A New Heuristic Algorithm for Reversible Logic Synthesis

Pawel Kerntopf
Institute of Computer Science, Department of Electronics and Information Technology
Warsaw University of Technology, Nowowiejska Str. 15/19, 00-665 Warsaw, Poland, phone: +48-22-6607915
P.Kerntopf@ii.pw.edu.pl

ABSTRACT
Reversible logic has applications in many fields, including quantum computing. Synthesis techniques for reversible circuits are not well developed, even for functions with a small number of inputs and outputs. This paper proposes an approach to reversible logic synthesis using a new complexity measure based on shared binary decision diagrams with complemented edges (instead of truth tables or PPRM forms, as in the previous algorithms). The approach can be used with arbitrary libraries of reversible logic gates and arbitrary cost functions. Experiments show promising results in comparison with the known approaches.

Categories and Subject Descriptors
B.6.3 [**Logic Design**]: Design Aids – Automatic Synthesis

General Terms
Design, Theory.

Keywords
Reversible Logic Circuits, Synthesis.

1. INTRODUCTION
Recently, reversible computing has become a fast developing area of research. It has attracted attention of researchers because of new perspectives to build almost energy loss-less, ultra-small, and ultra-fast quantum computers. Moreover, there are some tasks in other areas, including cryptography, digital signal processing, communication, and computer graphics, requiring that all the information encoded in the inputs be preserved in the outputs [16].

Classical reversible circuits are a special case of quantum circuits. Logic synthesis for them is a first step toward synthesis of quantum circuits. The synthesis of reversible circuits differs substantially from synthesis using traditional irreversible gates. A circuit is said to be reversible if there is a bijective mapping of the input assignments into the output assignments. So, it is assumed that the number of outputs in a reversible circuit (or gate) is the same as the number of inputs. In addition, the fan-outs are not allowed. Thus, logic design of reversible circuits is a new and challenging task. Synthesis techniques are not well developed for such circuits, even for small numbers of inputs and outputs.

DAC 2004, June 7–11, 2004, San Diego, California, USA.

Although some general ideas and algorithms for reversible logic synthesis were proposed in [15, 14, 6, 12, 3, 16, 1], satisfactory solutions for arbitrary libraries of gates and cost functions have not yet been found. An incremental algorithm for implementing a reversible function was proposed in [7]. Since that time, several variants of this algorithm were developed [13, 9, 2, 8, 10, 11]. It is a two-stage algorithm. First, a circuit is constructed in a number of steps by inspecting the truth table of a reversible function. In [9, 2, 10, 11] the concept of a template, first introduced in [3], was substantially extended and used in the second stage of the algorithm to simplify the circuit found in the first stage.

In this paper, a new incremental approach is presented, which uses shared binary decision diagrams for the representation of reversible functions (instead of truth tables) and for measuring their complexity. The proposed algorithm selects reversible gates, one at a time, based on the complexity of the reminder logic. Similarly to [1-2, 7-11, 16, 17], we assume that a reversible function to be implemented is realizable without temporary storage [16] and that the cost function is the gate count.

The paper is organized as follows. Section 2 introduces the basic concepts of reversible logic. Section 3 analyses the previous work. Section 4 introduces a new algorithm. Section 5 presents experimental results. Section 6 summarizes the paper. This paper assumes that the reader is familiar with the basic concepts of Binary Decision Diagrams (BDDs), including Shared BDDs (SBDDs) with complemented edges [17].

2. PRELIMINARIES
Definition 1 A completely specified *n*-input *n*-output Boolean function (referred to as *n*n* function) is called *reversible* if it maps each input assignment into a unique output assignment.

Definition 2 An *n*-input *n*-output gate (or circuit) is *reversible* if it realizes an *n*n* reversible function.

A variety of reversible gates have been proposed in the literature. In this paper, we consider only the most widely used gates.

Definition 3 NOT, CNOT, Toffoli, SWAP and Fredkin gates (in short, N, C, T, S and F, respectively) are defined as follows:

1*1 NOT: $a' = 1 \oplus a$;

2*2 CNOT: $a' = a$, $b' = a \oplus b$;

3*3 Toffoli: $a' = a$, $b' = b$, $c' = c \oplus ab$;

2*2 SWAP: $a' = b$, $b' = a$;

3*3 Fredkin: $a' = a$, if $a = 0$ then $b' = b$, $c' = c$ if $a = 1$ then $b' = c$, $c' = b$, or equivalently: $b' = b \oplus ab \oplus ac$, $c' = c \oplus ab \oplus ac$.

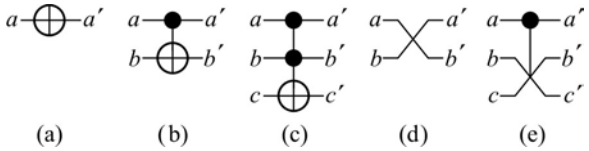

Figure 1: NOT, CNOT, Toffoli, SWAP and Fredkin gates.

The pictorial representations of the gates from Definition 3 are shown in Fig. 1a, 1b, 1c, 1d, and 1e. The set of the first three gates is called NCT library, the set of the first four gates is called NCTS library, and the set of all five gates is called NCTSF library. Note that each of the N, C, T, S, F gates is invertible (i.e. equal to its own inverse). Thus the equations for the functions describing dependence of inputs on outputs (inverse mapping) have the same form as the equations given in Definition 3 (after exchanging a, b, c with a', b', c', respectively). For each non-invertible gate it is possible to determine sets of equations defining both forward and inverse mappings induced by the gate.

In [2, 7-11, 13] the following definition is used.

Definition 4 The Hamming distance between two bit strings is the number of positions, in which they differ. Given a reversible function f, *the complexity measure* $C(f)$ is equal to the sum of the individual Hamming distances between input assignments and output assignments for all rows of the truth table of f.

After considering different variants of complexity measures based on decision diagrams, we have chosen the following one:

Definition 5 *The complexity measure* $D(f)$ of an $n*n$ reversible function f is equal to $D(f) = s(f) - n$, where $s(f)$ denotes the number of non-terminal nodes in the reduced ordered SBDD of f with complemented edges. Thus, D(identity function) = 0.

Example 1 $D(f)$ for functions represented by SBDDs shown in Figures 4a, 4b, 4c, 4d, 4e, 4f is equal to 5, 4, 3, 3, 2, 0, respectively (Figure 4f represents the identity function).

SBDDs provide a compact representation of multiple-output functions after choosing an appropriate variable ordering [17]. However, in the context of reversible logic synthesis, minimization of SBDDs is not necessary, because we deal with functions of a relatively small number of variables and we are only interested in relative changes of the value $D(f)$. Therefore, we are building SBDDs using the natural order of the inputs.

3. PREVIOUS WORK

Let us first outline the basic algorithm from [13], which we call the DMM algorithm. Its goal is to find a sequence of reversible gates, which transforms a reversible function to the identity function. The function representation is a standard truth table with input assignments arranged in the lexicographical order. At every step of the algorithm, some gates from NCT library are chosen and added to previously selected gates, starting from the end of the constructed circuit towards its beginning. The truth table is being inspected in the lexicographical order until the first output assignment is encountered, which is not equal to the input assignment located in the same row of the table. A subsequence of gates generated has to transform the table in such a way that the output assignment will be the same as the input assignment.

The algorithm is developed so that once an output assignment of the truth table is transformed into the correct pattern, it remains unchanged regardless of the transforms of the table req

uired for the later transformation of remaining rows with unequal input and output patterns. The algorithm always terminates successfully with a circuit implementing the given truth table. For NCTSF library, in which all gates are invertible, the reversible cascade can be built from either end. Thus, all synthesis algorithms can be applied from outputs to inputs, from inputs to outputs, or in both directions simultaneously (the last case is called bi-directional).

There are some disadvantages of the DMM approach. It was not shown how to take into account different costs of gates. Also, only algorithms for NCTSF library have been developed. Simplification of the sub-optimal circuits based on recognizing templates may require a lot of iterations. It is because, after each usage of a template, new template matches may appear. It is also possible that during the simplification a local optimum may be reached, instead of a global one.

In some cases, the circuits constructed by the DMM algorithm are far from optimal. It is because transformations corresponding to the sequence of gates in an optimal circuit do not yield a match of subsequent pairs of input and output assignments according to the lexicographical order. Also, the complexity measures $C(f)$ for the functions obtained by the transformations corresponding to subsequent gates in an optimal circuit do not form a non-increasing sequence.

4. NEW ALGORITHM

In Figure 2, the scheme of one basic step of our synthesis algorithm is shown. Let G_1 is the first from the left side gate of the circuit. As shown in [4, 5], the transformed $n*n$ reversible function f^T can be represented as an SBDD (such representation of a function f is called a function-driven decision diagram defined by the output function of G_1). We have chosen the representation f^T in the form of a reduced ordered SBDD with complemented edges. The transformation shown in Figure 2 can be iterated.

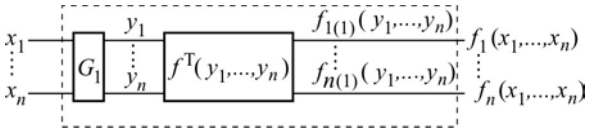

Figure 2: A general scheme of one step of our algorithm.

The functions $f_{1(1)}(y_1,\dots,y_n)$, ... , $f_{n(1)}(y_1,\dots,y_n)$ are derived by substituting $x_1=g_1(y_1,\dots,y_n)$, ... , $x_n=g_n(y_1,\dots,y_n)$ in the functions:

$$f_1(x_1, \dots, x_n), \dots, f_n(x_1, \dots, x_n).$$

Example 2 Let f be a 3*3 reversible function defined as follows:

$$f_{1(0)} = 1 \oplus a \oplus c \oplus ab \oplus ac$$
$$f_{2(0)} = 1 \oplus a \oplus b \oplus c$$
$$f_{3(0)} = 1 \oplus a \oplus b \oplus ab \oplus bc$$

We will show how to determine the reminder function f^T with respect to the Fredkin gate with the control input being a. Thus, according to the remark following Definition 3, we have

$a = a'$, $\quad b = b' \oplus a'b' \oplus a'c' \quad c = c' \oplus a'b' \oplus a'c'$.

We substitute the above expressions for a, b, c into the expressions defining the function f (in our algorithm, these calculations are performed using decision diagrams):

$f_{1(1)} = f(a, b, c) = 1 \oplus a \oplus c \oplus ab \oplus ac = 1 \oplus a' \oplus c' \oplus a'b' \oplus a'c' \oplus a'(b' \oplus a'b' \oplus a'c') \oplus a'(c' \oplus a'b' \oplus a'c') = 1 \oplus a' \oplus c' \oplus a'b' \oplus a'c' \oplus a'b' \oplus a'b' \oplus a'c' \oplus a'c' \oplus a'b' \oplus a'c = 1 \oplus a' \oplus c'$

$f_{2(1)} = 1 \oplus a \oplus b \oplus c = 1 \oplus a' \oplus b' \oplus a'b' \oplus a'c' \oplus c' \oplus a'b' \oplus a'c' = 1 \oplus a' \oplus b' \oplus c'$

$f_{3(1)} = 1 \oplus a \oplus b \oplus ab \oplus bc = 1 \oplus a' \oplus (b' \oplus a'b' \oplus a'c') \oplus a'(b' \oplus a'b' \oplus a'c') \oplus (b' \oplus a'b' \oplus a'c')(c' \oplus a'b' \oplus a'c') = 1 \oplus a' \oplus b' \oplus a'b' \oplus a'c' \oplus a'b' \oplus a'b' \oplus a'c' \oplus b'c' \oplus a'b' \oplus a'b'c' \oplus a'b'c' \oplus a'b' \oplus a'b'c' \oplus a'c' \oplus a'b'c' \oplus a'c' = 1 \oplus a' \oplus b' \oplus a'b' \oplus b'c'$

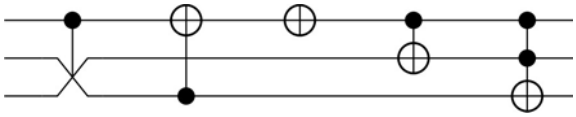

Figure 3: An optimal circuit for the function in Example 2.

Now we will make transformations for the next gates shown in Figure 3 (in the order from left to right). For simplicity, the new variables after each transformation are denoted using the same characters as for the previous names of variables. Namely, instead of *a*', *b*', *c*' we use *a*, *b*, *c* after each iteration. The transformed functions are as follows (the calculations are omitted):

$f_{1(2)} = 1 \oplus a$ $\quad f_{2(2)} = 1 \oplus a \oplus b$ $\quad f_{3(2)} = 1 \oplus a \oplus b \oplus c \oplus ab$

$f_{1(3)} = a$ $\quad f_{2(3)} = a \oplus b$ $\quad f_{3(3)} = a \oplus c \oplus ab$

$f_{1(4)} = a$ $\quad f_{2(4)} = b$ $\quad f_{3(4)} = c \oplus ab$

$f_{1(5)} = a$ $\quad f_{2(5)} = b$ $\quad f_{3(5)} = c$

The SBDDs for these functions are presented in Figure 4. The sequence of $D(f)$'s is non-increasing: 5, 4, 3, 3, 2, 0.

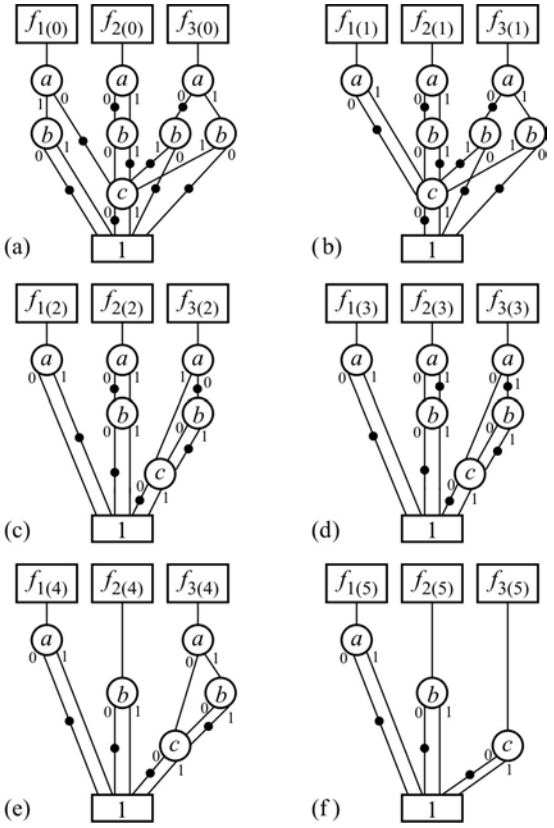

Figure 4: SBDDs for the functions calculated in Example 2.

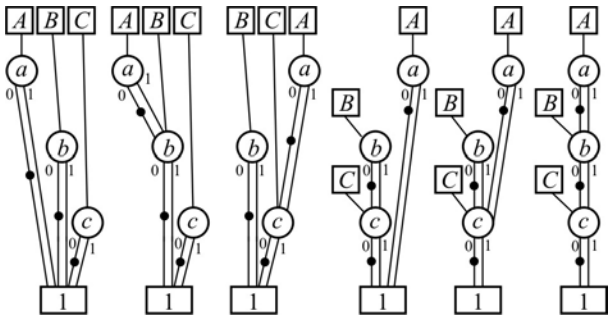

Figure 5: Possible structures of SBDDs for $D(f) = 0$, $n = 3$.

In every step of our algorithm all gates are examined and for each of them SBDD f^T is constructed. Next we select gates, for which the size of the transformed function is minimal. If there is more than one such gate, we proceed further with all of them. In our experiments, the algorithm always terminated with the circuit realizing the given function. Usually, more that one circuit was found. Thus, it is possible to select the circuits having the minimal cost for a given reversible function.

Figure 6: Moving SWAP gates.

The performance of the algorithm can be improved in two ways. Structures of SBDD with $D(f) = 0$ can be easily determined. In Figure 5, all such structures for $n = 3$ are shown (up to the positions of dots). For each of these structures determining the optimal subcircuit is straightforward. Depending on positions of dots, it will consist of up to n NOT gates, and up to n-1 SWAP gates. Thus, this part of the circuit can be constructed very fast once it is established that $D(f) = 0$. Also, note that the SWAP gate may be always exchanged with the next (or previous) gate in the order (see Figure 6) after an adequate transformation of the gate G_1 into the gate G_2 (consisting in permutations of inputs and outputs). Due to this property we can postpone considering SWAP gates until the end of the constructed circuit. In this way a speed-up of the algorithm can be obtained. These improvements are implemented in our algorithm.

5. EXPERIMENTAL RESULTS

To compare our results with those obtained using the DMM algorithm, our program synthesized all 40,320 reversible 3*3 functions using two versions of the algorithm:

(a) one-directional and bi-directional algorithms (columns "NewA" in Tables 1 and 2)

(b) two results for one-directional algorithm (run from the inputs to the outputs, and vice versa) are obtained and the best of them is chosen (columns "NewB" in Tables 1 and 2).

Tables 1 and 2 show how many 3*3 functions can be realized with the specified number of gates (column "Size") for NCTS and NCTSF libraries, respectively. "WA" shows weighted average of the circuit size. We compared the results of our algorithm to the results of optimal synthesis [16] (columns "Optimal") and the results of using the DMM algorithm. Column "DMM" in Table 1 denotes the results obtained without using templates (Table 4, column "(b)" and (d) in [13]). Such results have not been reported for the NCTSF library (this is why the third column in Table 2 is empty). The results for the bidirectional DMM algorithm without application of templates are shown in the sixth column of Table 1. The column "DMM+" in Table 2 shows the results of application of a small set of templates [2], while the column "DMM*" denotes the results of using an extended set of templates [11]. It has to be noted that some of the results for the DMM algorithm have been obtained after output permutation while the optimal and our values were calculated without using output permutation.

For NCTS library our algorithm "NewB" produces the circuits which are 106.8% of the optimal size on average. This is better than the results of the bi-directional DMM algorithm followed by exhaustive simplification procedures. For NCTSF library, our results are slightly worse than in "DMM*". However, application of templates to them would probably yield better results.

We have not yet optimized our programs, so the runtimes are not given. It seems that for synthesizing large functions our approach has better potential than previously reported algorithms due to using decision diagrams as a basic representation of reversible functions, instead of truth tables or PPRM.

Table 1: Number of 3*3 reversible functions using a specified number of gates for NCTS library

Size	Optimal	1-directional algorithms			Bidirectional algorithms		
		DMM	NewA	NewB	DMM	DMM+	NewA
14	0	4	0	0	0	0	0
13	0	72	0	0	0	0	0
12	0	477	0	0	3	0	0
11	0	1759	0	0	86	5	2
10	0	4179	24	0	493	110	33
9	0	6912	861	86	2312	792	794
8	32	8389	6174	2740	6944	4726	5722
7	6817	7766	12683	11774	11206	11199	12181
6	17531	5615	11419	13683	10169	12076	11715
5	11194	3183	6127	8068	5945	7518	6598
4	3752	1391	2278	3038	2375	2981	2474
3	844	453	619	781	650	767	661
2	134	104	119	134	121	130	124
1	15	15	15	15	15	15	15
0	1	1	1	1	1	1	1
WA	5.629	7.646	6.362	6.010	6.527	6.176	6.298
%	100	135.8	113.0	106.8	116.0	109.7	111.9

Table 2: Number of 3*3 reversible functions using a specified number of gates for NCTSF library

Size	Optimal	1-directional algorithms			Bidirectional algorithms		
		DMM	NewA	NewB	DMM+	DMM*	NewA
10	0	?	22	0	1	0	9
9	0	?	721	53	86	9	220
8	0	?	6968	2731	1877	512	2712
7	496	?	8579	7151	8419	5503	9607
6	14134	?	11300	13285	13606	13914	13602
5	17695	?	8292	11094	10595	13209	9248
4	6474	?	3360	4633	4437	5680	3750
3	1318	?	899	1170	1105	1290	984
2	184	?	160	184	175	184	169
1	18	18	18	18	18	18	18
0	1	1	1	1	1	1	1
WA	5.134	?	6.156	5.704	5.724	5.437	5.882
%	100	?	119.9	111.1	111.5	105.9	114.6

6. CONCLUSIONS

The approach presented here works with arbitrary libraries, while the DMM algorithm so far is developed only for NOT, CNOT, Toffoli, SWAP, and Fredkin gates or their generalizations. Our algorithm usually produces several circuits for a given function. Thus, for a library with gates of different cost, all generated circuits can be evaluated to select the circuit with the minimal cost. Such optimization is done on the global scale, in contrast to the local decisions of the algorithms presented in [1, 2, 7-13].

We are studying modifications of our complexity measure to incorporate more data for obtaining better efficiency of the algorithm. Ultimately, we aim to generalize the proposed approach to synthesis of multiple-valued reversible functions for arbitrary libraries of reversible MV gates.

7. REFERENCES

[1] Agrawal, A., Jha, N. K. Synthesis of reversible logic. In *Proc. Design and Test in Europe Conference*, Paris, France, February 2004, 1384-1385.

[2] Dueck, G. W., Maslov, D., and Miller, D. M. Transformation-based synthesis of networks of Toffoli/Fredkin gates. In *IEEE Canadian Conference on Electrical and Computer Engineering*, Montreal, Canada, May 2003.

[3] Iwama, K., Kambayashi, Y., and Yamashita, S. Transformation rules for designing CNOT-based quantum circuits. In *Proc. Design Automation Conference*, New Orleans, LA, June 2002, 419-425.

[4] Kerntopf, P. An approach to minimization of decision diagrams. In *Proc. EUROMICRO Symposium on Digital Systems Design*, Warsaw, Poland, Sept. 2001, 79-86.

[5] Kerntopf, P. Binary decision diagrams based on single and multiple generalized Shannon expansions. In *Proc. 6th International Symposium on Representations and Methodology of Future Computing Technology*, Trier, Germany, March 2003, 183-190.

[6] Khlopotine, A., Perkowski, M., and Kerntopf, P. Reversible logic synthesis by gate composition. In *Notes of the 11th IEEE/ACM International Workshop on Logic and Synthesis*, New Orleans, LA, June 2002, 261-266.

[7] Maslov, D., and Dueck, G. W. Garbage in reversible design of multiple output functions. In *Proc. 6th International Symposium on Representations and Methodology of Future Computing Technology*, Trier, Germany, March 2003, 162-170.

[8] Maslov, D., and Dueck, G. W. Asymptotically optimal regular synthesis of reversible logic networks. In *Notes of the 12th IEEE/ACM International Workshop on Logic and Synthesis*, Laguna Beach, CA, May 2003, 226-230.

[9] Maslov, D., and Dueck, G. W. Templates for Toffoli network synthesis. In *Notes of the 12th IEEE/ACM International Workshop on Logic and Synthesis*, Laguna Beach, CA, June 2003, 320-325.

[10] Maslov, D., Dueck, G. W., and Miller, D. M. Simplification of Toffoli networks via templates. In. *16th Symposium on Integrated Circuits and System Design*, Sao Paulo, Brazil, Sept. 2003.

[11] Maslov, D., Dueck, G. W., and Miller, D. M. Fredkin/Toffoli templates for reversible logic synthesis. *Proc. International Conference on Computer-Aided Design,* San Jose, CA, Nov. 2003.

[12] Miller, D. M., and Dueck, G. W. Spectral techniques for reversible logic synthesis. In *Proc. 6th International Symposium on Representations and Methodology of Future Computing Technology*, Trier, Germany, March 2003, 56-62.

[13] Miller, D. M., Maslov, D., and Dueck, G. W. A transformation based algorithm for reversible logic synthesis. *Proc. Design Automation Conference*, Anaheim, CA, June 2003, 318-323.

[14] Mishchenko, A., and Perkowski, M. Logic synthesis of reversible wave cascades. In *Notes of the 11th IEEE/ACM International Workshop on Logic and Synthesis*, New Orleans, LA, June 2002, 197-202.

[15] Perkowski, M., Jozwiak, L., Kerntopf, P., Mishchenko, A., Al-Rabadi, A., Coppola, A., Buller, A., Xiaoyu Song, Khan, M. H. A., Yanushkevich, S. N., Shmerko, V. P., and Chrzanowska-Jeske, M. A General Decomposition for Reversible Logic. *Proc. 5th International Workshop on Applications of Reed-Muller Expansion in Circuit Design*, Starkville, MS, Aug. 2001, 119-138.

[16] Shende, V. V., Prasad, A. K., Markov, I. L., and Hayes, J. P. Reversible logic circuit synthesis. *IEEE Trans. CAD*, 22, 6 (June 2003), 710-722.

[17] Sasao, T., and Fujita, M. (eds.) *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996.