Efficient Reversible and Quantum Implementations of Symmetric Boolean Functions

Dmitri Maslov

Abstract

It is a well known fact in logic design that synthesis of some special class of Boolean functions is often easier than the synthesis of a general unrestricted specification. In reversible logic, well-scaled synthesis methods with a reasonably small cost of the associated implementation have been found for only a few classes of functions. This includes synthesis of multiple-output symmetric and reversible linear functions.

In this work, we present an efficient reversible/quantum synthesis method for the class of multiple-output symmetric functions. Our method is purely theoretical, therefore its scaling on functions with a large number of inputs/outputs requires minimal resources. We calculate garbage, *i.e.* the number of outputs that are not required by the function specification, the number of reversible gates, and the quantum cost of the presented implementations. We then apply our approach to the synthesis of benchmark functions. Comparison of our designs to the previously reported implementations is favorable.

1 Introduction

Reversible logic implementations are such that the values of input variables can be deduced from the output values. Information loss does not contribute to heat dissipation in these circuits [4, 13]. Therefore, they potentially help to solve at least two problems: overheating and power saving, which implies longer life for batteries.

The reversible logic solution may be especially important in low-voltage designs of mobile systems, where both power saving and overheating are very important due to the need for light weight and independent power supply.

Reversible implementations have applications in quantum computing [12, 25] and nanotechnology [19, 20]. Quantum technology has received significantly more attention, and is often considered to be the most promising application for reversible computations. Consequently, in this paper, we calculate quantum costs for the presented designs. Numerous applications requiring reversible implementations have resulted in the appearance of multiple reversible synthesis papers, *e.g.* [2, 9, 10, 11, 15, 17, 21, 24, 26, 28, 30, 32]. Indeed, once a technology is discovered, the next step towards employing it is the creation of useful applications and the synthesis of the corresponding circuits.

The use of general purpose reversible synthesis methods [2, 9, 10, 11, 15, 17, 21, 24, 30, 32] usually results in large and thus, likely, technologically expensive specifications. Those methods, especially their heuristic parts, scale poorly. In fact, the largest benchmark function for which a heuristically synthesized circuit has been reported has only 20 inputs and 20 outputs [17]. Most of the above methods, except [15, 24, 32], target synthesis of the reversible specifications only. This limits their applicability to the synthesis of useful benchmark functions since the latter are usually specified irreversibly. The task of finding a reversible specification containing a given irreversible specification that can be effectively used by one or the other synthesis approach is difficult to solve and no reasonably good solution of this problem has yet been found. Synthesis approaches that work with irreversible specifications have been proposed [15, 24, 32]. The synthesis method suggested in [32] has neither been implemented nor tested. The synthesis approach in [15] works with functions having up to 10 input variables only. Usage of methods discussed in [24] allows synthesis of larger irreversible specifications. However, its application sometimes results in a significant number of garbage bits, wide and expensive gates (resulting in high overall technological cost), and it fails to employ the interdependence of bits in a reversible circuit.

Due to the above problems with the synthesis of a general non-resticted specification, it is a good idea to synthesize reversible circuits for classes of functions. Linear reversible functions with n input/output variables were synthesized with $O(n^2/\log(n))$ reversible/quantum CNOT operations [26]. This synthesis requires no garbage and is asymptotically optimal.

Symmetric functions were first synthesized as a separate class in [28]. The reversible gate count for an n-input m-output symmetric function synthesis in this method is $\frac{n^2}{2} + mn$. But, their implementation uses excessive garbage, $\frac{n(n+1)}{2}$. This will likely prevent their synthesis results from being used in quantum technology—most advanced of the existing quantum technologies is liquid NMR [1, 8] which imposes a strict limit on the number of qubits allowed in a single computation. Since, minimization of garbage is an important synthesis criterion in such application [8]. We also notice that an inexpensive quantum realization of the Kerntopf gates used in [28] for synthesis was never found. In particular, it was recently shown [22] that optimal quantum implementation of the Kerntopf gate requires 14 elementary operations in a well studied [14] quantum gate library composed with NOT, CNOT, and controlled-*sqrt*-of-NOT gates. Optimal NCV quantum implementation of the Toffoli gate used in this work requires only 5 such operations.

[28] mentions an approach to a non-symmetric Boolean specification extension into a larger but symmetric specification. This is very useful because it makes it possible to synthesize any function by first "symmetrizing" it through adding new input variables, and then synthesizing its extended symmetric specification. Thus, a good reversible synthesis procedure for a symmetric specification may be of interest in general reversible synthesis. This further motivates research of the ways to construct

inexpensive circuits for symmetric functions.

In this paper we present a synthesis method with the reversible gate count of at most $\frac{n^2}{2} + mn + o(n^2 + mn)$, quantum implementation cost of at most $\frac{5n^2}{2} + mn + o(n^2 + mn)$ and at most $2n - 2$ bits of garbage; and its modification with the reversible gate count of at most $\frac{(2n-k)(k-1)}{2} + mn + o(n^2 + mn)$, quantum cost of at most $\frac{5(2n-k)(k-1)}{2} + 12 * mn * \lfloor \log n \rfloor + o(n^2 + mn * \log(n))$ and garbage of at most $n + k - 1$, where $k = 2^{\lfloor \log n \rfloor}$.

2 Preliminaries

2.1 Reversible Circuits

Reversible logic design differs significantly from conventional logic design. A reversible circuit should be composed with reversible gates. In addition to the reversibility of gates, “no fan-outs” and “no feed-backs” [25] restrictions are applied. This leaves us with the cascade as the only possible structure. The circuit diagrams are built in the popular notations, such as those used in [25]. In short, horizontal “wires” carry information about single bit (qubit) each; the computation (time) in the circuit diagrams is propagated from left to right; gate controls (defined below) are depicted with $\bullet$; gate targets (defined below) appear as $\oplus$.

There is a limited number of reversible gates used for the synthesis of reversible circuits. Most popular among them are the gates associated with a cheap technological implementation. In this work we employ Toffoli gates [31] whose inexpensive quantum realizations are well known [3, 18, 25].

Definition 1. For the set of input variables $\{x_1, x_2, ..., x_n\}$ the **generalized Toffoli gate** has the form $TOF(C;t)$, where $C = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$, $t = \{x_j\}$ and $C \cap t = \emptyset$. It maps the Boolean pattern $\{x_1^0, x_2^0, ..., x_n^0\}$ to $\{x_1^0, x_2^0, ..., x_{j-1}^0, x_j^0 \oplus$

$x^0_{i_1}x^0_{i_2}...x^0_{i_k}, x^0_{j+1}, ..., x^0_n\}$. The set C which controls the change of the j-th bit is called the set of **controls** and t is called the **target**. ▶

$TOF(x_1) = \bar{x_1}$ is a NOT gate. $TOF(x_1; x_2) = (x_1, x_1 \oplus x_2)$[6] is often termed as a CNOT gate. In quantum technology, these gates are used as basic building blocks, therefore we associate a quantum cost of one with the use of each such gate. $TOF(x_1, x_2; x_3) = (x_1, x_2, x_3 \oplus x_1x_2)$ is usually referred to as a Toffoli gate [31]. The Toffoli gate is a composite gate, and in quantum it is simulated with 5 elementary operations [25]. Gates NOT, CNOT and Toffoli are depicted in Figure 1. Quantum implementations of larger Toffoli gates were also reported [3, 18]. For the purpose of future quantum cost calculation, we notice that a Toffoli gate with 3 controls can be simulated with 13 (or 15, depending on the set of basic quantum operations chosen) quantum operations [3] ([18]), and every Toffoli gate with $m > 3$ controls allows realization with $12m - 22$ elementary operations (assuming there are $m - 2$ temporary storage bits available) [18].

2.2 Multiple Output Symmetric Functions

Definition 2. Multiple output symmetric Boolean function $\overrightarrow{F}(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_m)$ is such a function that $\overrightarrow{F}(x_1, x_2, ..., x_n) = \overrightarrow{F}(\pi(x_1, x_2, ..., x_n))$, for any permutation π of its n inputs. ▶

A single output symmetric function with n inputs can be defined by its **carry vector** $(c_0, c_1, ..., c_n)$—a Boolean vector of length $n+1$ that consists of the output values c_i of the given symmetric function for the input patterns of weight i (where the weight is defined as a sum of ones in the pattern). Given a symmetric function, its carry vector can be computed through $n + 1$ variable substitutions: it suffice to compute symmetric function on the input values $(0, 0, ..., 0)$, $(0, 0, ..., 0, 1)$, $(0, 0, ..., 0, 1, 1)$, ... , $(1, 1, ..., 1)$. Due to its simplicity, *carry vector* is often used as an input/storage format for single output symmetric functions. Multiple output symmetric functions are stored as a set of Boolean carry vectors, or a single integer carry vector.

Definition 3. The **σ-function** $\sigma_n^k(x_1, x_2, ..., x_n)$ is defined as $\bigoplus_{\{i_1<i_2<...<i_k\}} x_{i_1}x_{i_2}...x_{i_k}$ for $k = 0, 1, ..., n$. ▶

Example 1. Over the set of 3 variables $\{x_1, x_2, x_3\}$, the σ-functions are those as listed below

$$\sigma_3^0 = 1$$
$$\sigma_3^1 = x_1 \oplus x_2 \oplus x_3$$
$$\sigma_3^2 = x_1x_2 \oplus x_1x_3 \oplus x_2x_3$$
$$\sigma_3^3 = x_1x_2x_3$$

The following two lemmas are well-known results employing the σ-functions.

Lemma 1. Every symmetric function can be written as a linear combination (with respect to EXOR operation) of not more than $(n + 1)$ different σ-functions.

Example 2. A 3-variable symmetric function $x_1x_2x_3 \oplus \bar{x_1}\bar{x_2}\bar{x_3}$ is equivalent to the linear combination $\sigma_3^0 \oplus \sigma_3^1 \oplus \sigma_3^2$. This can be easily verified through substitution $\bar{x_i} \to x_i \oplus 1$ on the left hand side and formulas for σ-functions (see Example 1) on the right hand side of the discussed equality.

Linear combination of the σ-functions is, in fact, equivalent to to the Positive Polarity Reed-Muller expansion (PPRM) [29], an EXOR polynomial with all literals appearing in the positive polarity. A more general object, Fixed Polarity Reed-Muller expansion (FPRM) for function $f(x_1, x_2, ..., x_n)$ is defined as

$$f(x_1, x_2, ..., x_n) = c_0 \oplus c_1x_1^* \oplus c_2x_2^* \oplus ... \oplus c_nx_n^* \oplus \; ... \; \oplus c_{2^n-1}x_1^*x_2^*...x_n^*,$$

where c_i are Boolean coefficients and x_i^* is either x_i or its complement $\bar{x}_i$. The term *Fixed Polarity* refers to the fact that each variable occurs in the expression in one way only, uncomplemented (x_i) or complemented ($\bar{x}_i$).

Any FPRM expression of a symmetric function with n inputs can be found in $O(n^3)$ time with $O(n^2)$ storage space [5]. This includes finding PPRM, which is equivalent to the linear combination of σ-functions. Similarly, an FPRM is a linear combination of polarized σ-functions. However, we will use PPRM when constructing the reversible implementations of symmetric functions. This is because the bottleneck of our approach is in construction of the largest degree σ-function that one needs to implement a given symmetric function. And, it can be shown that the degree of the largest degree σ-function participating in the expansion does not decrease when considering FPRMs instead of the plain PPRM.

Lemma 2. $\sigma_n^k(x_1, x_2, ..., x_n) = x_n\sigma_{n-1}^{k-1}(x_1, x_2, ..., x_{n-1}) \oplus \sigma_{n-1}^k(x_1, x_2, ..., x_{n-1})$ for $k \geq 2$.

Proof. Observe that the first part of the right hand side has all the terms of degree k which include variable x_n as a multiple. The second part has all the terms of degree k that do not include the variable x_n. Thus, right hand side has all the terms of degree k, which, according to the definition, forms the left hand side. ■

Example 3. We illustrate Lemma 2 for $n = 4$ and $k = 2$:

$$\begin{aligned}(RHS) \quad x_4\sigma_3^1(x_1, x_2, x_3) \oplus \sigma_3^2(x_1, x_2, x_3) \\ = x_4(x_1 \oplus x_2 \oplus x_3) \oplus (x_1x_2 \oplus x_1x_3 \oplus x_2x_3) \\ = x_1x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \\ = \sigma_4^2(x_1, x_2, x_3, x_4) \quad (LHS).\end{aligned}$$

Each function $\sigma_n^k(x_1, x_2, ..., x_n)$ is symmetric. Therefore, it can be described by a subset $M_k \subset \{0,1,2,...,n\}$ of the input weights where its output equals 1. In other words, M_k is a set of indices corresponding to the unit values of the carry vector of function $\sigma_n^k(x_1,x_2,...,x_n)$. Observe, that $l \in M_k \Leftrightarrow \binom{l}{k} \bmod 2 \equiv 1$. According to Kummer's Theorem [7] maximal $j : \binom{l}{k} | 2^j$ equals the number of bit carry over operations while adding numbers k and $l-k$ in binary. In particular, if k is a power of 2, that is, $k = 2^i$, and binary expansion of number l has zero i^{th} bit, then binary decomposition of $l-k$ also has zero i^{th} bit. In such case, addition of k with $l-k$ in binary requires no carry over, and thus $\binom{l}{k}$ is odd. Alternatively, if i^{th} bit of the binary expansion of number l equals one, i^{th} binary bit of $l-k$ also equals one, and there will be carry over while adding k with $l-k$ in binary. Summarizing, the set M_k for $k = 2^i$ consists of the numbers l with unit values in bit i. And so, every symmetric function equal to one for the input patterns with weight j only can be achieved as a product $\&_i(\sigma_{2^i} \oplus \alpha_i)$, where $\alpha_s ... \alpha_2\alpha_1$ is the binary expansion of j. This observation allows to formulate the following useful result.

Theorem 1. *Every symmetric function $y = f(x_1,x_2,...,x_n)$ with $M_f = \{j^1, j^2, ..., j^t\}$ can be computed using σ-functions $\sigma_1, \sigma_2, ..., \sigma_{2^s}$, $2^s \leq n < 2^{s+1}$ according to the formula*

$$y = \oplus_{r=1..t} \&_{i=1..s}(\sigma_{2^i} \oplus \alpha_i^r), \tag{1}$$

where $\alpha_s^r ... \alpha_2^r\alpha_1^r$ is the binary expansion of number j^r.

Note that from the point of view of applications, it makes sense to simplify expression (1) through applying techniques like EXORCISM [23]. In particular, designs of functions $sym12$ and $sym15$ shown in Table 5 will benefit from such simplification. 5 Toffoli5 gates used in the design of $sym12$ can be replaced with 1 Toffoli5 gate and 1 Toffoli gate (because $\bar{x_1}x_2\bar{x_3}\bar{x_4} \oplus \bar{x_1}x_2\bar{x_3}x_4 \oplus \bar{x_1}x_2x_3\bar{x_4} \oplus \bar{x_1}x_2x_3x_4 \oplus x_1\bar{x_2}\bar{x_3}\bar{x_4} = \bar{x_1}x_2 \oplus x_1\bar{x_2}\bar{x_3}\bar{x_4}$). Analogously, 6 Toffoli5 gates used in the design of $sym15$ can be replaced with 2 Toffoli5 and 2 CNOT gates (because $\bar{x_1}x_2\bar{x_3}x_4 \oplus \bar{x_1}x_2x_3\bar{x_4} \oplus \bar{x_1}x_2x_3x_4 \oplus x_1\bar{x_2}\bar{x_3}\bar{x_4} \oplus x_1\bar{x_2}\bar{x_3}x_4 \oplus x_1\bar{x_2}x_3\bar{x_4} = x_1\bar{x_2}x_3x_4 \oplus \bar{x_1}x_2\bar{x_3}\bar{x_4} \oplus x_1 \oplus x_2$).

3 Reversible Synthesis of Multiple Output Symmetric Functions

Our approach to the reversible synthesis of symmetric Boolean functions is as follows. We first consider a symmetric Boolean function defined by its carry vector. We next transform carry vector into the linear combination of σ-functions (PPRM) using procedure discussed in [5]. This is followed by construction of all needed σ-functions and synthesis of their linear combinations using simple modula-2 addition (through application of CNOT gates or the technique discussed in [26]). The above explains all except how and which σ-functions to construct. This is discussed next.

The statement of the Lemma 2 holds for $k = 1$. The result for $k = 1$ can be used to calculate $\sigma_n^1(x_1, x_2, ..., x_n)$. We suggest using Lemma 2 as a basis for the following dynamic programming algorithm which calculates the set of σ-functions $\{\sigma_n^1(x_1, x_2, ..., x_n), \sigma_n^2(x_1, x_2, ..., x_n), ..., \sigma_n^n(x_1, x_2, ..., x_n)\} = \{sigma[1], sigma[2], ..., sigma[n]\}$.

```
1. create Boolean array sigma[1..n]=0;
2. for i=1 to n
3.    for k=i down to 1
4.       if k>1 sigma[k] = (sigma[k] + x[i]*sigma[k-1]) mod 2;
5.       if k=1 sigma[k] = (sigma[k] + x[i]) mod 2;
6.    end for;
7. end for.
```

Note, that in a reversible circuit computing all σ-functions it is enough to have $2n$ rails marked $x_1, x_2, ..., x_n, \sigma_1, \sigma_2, ..., \sigma_n$ that initially hold the values $x_1, x_2, ..., x_n, 0, 0, ..., 0$. Rails $\sigma_1, \sigma_2, ..., \sigma_n$ are updated with regards to the formulas given in the above algorithm. It turns out that the gates associated with updating the values are very simple. In fact, a gate is only added to the reversible cascade on steps 4 and 5 of the algorithm. Step 4 requires addition of the Toffoli gate $TOF(x_i, \sigma_{k-1}; \sigma_k)$ and step 5 requires a simpler CNOT gate, $TOF(x_i; \sigma_k)$. Once all σ-functions are created, their linear composition can be used to construct all outputs, according to Lemma 1. These observations lead to the following theorem.

Theorem 2. *Every symmetric multiple output function* $\overrightarrow{F}(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_m)$ *can be realized with:*

- *at most* m *NOT gates, at most* $n + mn - 1$ *CNOT gates, and at most* $\frac{n(n-1)}{2}$ *Toffoli gates;*
- *at most* $n + m - 1$ *garbage bits;*
- *quantum implementation cost of at most* $\frac{5n(n-1)}{2} + mn + m + n - 1$ *elementary operations.*

Proof. The maximum of m NOT gates is used since every component of the output may require modula-2 addition of $\sigma_n^0 = 1$ function. The last can be EXORed with the target circuit rail through the use of one NOT gate. The number of outputs is m, making usage of m NOT gates sufficient for a computation of any m-output symmetric function.

Among the $n + mn - 1$ CNOT gates mn are used to create the outputs. Outputs are created on separate new rails $y_1, y_2, ..., y_m$ whose initial values are set to zero. According to Lemma 1, at most n CNOT gates are needed to create each output. The remaining $n - 1$ CNOTs are used in step 5 of the algorithm while constructing σ-functions. Their number is $n - 1$ instead of the expected n, since σ_1 can be computed on the input line x_n.

$\frac{n(n-1)}{2}$ Toffoli gates are used on the step 4 of the algorithm.

The rails in the constructed circuit are $x_1, x_2, ..., x_n (= \sigma_1), \sigma_2, ..., \sigma_n, y_1, y_2, ..., y_m$ and the outputs are built on the wires $y_1, y_2, ..., y_m$. Thus, the total garbage is $2n - 1$

bits. However, this number is reduced to $2n-2$ if output y_1 is created on the rail σ_i such that this σ_i belongs to the linear expansion of variable y_1. Further, it turns out that in most cases (in practice, all benchmarks considered in this paper have this property) it is possible to reuse wires $\sigma_1, \sigma_2, ..., \sigma_n$ for the output construction. This reduces the number of garbage bits to $2n-m-1$. ■

If all the outputs can be composed using the first $k+1$ $(2 \leq k \leq n)$ σ-functions $\sigma_n^0(x_1, x_2, ..., x_n), \sigma_n^1(x_1, x_2, ..., x_n), ..., \sigma_n^k(x_1, x_2, ..., x_n)$, there is no need to create the remaining $(n-k)$ σ-functions. This observation allows us to formulate the following result (proof is analogous to that of the previous theorem).

Theorem 3. *Every symmetric multiple output function $\overrightarrow{F}(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_m)$, such that its linear σ-function decomposition requires a σ-function of maximal degree k $(2 \leq k \leq n)$ can be realized with:*

- *at most m NOT gates, at most $n + mn - 1$ CNOT gates, and at most $\frac{(2n-k)(k-1)}{2}$ Toffoli gates;*
- *at most $n + k - 2$ garbage bits;*
- *and quantum implementation cost of at most $\frac{5(2n-k)(k-1)}{2} + mn + m + n - 1$ elementary operations.*

Example 4. Take a multiple output function $rd53$, which is the 5-input 3-output symmetric function whose output is the binary representation of the number of ones in its input. Carry vectors (their number is 3 according to the number of primary outputs) of this function are $(0,0,0,0,1,1)$, $(0,0,1,1,0,0)$, and $(0,1,0,1,0,1)$. They are first transformed to the vectors of PPRM coefficients [5]. The result of such transformation is a set of 3 vectors $(0,0,0,0,1,0)$, $(0,0,1,0,0,0)$, and $(0,1,0,0,0,0)$ with ones at position i meaning presence of σ^i in the linear σ-expansion of the target function. Next step, rewrite target function in terms of σ-functions—$(\sigma^4, \sigma^2, \sigma^1)$.

Written as a Boolean formula this expression is $(x_1x_2x_3x_4 \oplus x_1x_2x_3x_5 \oplus x_1x_2x_4x_5 \oplus x_1x_3x_4x_5 \oplus x_2x_3x_4x_5,\ x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_1x_5 \oplus x_2x_3 \oplus x_2x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_3x_5 \oplus x_4x_5,\ x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5)$. Note that σ^5-function needs not be built. Build the dynamic programming part that computes $\sigma^1, \sigma^2, \sigma^3$, and σ^4. Linear combinations need not be built because each output is a single σ-function. Also, rails σ^1, σ^2 and σ^4 can be reused for the output and thus new output rails y_1, y_2 and y_3 need not be introduced. Observe that the gates that affect a garbage bit whose changed value is not used by the design afterwards (the gate colored gray in Figure 2) can be deleted from the circuit without changing the output of the target function. The resulting circuit contains 12 gates.

In our further designs, if a gate affects a garbage bit whose changed value is not used in the circuit to affect useful output bits afterwards, it can be deleted from the design. This trivial procedure brings some simplification in almost every case. Also note, that the quantum cost of the boxed parts in the second circuit in Figure 2 is 4 (instead of 6=5+1 as one would expect), an implementation that was known to Peres [27]. Once these considerations are taken into account, the final quantum implementation cost will be lower than stated in the above theorems. We also suggest to use the templates [18] to further reduce the quantum costs of *all* presented designs.

Next, observe that using large Toffoli gates allows synthesis of symmetric specifications with a small number of outputs m, $m \prec \frac{n}{\log(n)}$ with smaller reversible gate count, garbage, and sometimes smaller quantum cost. Using Theorem 1 allows to compute σ-functions $\sigma_1, \sigma_2, ..., \sigma_{2^s}$, $2^s \leq n < 2^{s+1}$, first and then use at most n Toffoli gates with $\lfloor \log(n) \rfloor$ controls and at most s control bit negations to construct each of the m outputs of a given symmetric specification. The following Theorem summarizes this result.

Theorem 4. *Every symmetric multiple output function* $\overrightarrow{F}(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_m)$

can be realized with:

- *at most* m *NOT gates, at most* n *CNOT gates, at most* $\frac{(2n-k)(k-1)}{2}$ *Toffoli gates, and at most* nm *Toffoli gates with* $\lfloor \log(n) \rfloor$ *controls;*
- *at most* $n+k-1$ *garbage bits;*
- *quantum implementation cost of at most* $\frac{5(2n-k)(k-1)}{2} + 12 * nm * \lfloor \log(n) \rfloor + o(n^2 + mn * \log(n))$ *elementary operations,*

where $k = 2^{\lfloor \log(n) \rfloor}$.

4 Comparison of the Results

There were several design methods proposed in literature for the reversible design of multiple output Boolean functions. We would like to compare our results to the results of the RPGA method by Perkowski *et al.* [28] (the method designed to synthesize the symmetric functions with reversible gates), reversible wave cascades [24], Khan gate family synthesis [10, 11], generalized Toffoli gates family [15] and design of the Toffoli circuits using the templates [17]. The comparison consists of the three parts: comparison of the garbage, number of gates in the reversible cascade and comparison of the quantum costs.

Unfortunately, [28] do not provide a table of results, which makes it hard to do a precise comparison. The asymptotic reversible cost (number of gates) of both realizations, theirs and presented here, are the same, namely $O(n^2 + mn)$. But, the RPGA method uses excessive garbage, $\frac{n(n+1)}{2}$ (calculated in [15]), when the presented methods have the garbage of maximum $(2n-2)$. A good quantum realization of the Kerntopf gates used in [28] was never found, therefore we claim that from the point of view of quantum cost our method will produce quantum circuits which will be constant ($= \frac{14}{5}$ in quantum NOT, CNOT, controlled-*sqrt*-of-NOT gate library) times cheaper.

Comparison to the reversible wave cascades [24] (RWC columns), Khan gate family synthesis [11] (KGF columns) and generalized Toffoli gates family [15] (GT columns) reversible synthesis results is summarized in Table 5. Actual circuits for our designs can be found in [16].

The comparison in Table 5 is not quite fair. On one hand, the methods RWC, KGF and GT are general synthesis methods, which do not use special properties of functions. On the other hand, the cardinality of the set of gates of these is greater on the order than the number of gates used in the presented method.

It can be seen that our method produces better results for larger functions both from the point of view of the reversible cost and garbage. The presented method can never beat the generalized Toffoli gates family synthesis method in terms of the number of garbage bits, since the last uses theoretically minimal number of garbage bits. But, the GT method scales badly—it can produce circuits for functions with no more than 10 inputs. The RWC and KGF are synthesized heuristically and their usage is expected to cause problems when scaled.

We were not able to compare the quantum costs of the presented designs to those of earlier methods due to impossibility or hardness of getting access to the actual circuits. However, in few cases the comparison of quantum costs could be made. [17] gives an example of a circuit for $rd53$ function with 12 reversible gates, which seems to be the smallest (reversible gate count wise) among all known. The generalized Toffoli gates used in [17] are expensive (yet, generally, less expensive than the gates in RWC, KGF and GT) and the quantum cost calculation based on [16] reports the quantum cost of 120 for that realization. In the same costing metric, our 12-gate realization of $rd53$ has the quantum cost of 36 only, which is more than 3 times lower. [15] presents a circuit for $rd53$ with cost 232. Which, again, compares favorably to our 36.

Another interesting comparison can be made using $2of5$ function. Realization in [15] uses 7 reversible gates in comparison to 12 in the presented paper. However, the quantum cost of their realization is 158 in comparison to 32 for our circuit. Thus, we find our realization to be potentially more practical. This example also serves as a good illustration of the thesis [22] that a small number of reversible gates does not mean a cheap technological implementation.

Secondly, we synthesized reversible circuits for some symmetric functions or symmetric components of some benchmarks [5] whose reversible implementations were never reported before. The results can be found in Table 5. Columns **name**, **in**, **out#**, and **Car. Vec.** list the name of the benchmark function, total number of its inputs, symmetric output number (in case if not all outputs are symmetric), and carry vector of the symmetric component considered. We next compute PPRM expansion using the transeunt triangle [5]. It is listed in column **PPRM coef.** All circuits were synthesized assuming there might be an additional computation on top of computing the symmetric components. Thus, all primary inputs are returned unaltered. Some properties of the synthesized circuits are listed in the remaining three columns. Column **gates** lists the gate counts according to their types with AN representing the number of NOT gates, AC representing the number of CNOT gates, AT representing the number of Toffoli gates, and ATB representing the number of generalized Toffoli gates of size B, where A is a constant equal to the number of gate types used. Column **garb.** lists the number of garbage bits required (in this setting, garbage bits are all outputs except primary inputs that are being passed unchanged and the primary output). Finally, column **QC** lists an *upper bound* for the quantum cost of the circuits. We stress that this is an *upper bound*, because the listed number is the weighted gate count, which, for instance, does not take it into account that some sequences of Toffoli-CNOT gates added into the total sum as 6 are Peres gate with the cost 4. Further simplification could be achieved through dropping the gates affecting garbage outputs only and applying local optimization techniques [18].

5 Conclusion

In this paper we presented an efficient reversible/quantum synthesis method for the class of multiple output symmetric Boolean functions. As compared to the best previously reported method targeting the synthesis of symmetric Boolean functions, our method uses simpler gates (resulting in technologically preferable circuit specifications) and requires significantly less garbage bits. We compared our designs to those presented previously and found that our circuits are smaller. We presented reversible implementations for some well known symmetric benchmark functions whose reversible circuits were never reported before. Further advance of our synthesis approach includes optimization of the presented circuits; and synthesizing almost symmetric functions (which is, likely, a separate problem rather than a trivial extension of the presented technique).

Acknowledgments

The author wishes to thank Prof. S. J. Lomonaco from the University of Maryland Baltimore County and Prof. S. B. Gashkov from Lomonosov's Moscow State University for their useful discussions, and C. Young from the University of Victoria for help in preparation of this manuscript.
This work was supported by NSERC, Canada.

References

[1] IBM's Test-Tube Quantum Computer Makes History. IBM T.J. Watson Research Center, http://researchweb.watson.ibm.com/resources/news/20011219_quantum.shtml, December, 2001.

[2] A. Agrawal and N. K. Jha. Synthesis of reversible logic. In *DATE*, pages 21384–21385, Paris, France, February 2004.

[3] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, 1995.

[4] C. H. Bennett. Logical reversibility of computation. *IBM J. Research and Development*, 17:525–532, November 1973.

[5] J. T. Butler, G. W. Dueck, S. Yanushkevich, and V. Shmerko. Comments on Sympathy: Fast exact minimization of fixed polarity Reed-Muller expasion for symmetric functions. *IEEE Transactions on CAD*, 19(11):1386–1388, November 2000.

[6] R. Feynman. Quantum mechanical computers. *Optic News*, 11:11–20, 1985.

[7] S. B. Gashkov and V. N. Chubarikov. *Arithmetic. Algorithms. Complexity of evaluations.* Vysshaya Shkola, Moscow, Russia, 2000 (in Russian).

[8] R. Hughes and others. *Quantum Computing Roadmap.* University of California for the National Nuclear Security Administration, of the US Department of Energy, http://qist.lanl.gov, April 2004.

[9] P. Kerntopf. A new heuristic algorithm for reversible logic synthesis. In *DAC*, pages 834–837, June 2004.

[10] M. H. A. Khan and M. Perkowski. Logic synthesis with cascades of new reversible gate families. In *6th International Symposium on Representations and Methodology of Future Computing Technology*, pages 43–55, March 2003.

[11] M. H. A. Khan and M. Perkowski. Multi-output ESOP synthesis with cascades of new reversible gate family. In *6th International Symposium on Representations and Methodology of Future Computing Technologies*, pages 144–153, March 2003.

[12] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, pages 46–52, January 2001.

[13] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Research and Development*, 5:183–191, 1961.

[14] S. Lee, S. Lee, T. Kim, J. Lee, J. Biamonte, and M. Perkowski. The cost of quantum gates. *IEEE International Journal of Multi Valued Logic*, 2005. In review.

[15] D. Maslov and G. Dueck. Reversible cascades with minimal garbage. *IEEE Transactions on CAD*, 23(11):1497–1509, November 2004.

[16] D. Maslov, G. Dueck, and N. Scott. Reversible logic synthesis benchmarks page. http://www.cs.uvic.ca/˜dmaslov/, August 2004.

[17] D. Maslov, G. W. Dueck, and D. M. Miller. Toffoli network synthesis with templates. *IEEE Transactions on CAD*, 24(6):807–817, June 2005.

[18] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck. Quantum circuit simplification using templates. In *DATE*, March 2005.

[19] R. C. Merkle. Reversible electronic logic using switches. *Nanotechnology*, 4:21–40, 1993.

[20] R. C. Merkle. Two types of mechanical reversible logic. *Nanotechnology*, 4:114–131, 1993.

[21] D. M. Miller. Spectral and two-place decomposition techniques in reversible logic. In *Midwest Symposium on Circuits and Systems*, August 2002.

[22] D. M. Miller and D. Maslov. Comparison of the cost metrics for reversible and quantum logic synthesis. Technical Report quant-ph/0511008, November 2005.

[23] A. Mishchenko and M. Perkowski. Fast Heuristic Minimization of Exclusive Sum-of-Products. *5th International Reed-Muller Workshop*, pages 242–250, August, 2001.

[24] A. Mishchenko and M. Perkowski. Logic synthesis of reversible wave cascades. In *IWLS*, pages 197–202, June 2002.

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

[26] K. N. Patel, I. L. Markov, and J. P. Hayes. Efficient synthesis of linear reversible circuits. In *IWLS*, pages 470–477, Temecula Creek, CA, June 2004.

[27] A. Peres. Reversible logic and quantum computers. *Physical Review A*, 32:3266–3276, 1985.

[28] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, and B. Massey. Regularity and symmetry as a base for efficient realization of reversible logic circuits. In *IWLS*, pages 245–252, 2001.

[29] T. Sasao. *Switching theory for logic synthesis.* Kluwer Academic Publishers, Norwell, MA, 1999.

[30] V.V. Shende, A.K. Prasad, I.L. Markov, and J.P. Hayes. Synthesis of reversible logic circuits. *IEEE Transactions on CAD*, 22(6):723–729, June 2003.

[31] T. Toffoli. Reversible computing. *Tech memo MIT/LCS/TM-151, MIT Lab for Comp. Sci*, 1980.

[32] I. M. Tsai and S. Y. Kuo. Quantum boolean circuit construction and layout under locality constraint. In *IEEE Conference on Nanotechnology*, pages 111–116, 2001.

Authors' affiliations:

Dmitri Maslov (Department of Computer Science, University of Victoria, Victoria, BC, V8W 3P6, Canada)

E-mail: dmaslov@uvic.ca

Table captions:

Tab. 1 Comparison of the results to RWC, KGF and GT

Tab. 2 The results of synthesis for some symmetric benchmarks

Figure captions:

Fig. 1 (a) $TOF(x_1)$, (b) $TOF(x_1; x_2)$ and (c) $TOF(x_1, x_2; x_3)$ Toffoli gates

Fig. 2 Circuit for $rd53$

Table 1

function			number of gates				garbage			
name	in	out	RWC	KGF	GT	Ours	RWC	KGF	GT	Ours
2of5	5	1	N/A	N/A	7	12	N/A	N/A	5	6
rd53	5	3	14	17	13	12	19	19	4	5
rd73	7	3	36	43	37	20	43	47	6	7
rd84	8	4	58	64	N/A	28	66	68	7	11
6sym	6	1	N/A	N/A	13	20	N/A	N/A	6	9
9sym	9	1	52	52	N/A	28	61	60	9	11
xor5	5	1	5	5	4	4	10	9	4	4

Table 2

function					circuit		
name	in	out#	Car. Vec.	PPRM coef.	gates	garb.	QC
co14	14	0	010^{13}	010^{13}	13T	0	13
m1	6	6	10^{6}	10^{6}	1N	0	1
m4	6	8	10^{6}	10^{6}	1N	0	1
misex5	5	5	$0^{5}1$	$0^{5}1$	8C+9T+1T4	3	67
misj	10	10	$1^{10}0$	$10^{9}1$	1N+18C+45T	7	244
sym4	4	0	$01^{2}0^{2}$	$01^{2}0^{2}$	8C+3T	0	23
sym10	10	0	$0^{4}1^{5}0^{2}$	$0^{4}10^{3}1^{2}0$	20C+44T	6	240
sym12	12	0	$0^{4}1^{5}0^{4}$	$0^{4}10^{3}1^{3}0$	22C+56T+5T5	7	432
sym15	15	0	$0^{5}1^{6}0^{5}$	$0^{5}1^{4}0^{2}1^{4}0$	28C+77T+6T5	7	569
dbruijn_2	4	0	$0^{2}1^{2}0$	$0^{2}10^{2}$	6C+3T	0	21
dbruijn_3	9	0	$0^{3}101^{3}0^{2}$	$0^{3}101^{2}0^{3}$	18C+30T	4	168
dbruijn_4	18	0	$0^{4}10^{2}1^{2}0101^{4}0^{3}$	$0^{4}1^{3}01^{2}0^{2}1010^{4}$	40C+143T	12	755
dbruijn_5	35	0	$0^{5}10^{3}1^{2}0^{2}101$	$0^{5}10101^{2}0^{2}1^{3}$	90C+580T	28	2990
			$0^{2}1^{3}0101^{2}01^{5}0^{4}$	$0^{2}10101^{7}010^{5}$			

Figure 1

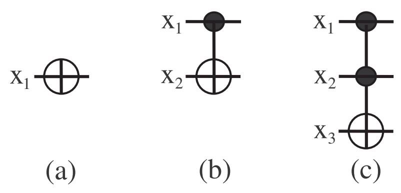
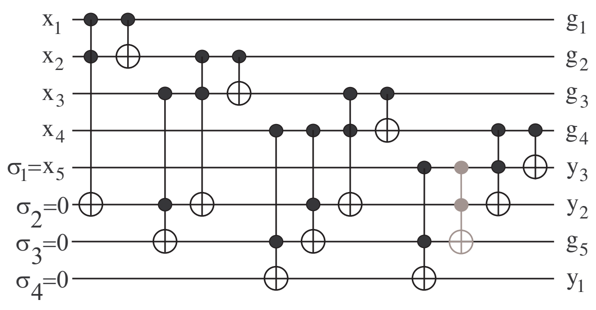
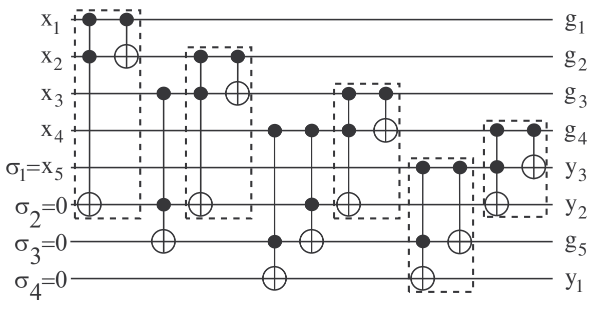


Figure 2



x_1
x_2
x_3
x_4
$\sigma_1=x_5$
$\sigma_2=0$
$\sigma_3=0$
$\sigma_4=0$
g_1
g_2
g_3
g_4
y_3
y_2
g_5
y_1



x_1
x_2
x_3
x_4
$\sigma_1=x_5$
$\sigma_2=0$
$\sigma_3=0$
$\sigma_4=0$
g_1
g_2
g_3
g_4
y_3
y_2
g_5
y_1