Ordering Techniques for ESOP-Based Toffoli Cascade Generation

J. E. Rice and N. M. Nayeem
Dept. of Math & Computer Science
University of Lethbridge
Lethbridge, AB, Canada
{j.rice, noor.nayeem}@uleth.ca

Abstract

This paper describes three techniques for ordering ESOP cubes prior to generation of a Toffoli gate generation. Two of these techniques are from earlier work, while the third is a new approach. The new approach both reorders the cubes and manipulates them to reduce their size and complexity. Our experiments demonstrate that the new approach is much more effective than either of the two previous approaches. We apply template matching as a post-processing step, which results in even further reductions in the number of Toffoli gates.

1. Introduction

The motivation behind reversible logic is that the laws of physics limit the energy efficiency of traditional irreversible logic, a limit which is likely to be reached within the next 10 to 20 years. In order to advance beyond this point circuits will require an increasing amount of reversibility in their design. It was demonstrated as early as the 1960s [7] that reversible logic would be necessary for lower power dissipation in circuits. Reversible logic also has connections to quantum computing, and in fact reversible circuits can be viewed as a special case of quantum circuits [12]. One author notes that "... logic synthesis for classical reversible circuits is a first step toward synthesis of quantum circuits" [15]. Synthesis techniques for traditional irreversible logic are of little use when the goal is a reversible cascade of gates, and thus many researchers are investigating new techniques in this area.

This paper compares the results of using different cost metric functions for synthesizing Toffoli gate networks from an initial exclusive-or sum-of-products (ESOP) representation of a Boolean function as well as presenting a new method. The initial function may be reversible or irreversible; our technique ensures that the final cascade is reversible. Since the use of templates is a well-known reversible logic technique that requires the circuit to already be in reversible form, we compare results of our cost metrics both before and after applying templates.

2. Background

2.1. ESOP Synthesis

An n-bit generalized Toffoli gate is a reversible logic gate that has n inputs and n outputs and is described as $(x_1, x_2, \ldots, x_n) \rightarrow (x_1, x_2, \ldots, (x_1 \cdot x_2 \cdots x_{n-1}) \oplus x_n)$. A NOT gate is a special case of a Toffoli gate where $n = 1$.

An exclusive-or sum-of-products (ESOP) representation of a switching function is similar to the standard sum-of-products representation except that ORs are replaced with XORs. An ESOP can be used to represent any Boolean function [16], and a gate-level model of a reversible circuit may be generated from an ESOP by simply replacing each cube with a Toffoli gate. The straightforward method of replacing cubes with Toffoli gates would require that the circuit have $2n + m$ input lines, or qubits;[1] one for each inverted literal, one for each non-inverted literal, and one for each output. To reduce this to $n + m$ we use a NOT gate to invert the line when needed; however with a poor ordering of the cubes this can result in an unnecessary number of gates. It is therefore important to use some technique to find a "good" ordering.

For the first two ordering methods in this work a cost metric is calculated for each input variable. The input variable with the lowest cost is chosen, and the ESOP cubes are sorted such that all cubes containing the non-negated form of the selected variable appear before all cubes containing its negated form. This allows a single NOT gate to divide the negated and non-negated forms. Where the variable in a cube is a don't-care (-) the cube is placed in the non-negated list. The process is continued for the negated and non-negated lists until all variables have been selected.

[1] The technically correct term is, in this area, qubits, since we assume some future quantum implementation. The reader is referred to [12] for details on quantum computing.

2.2. The Cost Metrics

2.2.1 Alpha-Beta Cost Metric

The following technique was introduced in [3].

$$\text{cost}_v = \alpha \frac{1}{\sum |v_i|} + \beta \left| \sum v_i \right| \tag{1}$$

$$\text{where} \begin{cases} v \text{ in cube}_i \begin{cases} v \text{ is positive } v_i = 1 \\ v_i = -1 \end{cases} \\ v_i = 0 \end{cases}$$

The cost metric described above is computed for each variable v in turn. The frequency sum (which is multiplied by α) determines how often the variable is used in one form or another over the entire function, while the balance sum (which is multiplied by β) determines the absolute value of the difference between the number of 0s and the number of 1s assigned to that variable. Thus if a variable appears in the function relatively rarely and has close to an equal number of 0s and 1s its cost is computed to be lower. Further details of this technique are given in [3] and [13].

2.2.2 AC Cost Metric

The autocorrelation (AC) transform, based on Equation 2, can be used to transform a function from the traditional Boolean domain to the spectral domain. The resulting data can be used to perform an analysis of the relative dependency of the function on its variables.

$$B^{fg}(\tau) = \sum_{v=0}^{2^n-1} f(v) \cdot g(v \oplus \tau) \tag{2}$$

The AC transform is obtained when f and g in Equation 2 are the same function. Applying the AC transform provides a value of how a function compares with itself at a "shift" of some value, given by τ. This shift corresponds in effect to inverting the inputs corresponding to values of 1 in the binary expansion of τ. Of the resulting AC coefficients, only first-order coefficients (where a single variable is inverted) are used in this cost metric as it allows a measurement of the function's dependency on that variable. Theoretically, a higher value for a coefficient indicates less of a dependency, thus a variable with a low coefficient should be chosen first for generating an ordering. An example is shown in Figure 1. Coefficients for multiple-output functions can be computed by applying the AC transform to each output separately and then combining the results for all outputs. Further details and examples are given in [14].

$x_3x_2x_1$	$f(X)$
000	0
001	1
010	0
011	1
100	0
101	1
110	1
111	1

(a) Truth table.

τ	$B(\tau)$
000	5
001	2
010	4
011	
100	4
101	
110	
111	

(b) First order AC coeffs.

Figure 1: The AC coeffs for $f(X) = x_1 \oplus x_3x_2\overline{x}_1$.

2.3. Related Work

There are a variety of synthesis techniques for reversible logic in the literature, for instance [15, 9, 4] and [6]. We briefly mention a few of these techniques which use an ESOP or similar representation.

Gupta *et al.* [4] present a reversible logic synthesis technique based on a related representation, the positive-polarity Reed-Muller (PPRM) expansion. This work uses a tree structure to investigate all possible factors of each term, allowing the construction of a circuit that shares factors. However the PPRM representation is a special type of ESOP with a more rigorous definition, and thus will almost always have more terms than the ESOP representation used in our work.

The techniques suggested by Perkowski *et al.* in [6] and an earlier work [11] also have some relation to the general ESOP-based approach. The more recent of these works requires a factorization of each of the ESOPs representing the multiple outputs, and a new class of reversible gates is introduced. This method reported achieving good results in terms of gate numbers, but appears to require a large number of garbage outputs.

More recently Hamza and Dueck [5] have suggested a linear programming approach to ESOP cube ordering. Further comparisons to this work are given in Section 5.

2.4. Template Optimization

One of our investigations involved the application of templates in order to further optimize the circuit sizes. According to [2] a template is used to replace a sequence of gates within a Toffoli network with a different sequence of gates without altering a newtork's function. In addition the replacement sequence should be smaller than the replaced sequence. Due to lack of space we direct the reader to [2] for further details.

3. Proposed New Method

The previous two ordering methods for ordering ESOP cubes in preparation for generating a Toffoli-gate cascade can be improved upon, and in this Section we propose a new method. This method has two steps: in the first step a number of rules are applied to the ESOP cube-list, and in the second step the list is then ordered to minimize the number of NOT gates. Unlike the previous two methods, in which the only improvement can be reduction of the number of NOT gates, the purpose of the following rules is to reduce the both the number of NOT gates and/or the size of the Toffoli gates.

3.1. Rules

For Rules 1 and 2 we consider the following cube:

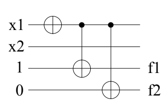
$$x_1x_2\ldots x_n \rightarrow f_1f_2\ldots f_m \tag{3}$$

- **Rule 1** If the cube contains only don't care values, *i.e.* $x_1 = x_2 = \ldots = x_n = -$, then for every output $f_p = 1$, where $p \in \{1, 2, \ldots, m\}$, the qubit of the corresponding output line is inverted and the cube is removed from the cube-list. Usually, output lines are initialized with a 0 qubit. The purpose of this rule is to remove the NOT gates required for this cube and complement the initial values of the corresponding output lines.

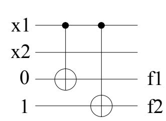
- **Rule 2** If $x_i = 0$ and $x_k = -, \forall k \in \{1, 2, \ldots, n\} - \{i\}$, then for every output $f_p = 1$ where $p \in \{1, 2, \ldots, m\}$, the qubit of the corresponding output line is inverted and x_i is set to 1. The purpose of this rule is to complement x_i.

x_1	x_2	f_1	f_2
-	-	1	0
0	-	1	1

(a) Cube-list



(b) After applying Rule 1.



(c) After applying Rules 1 and 2.

Figure 2: An example of applying Rules 1 and 2 to generate a Toffoli gate cascade from an ESOP cube-list.

An example is shown in Figure 2.

The following illustrates the labeling used for explanation of the next four rules:

Cube1 $x_1x_2\ldots x_{i-1}x_ix_{i+1}\ldots x_{j-1}x_jx_{j+1}\ldots x_n$
$\rightarrow f_1f_2\ldots f_{p-1}f_pf_{p+1}\ldots f_m$
Cube2 $y_1y_2\ldots y_{i-1}y_iy_{i+1}\ldots y_{j-1}y_jy_{j+1}\ldots y_n$
$\rightarrow g_1g_2\ldots g_{p-1}g_pg_{p+1}\ldots g_m$

- **Rule 3** If $x_i = 0$, $x_j = -$, $y_i = -$, $y_j = 0$, $x_k = y_k$, $\forall k \in \{1, 2, \ldots, n\} - \{i, j\}$, and $f_p = g_p = 1$ for any $p \in \{1, 2, \ldots, m\}$, then Cube1 and Cube2 can be transformed into the following four cubes.

  Cube1′ $x_1x_2\ldots x_{i-1}\,1\,x_{i+1}\ldots x_{j-1} - x_{j+1}\ldots x_n$
  $\rightarrow 00\ldots010\ldots0\ (f_q = 0\ \forall\, q \neq p)$
  Cube2′ $y_1y_2\ldots y_{i-1} - y_{i+1}\ldots y_{j-1}\,1\,y_{j+1}\ldots y_n$
  $\rightarrow 00\ldots010\ldots0\ (g_q = 0\ \forall\, q \neq p)$
  Cube3 $x_1x_2\ldots x_{i-1}\,0\,x_{i+1}\ldots x_{j-1} - x_{j+1}\ldots x_n$
  $\rightarrow f_1f_2\ldots f_{p-1}\,0\,f_{p+1}\ldots f_m$
  Cube4 $y_1y_2\ldots y_{i-1} - y_{i+1}\ldots y_{j-1}\,0\,y_{j+1}\ldots y_n$
  $\rightarrow g_1g_2\ldots g_{p-1}\,0\,g_{p+1}\ldots g_m$

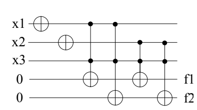
The objective of applying this rule is to complement both x_i and y_j. We note that this rule can create two more cubes, however if $f_q = g_q = 0\ \forall q \in \{1, 2, \ldots, m\} - \{p\}$ then the last two cubes are not required since the output parts of these cubes are all zero. In order to avoid creating any new cubes unnecessarily this rule is only applied when $f_p = g_p\ \forall p \in \{1, 2, \ldots, m\}$. An example is shown

x_1	x_2	x_3	f_1	f_2
0	-	1	1	1
-	0	1	1	1

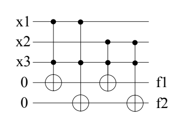
(a) The initial cube-list.

x_1	x_2	x_3	f_1	f_2
1	-	1	1	1
-	1	1	1	1

(b) The result of applying Rule 3.



(c) Circuit generated by creating Toffoli gates for each cube in Figure 3a.



(d) Toffoli cascade for the cube-list in Figure 3b.

Figure 3: Example of applying Rule 3.

in Figure 3.

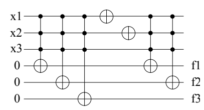
- **Rule 4** [1] If $x_i = 1$, $x_j = 1$, $y_i = 0$, $y_j = 0$, and $x_k = y_k\ \forall k \in \{1, 2, \ldots, n\} - \{i, j\}$, and $f_p = g_p = 1$ for any $p \in \{1, 2, \ldots, m\}$, then Cube1 and Cube2 can be transformed into the following five cubes.

  Cube1′ $x_1x_2\ldots x_{i-1}\,1\,x_{i+1}\ldots x_{j-1} - x_{j+1}\ldots x_n$
  $\rightarrow 00\ldots010\ldots0\ (f_q = 0\ \forall\, q \neq p)$
  Cube2′ $y_1y_2\ldots y_{i-1} - y_{i+1}\ldots y_{j-1}\,1\,y_{j+1}\ldots y_n$
  $\rightarrow 00\ldots010\ldots0\ (g_q = 0\ \forall\, q \neq p)$
  Cube3 $y_1y_2\ldots y_{i-1} - y_{i+1}\ldots y_{j-1} - y_{j+1}\ldots y_n$
  $\rightarrow 00\ldots010\ldots0\ (g_q = 0\ \forall\, q \neq p)$
  Cube4 $x_1x_2\ldots x_{i-1}\,1\,x_{i+1}\ldots x_{j-1}\,1\,x_{j+1}\ldots x_n$
  $\rightarrow f_1f_2\ldots f_{p-1}\,0\,f_{p+1}\ldots f_m$
  Cube5 $y_1y_2\ldots y_{i-1}\,0\,y_{i+1}\ldots y_{j-1}\,0\,y_{j+1}\ldots y_n$
  $\rightarrow g_1g_2\ldots g_{p-1}\,0\,g_{p+1}\ldots g_m$

The purpose of this rule is to reduce the size of the Toffoli gates when transforming the cubes into gates. This rule can create three extra cubes, which is not at all desirable. Like Rule 3, if $f_q = g_q = 0\ \forall q \in \{1, 2, \ldots, m\} - \{p\}$, then the last two cubes are not required. For example, we can apply Rule 4 to the cubes shown in Figure 4a. Only outputs f_1 and f_2 contain both cubes. As the output f_3 does not include the second cube, application of this rule will generate two extra cubes as shown in Figure 4c.

x_1	x_2	x_3	f_1	f_2	f_3
1	1	1	1	1	1
0	0	1	1	1	0

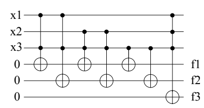
(a) An initial cube-list.



(b) Circuit generated by creating Toffoli gates for each cube in Figure 4a.

x_1	x_2	x_3	f_1	f_2	f_3
1	-	1	1	1	0
-	1	1	1	1	0
-	-	1	1	1	0
1	1	1	0	0	1

(c) A cube-list equivalent to Figure 4a.



(d) Toffoli cascade for the cube-list in Figure 4c.

Figure 4: Example of applying Rule 4.

- **Rule 5** If $x_i = 1$ or 0, $y_i = -$ and $x_k = y_k$ $\forall k \in \{1, 2, \ldots, n\} - \{i\}$, and $f_p = g_p = 1$ for any $p \in \{1, 2, \ldots, m\}$, then Cube1 and Cube2 can be transformed into the following three cubes.

Cube1′	$x_1x_2 \ldots x_{i-1}\overline{x}_i x_{i+1} \ldots x_{j-1}x_jx_{j+1} \ldots x_n$ $\rightarrow 00 \ldots 010 \ldots 0\ (f_q = 0\ \forall\ q \neq p)$
Cube3	$x_1x_2 \ldots x_{i-1}x_ix_{i+1} \ldots x_{j-1}x_jx_{j+1} \ldots x_n$ $\rightarrow f_1f_2 \ldots f_{p-1}\ 0\ f_{p+1} \ldots f_m$
Cube4	$y_1y_2 \ldots y_{i-1}\ -\ y_{i+1} \ldots y_{j-1}y_jy_{j+1} \ldots y_n$ $\rightarrow g_1g_2 \ldots g_{p-1}\ 0\ g_{p+1} \ldots g_m$

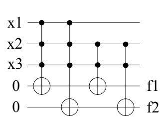
This rule eliminates one cube but may create two extra cubes. If $f_q = g_q = 0\ \forall q \in \{1, 2, \ldots, m\} - \{p\}$, then the latter two cubes are not generated. Similar to Rule 4, this rule helps reduce the size of required Toffoli gates. For example, consider the cube-list given in Figure 5a. We apply Rule 5 to generate the modified cube-list as shown in Figure 5b, which consists of only one cube. Since two functions f_1 and f_2 have both cubes, one cube from the original cube-list (Figure 5a) can be removed and no more cubes are generated.

x_1	x_2	x_3	f_1	f_2
1	1	1	1	1
-	1	1	1	1

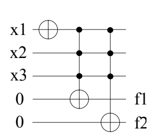
(a) An example cube-list.

x_1	x_2	x_3	f_1	f_2
0	1	1	1	1

(b) The result of applying Rule 5.



(c) Circuit for Figure 5a.



(d) Circuit for Figure 5b.

Figure 5: Example of applying Rule 5.

Applying the rules in different order can change the results. We have found through experimentation that applying the rules in the order 4, 5, 1, 2 and 3 produces better results for a number of benchmark circuits. Further investigation and experimentation into this is required.

3.2. Ordering

After applying the rules, a greedy approach is used to reorder the cubes. All the input variables initially have positive polarity. A sequence with the length of number of input variables is used to determine the current polarity of each variable. This sequence is initialized with all 1s and changed after reordering each cube (adding a NOT gate) which alters the polarity. The algorithm begins with the cube which has the least number of 0s *i.e.*, the highest number of 1s and don't care values. The sequence is then updated, and the distance between this sequence and each of the remaining cubes is calculated. The term distance refers to the Hamming Distance, or the count of the number of variables with positive polarity in one cube and negative polarity in the other. The cubes with distance zero are reordered without changing the sequence, and hence no more NOTs are needed for these cubes. Next the cube with the minimum positive distance is considered due to the requirement of the fewest possible NOT gates when this cube will be transformed into a Toffoli gate. If two or more cubes have the same distance then the cube with more don't care values is selected. The process is repeated until all cubes are processed.

4. Experiments

The following process was used to generate the final benchmark test results: 1. generate an ESOP representation for each benchmark using EXORCISM-4 [10]; 2. generate a Toffoli gate cascade using each of the different ordering and/or cube manipulation approaches; 3. convert the synthesized circuits into a form usable by the template optimization algorithm; and 4. perform template optimization on each of the reformatted synthesized circuits. Comparisons among the approaches were performed both before and after templates were applied. All experiments were performed on a 3.00GHz Intel(R) Pentium(R) 4 machine with 1GB RAM running CentOS release 5.3. The benchmarks used are listed in Table 2.

	avg % reduction gate count	num. of smaller circuits generated
AB vs AC	18%	56 out of 77
AB+t vs AC+t	2%	47 out of 77
NM vs AB	91%	70 out of 77
NM vs AB+t	11%	61 out of 77
NM+t vs AB+t	14%	63 out of 77

Table 1: Comparisons between the various ESOP ordering and optimization methods.

5. Results & Discussion

The AB and AC approaches were first compared in [14], although this previous work did not report comparisons after applying template optimization. Our tests show that even with the application of templates the AB cost metric still outperforms the AC method.

The proposed new method (NM) of sorting ESOP cubes prior to generating a Toffoli gate cascade proved to be very effective. We have restricted our comparisons to the AB cost metric method, as that was the better of the prior two methods tested. For these comparisons we use the best result generated by all values of alpha for each of the 77 benchmarks. Without applying template optimization, the NM resulted in a 90% average reduction in gates, and generated a smaller circuit on 70 out of 77 of the benchmarks tested. After applying template optimization to both methods we still obtained improvements of an average 81% reduction in gates and a smaller circuit on 63 out of 77 benchmarks. Table 1 summarizes the results. The notation "+t" indicates that template optimization was used as a post-processing step. It is interesting to note that template optimization is able to reduce 67 of the benchmarks by an average of 19 gates when applied after the AB method, while for the new method, template optimization only improves the resulting circuit for 40 benchmarks, and for those 40 there is an average reduction of 5 gates.

We briefly discuss our work in comparison to the work by Dueck and Hamza [5] which also looks at a more effective ordering approach combined with attempting to reduce cube sizes. The most significant difference lies in our pre-application of rules to minimize the number and sizes of the Toffoli gates, which does not take place in the Dueck and Hamza approach. It is difficult to make accurate comparions between their work and ours as there is a relatively small overlap of benchmarks, and we also suspect that benchmarks that appear to be the same are not necessarily so. Future work will involve generating a common list of benchmarks, likely from the Revlib library [17], in order to carry out a thorough comparison between our two efforts. In order for the readers to examine our results for themselves we have included in Table 2 the list of benchmarks used in this work, as well as both gate count and quantum cost results. The quantum costs are based on values for Toffoli gates as given in [8];

6. Conclusions and Future Work

This paper introduces a new technique for manipulating and ordering ESOP cubes prior to generating a cascade of Toffoli gates from the ESOP list. In addition we apply template optimization to results from our new technique as well as two previously published techniques, in order to determine whether any technique might be better or worse suited to post-application of template optimization. We found that the AC cost metric is generally the least effective, while the AB cost metric with $\alpha = 0$ was more effective. Application of templates improved the results from both of these approaches, but we still found the AB results to be better than the AC results. However, the proposed new rule-based method achieved better results than even of the two previously introduced methods, before and after templates were applied. Work on the new proposed method is still in its preliminary phases, and can benefit from further experimentation and in particular comparisons to results such as those reported by [4, 6] and in particular [5].

Acknowledgment

This research was funded by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors would like to thank J. Zaretski for his aid in generating some of the data for this paper.

References

[1] D. Brand and T. Sasao. Minimization of AND-EXOR expressions using rewrite rules. *IEEE Transactions on Computers*, 42(5):568–576, 1993.

[2] G. W. Dueck, D. M. Miller, and D. Maslov. Toffoli network synthesis with templates. *IEEE Tranactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):807–817, June 2005.

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

[4] P. Gupta, A. Agrawal, and N. K. Jha. An algorithm for synthesis of reversible logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(11):2317–2330, Nov. 2006.

[5] Z. Hamza and G. W. Dueck. Near-optimal ordering of ESOP cubes for Toffoli networks. In *Proceedings of the 2nd Annual Workshop on Reversible Computation (RC)*, 2010. July 2–3 Bremen, Germany.

Function	inputs	AC	AB	NM	
		GC	GC	GC	QC
ex2	6	12	13	11	160
ex3	6	7	7	7	97
majority	6	8	8	8	125
xor5	6	7	7	5	5
C17	7	9	9	9	105
cm82a	8	17	22	19	143
f2	8	22	19	20	246
rd53	8	24	27	27	269
con1	9	19	21	23	206
9sym	10	132	129	111	6501
9symml	10	132	129	111	6501
life	10	114	107	92	3711
life_min	10	114	107	92	3711
max46	10	115	107	89	4968
rd73	10	85	80	73	1150
sqn	10	91	76	63	1675
dc1	11	36	39	37	454
sym10	11	197	193	162	10358
wim	11	30	25	20	218
z4	11	50	48	48	642
z4ml	11	50	48	48	642
cm152a	12	15	16	15	215
rd84	12	134	111	99	2558
sqrt8	12	47	40	38	616
adr4	13	46	55	34	630
dist	13	237	185	170	7210
radd	13	36	48	51	669
root	13	142	99	88	3393
squar5	13	30	43	41	465
clip	14	195	174	148	6570
cm42a	14	35	35	20	270
cm85a	14	75	69	64	2222
pm1	14	35	35	20	270
sao2	14	129	88	77	7495
co14	15	30	30	30	3488
dc2	15	103	75	65	1778
misex1	15	63	55	55	1012
alu2	16	184	157	136	5118
example2	16	184	157	136	5118
inc	16	118	93	84	1969
mlp4	16	151	131	122	3500
5xp1	17	102	85	75	1229
parity	17	32	32	16	16
ryy6	17	44	44	44	4298
t481	17	21	21	20	236
x2	17	49	38	30	576
alu3	18	96	94	81	2346
dk27	18	31	24	24	252
sqr6	17	66	81	69	1030
add6	19	146	229	197	5757
alu1	20	37	32	29	239
cmb	20	40	18	16	908
ex1010	20	2286	2611	1965	126108
C7552	21	47	80	31	673
decod	21	47	80	31	673
dk17	21	57	49	45	1465
pcler8	21	31	22	19	340
alu4	22	1414	1063	842	47844
apla	22	104	80	70	2884
cm150a	22	53	53	49	833
f51m	22	877	663	555	34084
mux	22	35	35	31	815
tial	22	1426	1041	833	49189
b12	24	74	62	55	1259
cordic	25	3045	2533	1790	348779
cu	25	49	40	39	964
gary	26	503	338	285	17407
in0	26	503	338	285	17407
pcle	26	23	24	23	653
apex4	27	2018	5376	4037	187678
cm151a	28	29	33	33	888
misex3	28	2025	1752	1418	108932
misex3c	28	797	1721	1368	102617
table3	28	1742	1012	744	63286
cm163a	29	43	39	30	732
in2	29	516	405	323	20098
frg1	31	305	212	200	15085

Table 2: Benchmarks used, along with results from each of the AC, AB, and NM approaches. All results are after template matching.

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

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

[8] D. Maslov and G. W. Dueck. Improved quantum cost for n-bit Toffoli gates. *Electronics Letters*, 39(25):1790–1791, Dec 2003.

[9] D. M. Miller, D. Maslov, and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In *DAC '03: Proceedings of the 40th Cconference on Design Automation*, pages 318–323, New York, NY, USA, 2003. ACM.

[10] A. Mishchenko and M. Perkowski. Fast heuristic minimization of exclusive sum-of-products. In *Proceedings of the 5th International Reed-Muller Workshop (RM)*, pages 242–250, 2001. August 10–11, Starkville, Mississippi.

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

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

[13] J. E. Rice, K. B. Fazel, M. A. Thornton, and K. B. Kent. Toffoli gate cascade generation using ESOP minimization and QMDD-based swapping. In *Proceedings of the International Symposium on Representations and Methodology of Future Computing Technologies (RM)*, pages 63–71, 2009.

[14] J. E. Rice and V. Suen. Using autocorrelation coefficients-based cost functions in ESOP-based Toffoli gate cascade generation. In *Proceedings of the 23rd IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6, 2010. May 2–5, Calgary, Canada.

[15] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes. Synthesis of reversible logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(6):710–722, June 2003.

[16] B. Steinbach and A. Mishchenko. SNF: a special normal form for ESOPs. In *Proceedings of the International Symposium on Representations and Methodology of Future Computing (RM)*, pages 66–81, 2001.

[17] R. Wille, D. Große, L. Teuver, G. W. Dueck, and R. Drechsler. Revlib: an online resources for reversible functions and reversible circuits. In *Proceedings of the 38th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 220–225, 2008. May 22–23, Dallas, Texas.