the circuit for Hamming weight computation using cascaded Per_n gates.

Lemma 2. *The upper bound of QC for an n-variable Hamming weight computation circuit is* $(\lfloor \log_2 n \rfloor)^2 + n(\lfloor \log_2 n \rfloor) + n - 1$.

Proof. For an n-variable Boolean function, the Hamming weight computation requires $(n-1)$ cascaded Per_k gates, where the maximum value of k can be $\lfloor \log_2 n \rfloor + 1$. Considering the worst-case scenario, total $(n-1)$ cascaded $Per_{\lfloor \log_2 n \rfloor + 1}$ gates are needed. By using the result from Corollary 1, the upper bound on QC is $(\lfloor \log_2 n \rfloor + 1)^2 + (n-2)(\lfloor \log_2 n \rfloor + 1)$. Simplification of this leads to the result. □

Theorem 1. *The upper bound of QC for an n-variable Symmetric Boolean function is* $(2n+1)(\lfloor \log_2 n \rfloor)^2 - n(\lfloor \log_2 n \rfloor) + 4n - 1$.

Proof. The computation of Symmetric function is composed of Hamming weight calculation followed by a set of comparators. Each comparator is due to one Hamming weight value. There can be $(n+1)$ different Hamming weights for an n-variable Boolean function. However, at most n Hamming weights can contribute to the generation of the Symmetric function, as otherwise, it will become a constant function. The Hamming weights are stored in $(\lfloor \log_2 n \rfloor + 1)$ lines. Each comparator for a specific Hamming weight value require a mixed-polarity Tof_k gate, where k is at most $(\lfloor \log_2 n \rfloor + 1)$. This leads to the worst-case QC value from the comparator circuit as $n(2(\lfloor \log_2 n \rfloor)^2 - 2(\lfloor \log_2 n \rfloor) + 3)$. By adding the upper bound of QC from the Hamming weight circuit with the comparator circuit, we obtain the result. □

Clearly, the upper bound derived in Theorem 1 based on the constructive approach is tighter compared to the generic upper bounds presented recently in [1].

It can be also noted that the QC values obtained for benchmark circuits, presented in Table 6 is significantly less than the upper bound presented in Theorem 1. To predict a tighter bound of QC for individual circuits compared to theorem 1, we need to enumerate the number of Per_1, $Per_2 \cdots Per_{\lfloor \log_2 n \rfloor + 1}$ gates, which follows a piecewise function. Let us define the number of Per_k gates for an n-variable Hamming weight computation circuit as $C_n(Per_k)$. It can be easily shown that,

$$C_n(Per_k) = \begin{cases} 0 & \text{if } n < 2^{k-1} \\ (n+1) - 2^{\lfloor \log_2 n \rfloor} & \text{if } 2^{k-1} \leq n \leq 2^k - 1 \\ 2^{k-1} & \text{if } n \geq 2^k \end{cases}$$

5 Summary and Future Work

In this paper, a novel, constuctive reversible logic synthesis method is presented for Boolean functions with special properties. It has been shown that this synthesis methods outperforms state-of-the-art, general reversible logic synthesis methods. Detailed experimental studies are presented to support the claim.

Due to the desirability of Boolean functions with specific properties, new constructions are continuously being proposed. The presented techniques can be extended to cover further Boolean function construction methods. The interdependence between the Boolean function properties and the implementation efficiency is an interesting open research problem. Further, we will like to explore the usage of efficient reversible circuits for symmetric Boolean functions in the context of reversible logic synthesis and adder circuit realizations.

Acknowledgement. The authors will like to thank the anonymous reviewers, whose critical feedback helped to improve the paper considerably. The first author will like to acknowledge the help of Prof. Subhamoy Maitra, Indian Statistical Institute, Kolkata, India for understanding several Boolean function construction techniques.

References

1. Abdessaied, N., Soeken, M., Thomsen, M.K., Drechsler, R.: Upper bounds for reversible circuits based on Young subgroups. Information Processing Letters 114(6), 282–286 (2014) ISSN 0020-0190, http://dx.doi.org/10.1016/j.ipl.2014.01.003
2. Bennett, C.H.: Logical Reversibility of Computation. IBM Journal of Research and Development 6, 525–532 (1973)
3. Bérut, A., Arakelyan, A., Petrosyan, A., Ciliberto, S., Dillenschneider, R., Lutz, E.: Experimental verification of Landauer's principle linking information and thermodynamics. Nature, 187–189 (March 2012)
4. Beth, T., Rötteler, M.: Quantum algorithms: Applicable Algebra and Quantum physics. In: Quantum Information, pp. 96–150. Springer (2001)
5. Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes. In: Crama, Y., Hammer, P. (eds.) Boolean Methods and Models, pp. 257–397. Cambridge University Press (2010), http://www.math.univ-paris13.fr/~carlet/pubs.html
6. Chee, S., Lee, S., Lee, D., Sung, S.H.: On the Correlation Immune Functions and Their Nonlinearity. In: Kim, K.-C., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 232–243. Springer, Heidelberg (1996)
7. Chakrabory, K., Maitra, S.: Quantum algorithm to check Resiliency of a Boolean function. In: International Workshop on Coding and Cryptography (2013)
8. Deb, A., Das, D.K., Rahaman, H., Bhattacharya, B.B., Wille, R., Drechsler, R.: Reversible Circuit Synthesis of Symmetric Functions Using a Simple Regular Structure. In: Workshop on Reversible Computation, pp. 182–195 (2013)
9. Dillon, J.F.: Elementary Hadamard Difference Set, PhD Dissertation, University of Maryland, College Park, MD (1974)
10. Gupta, P., Agrawal, A., Jha, N.K.: An Algorithm for Synthesis of Reversible Logic Circuits. IEEE TCAD 25(11), 2317–2330 (2006)
11. Landauer, R.: Irreversibility and heat generation in the computing process. IBM Journal of Research and Development 5, 183–191 (1961)
12. Maitra, S., Pasalic, E.: A Maiorana–McFarland type Construction for Resilient Boolean functions on n variables (n even) with nonlinearity. Discrete Applied Mathematics 154(2), 357–369 (2006)
13. Maslov, D.: Reversible Benchmarks (2014), http://webhome.cs.uvic.ca/~dmaslov (last accessed March 2014)
14. Maslov, D., Mathew, J., Cheung, D., Pradhan, D.K.: An O(m^2)-depth quantum algorithm for the elliptic curve discrete logarithm problem over GF(2^m)^a. In: Quantum Information & Computation, pp. 610–621 (2009)

15. Maslov, D.: Efficient reversible and quantum implementations of symmetric Boolean functions. IEE Proceedings of Circuits, Devices and Systems 153(5), 467–472 (2006)
16. Maslov, D., Dueck, G.W., Miller, D.M., Negrevergne, C.: Quantum Circuit Simplification and Level Compaction. IEEE TCAD 27(3), 436–444 (2008), doi:10.1109/TCAD.2007.911334
17. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993)
18. Miller, D.M., Maslov, D., Dueck, G.W.: A Transformation Based Algorithm for Reversible Logic Synthesis. In: Proceedings of DAC, pp. 318–323 (2003)
19. Moraga, C., Hadjam, F.Z.: On Double gates for Reversible Computing Circuits. In: Proceedings of International Workshop on Boolean Problems (2012)
20. Nayeem, N.M., Rice, J.E.: Improved ESOP-based Synthesis of Reversible Logic. In: Proceedings of the Reed-Muller Workshop (2011)
21. Patel, K.N., Markov, I.L., Hayes, J.P.: Optimal synthesis of linear reversible circuits. Quantum Information & Computation 8(3), 282–294 (2008)
22. Pasalic, E., Maitra, S., Johansson, T., Sarkar, P.: New constructions of resilient and correlation immune Boolean functions achieving upper bound on nonlinearity. Electronic Notes in Discrete Mathematics 6, 158–167 (2001)
23. Perkowski, M., Kerntopf, P., Buller, A., Chrzanowska-Jeske, M., Mischenko, 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: Proceedings of IWLS, pp. 90–95 (2001)
24. Pieprzyk, J., Finkelstein, G.: Towards Effective Nonlinear Cryptosystem Design. In: Proceedings of IEEE Computers and Digital Techniques, vol. 135(6), pp. 143–7062 (November 1988) ISSN:0143-7062
25. Preneel, B., Van Leekwijck, W., Van Linden, L., Govaerts, R., Vandewalle, J.: Propagation characteristics of Boolean functions. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 161–173. Springer, Heidelberg (1991)
26. Saeedi, M., Markov, I.L.: Synthesis and Optimization of Reversible Circuits - A Survey. CoRR abs/1110.2574 (2011), http://arxiv.org/abs/1110.2574
27. Sarkar, P., Maitra, S.: Construction of nonlinear Boolean functions with important cryptographic properties. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 485–506. Springer, Heidelberg (2000)
28. Soeken, M., Frehse, S., Wille, R., Drechsler, R.: RevKit: A toolkit for reversible circuit design. In: Workshop on Reversible Computation, pp. 69–72 (2010)
29. Szyprowski, M., Kerntopf, P.: Reducing Quantum Cost of Pairs of Multi-Control Toffoli Gates. In: International Workshop on Boolean Problems (2012)
30. Szyprowski, M., Kerntopf, P.: Low Quantum Cost Realization of Generalized Peres and Toffoli Gates with Multiple-Control Signals. In: 13th IEEE International Conference on Nanotechnology, pp. 802–807 (2013)
31. Tarannikov, Y.V.: New Constructions of Resilient Boolean Functions with Maximal Nonlinearity. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, p. 66. Springer, Heidelberg (2002)
32. Wille, R., Drechsler, R.: BDD-based Synthesis of Reversible Logic for Large Functions. In: Proceedings of DAC, pp. 270–275 (2009)
33. Yarlagadda, R., Hershey, J.E.: Analysis and synthesis of bent sequences. IEEE Proceedings on Computers and Digital Techniques 136(2), 112–123 (1989)
34. Younes, A.: Synthesis and Optimization of Reversible Circuits for Homogeneous Boolean Functions. arXiv:0710.0664 [quant-ph] (2007)
35. Zhang, F., Hu, Y., Ma, H., Xie, M.: Constructions of Maiorana-McFarland's Bent Functions of Prescribed Degree. In: International Conference on Computational Intelligence and Security (CIS), pp. 315–319 (2010)