USING AUTOCORRELATION COEFFICIENT-BASED COST FUNCTIONS IN ESOP-BASED TOFFOLOI GATE CASCADE GENERATION

J. E. Rice[*]

University of Lethbridge
Department of Mathematics & Computer Science
Lethbridge, Alberta, Canada

V. Suen

University of Toronto
Faculty of Computer Science
Toronto, ON, Canada

ABSTRACT

A revision of an ESOP-based Toffoli gate cascade synthesis technique [1] is presented. The cost metric used previously was replaced with a new cost metric based on autocorrelation coefficients to determine the placements of Toffoli gates. The algorithm remains capable of generating reversible circuits for large functions within reasonable time.

Index Terms— reversible logic, logic synthesis, ESOP, toffoli gates

1. INTRODUCTION

Moore's law states that the number of transistors on a chip doubles every two years [2]. Holding true to this law, technologies have been getting smaller yet increasing in performance over the last few decades. This trend, however, must stop eventually when components reach their physical shrinking limits. In addition, Landauer's principle states that each bit of information lost must dissipate a certain amount of energy [3]. In other words, there is a minimum energy cost for current irreversible logic operations [4]. Some researchers believe that computing may reach this bound, and computing may stop improving by as early as 2015 [4]. As we approach both these limits, new ideas must be considered in order to continue improving computer performance.

Reversible computing is one solution to consider. Bennett showed that circuits must be reversible in order to not dissipate any power [5]. In fact, it is theoretically possible for reversible logic operations to re-use up to 100% of the energy that an irreversible logic operation will otherwise dissipate [3], meaning that the lower bound on energy cost would no longer apply. Not only does reversible computing have this interesting property, but it is also linked with quantum computing since all quantum gates are also reversible [6].

This paper revisits the exclusive-or sum-of-products (ESOP) based toffoli gate cascade generation technique described in [1]. This technique synthesizes functions in ESOP representation into a cascade of reversible Toffoli gates. However, rather than using the alpha/beta cost metric to optimize the number of signal lines, an autocorrelation coefficient-based cost function was used instead. The purpose of this work was to determine whether or not this new cost metric could produce more optimal circuits than the previous method.

The next section provides the necessary background on reversible logic, ESOP representation, and autocorrelation coefficients, finishing off with an overview of related work. We then explain briefly the ESOP mapping method, followed by an explanation of the changes made from [1]. The following section describes our experimental results and findings. The last section discusses our conclusions and suggestions for future work.

2. BACKGROUND

2.1. Reversible Logic

A logical gate or function is considered reversible if and only if it is bijective (one-to-one and onto) [7]. In other words, for each n-bit input, there is a unique corresponding n-bit output. The traditional NOT gate is reversible, but the traditional AND gate is not, as shown by their truth tables in Figure 1.

x	$f(x)$
0	1
1	0

(a)

$x\ y$	$f(x,y)$
00	0
01	0
10	0
11	1

(b)

Fig. 1. (a) NOT gate (reversible) and (b) AND gate (non-reversible)

Various reversible gates exist, but only NOT and Toffoli gates will be used and discussed in this paper. An $n \times n$ Toffoli gate inverts the nth "target" line if and only if the other $n-1$ "control" lines are 1 [8]. The control lines remain unchanged. The NOT gate is a special case of the Toffoli gate in which there are no control lines. The NOT gate can also be described as $(x) \rightarrow (x \oplus 1)$. Likewise, the behavior of

[*]Dr. Rice would like to thank NSERC and the CDMP for their support of this work.

an n-input Toffoli gate can be written as $(x_1, x_2, ..., x_n) \rightarrow (x_1, x_2, ..., x_{n-1}, x_1x_2...x_{n-1} \oplus x_n)$ [1]. Figure 2 shows the NOT gate, a Toffoli gate with one control line, and a Toffoli gate with $n-1$ control lines.

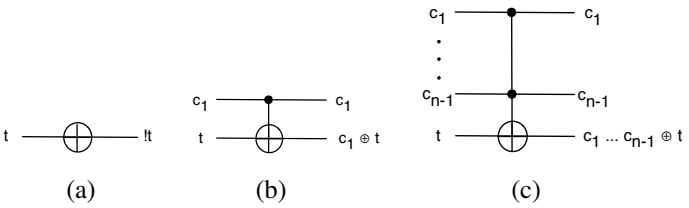

Fig. 2. (a) NOT gate (b) TOF2 gate (c) TOFN gate

Because feedback and fanout are restricted when synthesizing reversible circuits, generated circuits are produced with the cascade structure [8]. Control lines run from left to right, and gates are placed in sequence one after another along the signal lines, creating a cascade of gates.

2.2. ESOP Representation

Exclusive-or sum-of-products (ESOP) representations are very similar to the more traditional sum-of-products (SOP) representations as they are both used in traditional logic synthesis [1]. Given $f(a, b, c) = ab + ac$ in SOP form, replacing $+$ (or) with $\oplus$ (exclusive-or) will give a new function $g(a, b, c) = ab \oplus ac$, which is in ESOP form. Like the or-operator, the exor-operator is also associative.

2.3. Autocorrelation Transform

The autocorrelation transform is used to transform a function to the spectral domain [9]. The correlation transform is defined as [10]

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

When f and g are equal functions, Equation (1) becomes the autocorrelation transform. The superscripts are usually omitted when f and g are the same function. Applying the autocorrelation transform compares a function with itself, shifted by an amount, τ [10]. The results of the applications are called the autocorrelation coefficients of the function [9]. For this study we use only the first order coefficients; that is, the binary value of τ has one and only one number of ones. These coefficients are of interest because they give a measure of how dependent a function is on a particular variable. For example, assuming a variable ordering of xyz, computing $B(001)$ would give the measure for z. In theory, the higher the coefficient, the less dependent the function is on a variable. Likewise, the lower the coefficient, the more dependent it is [9].

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

Table 1. Truth table for $f(X) = x_1x_2 + x_3$

Below is an example of how $B(\tau)$ is computed given the function $f(X) = x_1x_2 + x_3$ and $\tau = 100$. The truth table is shown in Table 1.

$$\begin{aligned} B(001) &= \sum_{v=0}^{2^n-1} f(v) \cdot f(v \oplus 001) \\ &= [f(000) \cdot f(000 \oplus 001)] + ... \\ &\quad +[f(111) \cdot f(111 \oplus 001)] \\ &= [f(000) \cdot f(001)] + ... + [f(111) \cdot f(110)] \\ &= [0 \cdot 1] + [1 \cdot 0] + [0 \cdot 1] + [1 \cdot 0] + [0 \cdot 1] \\ &\quad +[1 \cdot 0] + [1 \cdot 1] + [1 \cdot 1] \\ &= 0 + 0 + 0 + 0 + 0 + 0 + 1 + 1 \\ &= 2 \end{aligned}$$

It is also possible to calculate autocorrelation coefficients for multiple output functions by combining the autocorrelation function for each individual function into the total autocorrelation function [9].

2.4. Related Work

There are a variety of synthesis techniques for reversible logic in the literature, for instance [7, 8, 11, 12, 13] and [14], to name just a few. Here we briefly describe some such techniques which also utilize an ESOP or similar representation. Gupta *et al.* [12] present a reversible logic synthesis technique based on a related representation, the positive- polarity Reed-Muller (PPRM) expansion. They utilize a tree structure to enable investigation of all possible factors of each term, rather than use an approach requiring one gate for each term in the expansion. This allows the construction of a circuit that shares factors. The PPRM representation, while canonical, 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. Maslov and Dueck have also conducted an analytical comparison of their technique and EXOR PLAs, a structure that can implement ESOP representations [13]. They find that their reversible cascade with minimal garbage (RCMG) model compares favorably to an EXOR PLA, particularly for one class of functions for which

the ESOP representation has exponential complexity. However, their analysis is based entirely on circuit complexity. Finally, Perkowski and other researchers have also looked into using ESOPs as a starting point for reversible logic synthesis [14]. In [14] a new class of reversible gates is introduced, allowing modification of two qubits, but requiring a significantly higher level of complexity. The technique in [14] also requires a factorization of each of the ESOPs representing the multiple outputs.

3. ESOP MAPPING METHOD

The basic algorithm for generating a cascade of Toffoli gates from an ESOP representation remains unchanged from [1]. We assume a function is given in an ESOP cube-list representation. $2n$+m signal lines are required in the final circuit, where n is the number of inputs and m is the number of outputs. $2n$ lines are required for the inputs and their inverse. Then, for each cube in the cube-list for each output, a Toffoli gate is generated [1]. Figure 3, taken from [1], outlines the described algorithm.

```
basicCascadeGen(esop)
  cascade.toffoliList = empty;
  //create signals
  foreach i in esop.inputs
    cascade.addQubit(i, positive);
    cascade.addQubit(i, negative);
  foreach o in esop.outputs
    //add a constant 0 qubit for each output
    cascade.addQubit(o, constant 0)
  //create TOF gates
  foreach c in esop.cubes
    foreach o in esop.outputs
      If c in onset(o)
        //add a toffoli gate
        t = new ToffoliGate
        t.target = cascade.getQubit(output)
        foreach literal in c
          t.addControl(cascade.getQubit(literal))
        cascade.addToffoli(t)
```

Fig. 3. Algorithm for Mapping ESOP Representations to a Cascade of Toffoli Gates

4. OPTIMIZATION

As discussed in the previous section, the basic algorithm produces $2n$ signal lines for the inputs alone. It is likely, however, that a lot of the signal lines, inverses in particular, will never be used. Since it is possible to get the negated input lines by using a NOT gate whenever necessary, we try to order the given cube-list (gates) in such a way that would allow the least number of NOT gates. We are able to move the gates around freely in the cascade because the cube-list is in ESOP representation, which is associative.

In this work, we use and calculate autocorrelation coefficients in our algorithm to determine how a given cube-list should be reordered. The autocorrelation coefficient of each input variable is calculated, and the cube-list is reordered according to the variable with the lowest coefficient. All the cubes with the non-negated form, including don't cares, of the variable are moved to the beginning of the cascade, and all the cubes with the negated form are moved to the end. As stated earlier, the lower the coefficient, the more dependent the function is on that variable [9]. In theory, this certain variable should be dealt with first. The list is then split into two cube-lists. One list contains cubes of the non-negated form of the variable, and the other list contains the cubes with the negated form. Within each of these lists, autocorrelation coefficients are calculated again, and the lists are reordered and split once again according to the new values. The process continues until no more splits can be made.

For example, given an abitrary ESOP cube list, the autocorrelation coefficients were found to be 6, 4, 6, and 6 for a, b, c, and d respectively. Since b has the lowest autocorrelation coefficient, the cube-list is reordered according to b. It is then split into two lists, one with the non-negative values of b and the other with the negative values of b. In each new list, autocorrelation coefficients are recalculated again. More re-orderings and splits are made as necessary. See Figures 4 and 5 for the corresponding cube-lists.

.i 4	
.o 1	
.type esop	
–01 1	
1-0- 1	0001: 6
000- 1	0010: 4*
01-1 1	0100: 6
10-1 1	1000: 6
.e	
(a)	(b)

Fig. 4. (a) ESOP cube-list (b) Autocorrelation Coefficients

.i 4		
.o 1		
.type esop		.i 4
01-1	.i 4	.o 1
10-1	.o 1	.type esop
–01	.type esop	–01
1-0-	01-1	1-0-
000-	10-1	000-
.e	.e	.e
(a)	(b)	(c)

Fig. 5. (a) Reordered ESOP cube-list (b) List 1 (c) List 2

Because the program provided for generating autocorrelation coefficients only works on inputs with a single output, m-output functions are divided into m different cube-lists of one output. Each of these cube-lists go through the same pro-

cedure described above. After cascades are generated for each separate output, necessary NOT gates are added to each cascade to reset signal lines. They are then put in sequence one after another for the final circuit.

Using this method reduces the number of signal lines from $2n$+m to n+m.

```
reorder(cubes, polarity, vars)
  if(cubes.isEmpty || vars.isEmpty)
    return cubes
  bestVar = calcBestVar(cubes, vars)
  {pCubes, nCubes} = split(cubes, bestVar)
  pReorder = reorder(pCubes, positive, vars  bestVar)
  nReorder = reorder(nCubes, negative, vars  bestVar)
  if(polarity==negative)
    nReorderNots = addNots(nReordered, bestVar)
  reorderedCubes = reconnect(pReorder, nReorderNots)
  return reorderedCubes

cascadeGen(esop)
  reorderedCubes = reorder(esop.cubes, positive, esop.vars)
  cascade = convertCubesToToffoli(reorderedCubes)
  removeExtraNots(cascade)
```

Fig. 6. Algorithm for NOT gate insertion

5. EXPERIMENTAL RESULTS

The purpose of this work is to determine if using autocorrelation coefficients in our algorithm can reduce and produce more optimal circuits than the previous method of using the alpha/beta cost metric. We took the implementation of EXORCISM-4 [15] used in [1] and modified the cost function to reorder cube-lists using the lowest autocorrelation coefficient rather than the lowest cost metric. The experiments were run on a 3.00GHz Intel(R) Pentium(R) 4 machine with 1GB RAM running CentOS release 5.3. The benchmarks used are the same as in [16], where the total number of variables are less than or equal to 31. This is because we wanted to use template matching on the resulting cascades as was done in [16], but the process has been omitted in this work.

Table 2 shows the results obtained. The following information is given for each input:

- circuit: circuit name
- num signals: total number of inputs and outputs
- gates $\alpha = 0$: the number of gates in the cascade produced with $\alpha = 0$ in the previous method
- time $\alpha = 0$: the time, to the nearest second, it took to minimize and generate the final circuit using the alpha/beta cost metric
- gates AC: the number of gates in the cascade produced using autocorrelation coefficients
- time AC: the time, to the nearest second, it took to minimize and generate the final circuit using autocorrelation coefficients

Interestingly, 79/79 benchmarks generated cascades successfully with the autocorrelation coefficient method, while only 78/79 cascades were generated successfully using the alpha/beta cost metric. It is uncertain as to why this is the case. Although the autocorrelation coefficient method ran sucessfully on more benchmarks than the alpha/beta method, it is clear that the previous method, on average, generates better results. The alpha/beta method also ran about four times faster than the autocorrelation method, taking a total of about 316 seconds compared to 1318 seconds of the newer method. The following list summarizes our results:

- 14/79 (17.7%) of resulting cascades had the same number of gates
- 9/79 (11.4%) of resulting cascades had less gates than the alpha/beta method
  - 3/9 (33.3%) of these had about half as many gates as the alpha/beta method
  - 1/9 (0.1%) of these only generated a circuit with the AC method and not the alpha/beta method
- 56/79 (70.9%) of resulting cascades had more gates than the alpha/beta method
  - 6/56 (0.1%) of these had about two times as many gates as the alpha/beta method
  - 1/56 (0.01%) of these had over three times as many gates as the alpha/beta method

6. CONCLUSIONS AND FUTURE WORK

Using autocorrelation coefficients to optimize signal lines in the presented ESOP-based algorithm may still be effective. The number of gates in many of our generated cascades are comparable to the cascades generated using the alpha/beta method. It is still possible to optimize the number of gates in the resulting cascades, in fact. Adding NOT gates to reset signal lines is not always necessary. It is possible to check for redundancy and eliminate many of these extra gates, but due to time constraints, we were unable to include it in this work. Other optimization methods such as template matching can also be applied.

Although the synthesis times are a lot slower than using the previous cost metric, it is still manageable. The current implementation was written to be functional only. Efficiency and speed were not considered at all at this point since our aim was to see if using autocorrelation coefficients was even probable. Other future work includes modifying the autocorrelation program to work with multiple output functions. This would prevent the algorithm from running multiple times, and the number of gates could potentially reduce.

circuit	num signals	gates $\alpha = 0$	time $\alpha = 0$	gates AC	time AC
ex1	6	7	0	7	0
ex2	6	14	0	12	1
ex3	6	7	0	7	1
majority	6	8	0	8	0
xor5	6	7	0	7	1
C17	7	11	0	12	1
cm82a	8	25	0	29	1
f2	8	21	1	33	2
rd53	8	27	0	36	1
con1	9	24	0	24	2
9sym	10	147	0	151	5
9symml	10	147	0	151	5
life	10	123	0	132	4
life_min	10	123	0	132	3
max46	10	127	0	127	4
rd73	10	86	0	105	3
sqn	10	82	0	118	4
dc1	11	43	0	69	3
sym10	11	225	1	229	7
wim	11	29	0	49	2
z4	11	55	0	76	3
z4ml	11	55	0	76	3
cm152a	12	19	0	19	1
rd84	12	127	0	168	7
sqrt8	12	43	0	70	3
adr4	13	60	0	54	3
dist	13	196	0	291	9
log8mod	13	N/A	N/A	116	5
radd	13	62	0	40	2
root	13	111	0	196	7
squar5	13	48	1	49	3
clip	14	183	0	230	10
cm42a	14	54	0	67	3
cm85a	14	70	0	91	4
pm1	14	54	0	67	3
sao2	14	104	0	166	8
co14	15	50	0	50	2
dc2	15	83	0	145	7
misex1	15	56	0	114	5
alu2	16	182	0	229	12

Table 2. Table showing gate counts and runtime for previous vs. new technique.

circuit	num signals	gates $\alpha = 0$	time $\alpha = 0$	gates AC	time AC
example2	16	182	0	229	11
inc	16	100	0	174	9
mlp4	16	139	1	191	6
5xp1	17	91	0	133	6
parity	17	32	1	32	6
ryy6	17	44	0	44	2
t481	17	21	0	21	3
x2	17	41	0	74	4
alu3	18	98	0	150	8
dk27	18	25	0	59	5
sqr6	18	90	0	98	4
add6	19	265	1	150	5
alu1	20	36	0	53	4
cmb	20	18	0	42	3
ex1010	20	2768	137	2761	79
C7552	21	92	0	79	5
decod	21	92	0	79	4
dk17	21	54	0	167	9
pcler8	21	22	0	47	5
alu4	22	1194	4	1687	70
apla	22	84	0	243	11
cm150a	22	60	0	60	10
f51m	22	731	4	1075	47
mux	22	42	0	42	35
tial	22	1184	6	1707	74
b12	24	70	0	121	9
cordic	25	2533	24	3522	193
cu	25	50	0	119	8
gary	26	377	0	673	43
in0	26	377	0	673	43
pcle	28	27	0	53	9
apex4	28	5505	105	2449	73
cm151a	28	35	0	67	12
misex3	28	1829	17	2320	95
misex3c	28	1808	12	970	50
table3	28	1062	0	2142	102
cm163a	29	45	1	68	9
in2	29	426	0	695	44
frg1	31	241	0	359	34

Table 3. Continuation of Table 2.

7. REFERENCES

[1] 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)*, 2007, pp. 206–209, Aug. 22–24 2007, Victoria, BC, Canada, IEEE Press.

[2] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, pp. 114–117, 1965.

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

[4] M. P. Frank, "Introduction to reversible computing: Motivation, progress, and challenges," in *Proceedings of the 2nd Conference on Computing Frontiers*, 2005, pp. 385–390, May 4–6, Ischia, Italy, ACM Press.

[5] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 6, pp. 525–532, 1973.

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

[7] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, June 2003.

[8] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *DAC '03: Proceedings of the 40th conference on Design automation*, New York, NY, USA, 2003, pp. 318–323, ACM.

[9] J. E. Rice, "The autocorrelation transform and its application to the classification of Boolean functions," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2009, pp. 94–99.

[10] J. E. Rice and J. C. Muzio, "Properties of autocorrelation coefficients," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2003, pp. 577–580, Victoria, Canada.

[11] J. Donald and N. K. Jha, "Reversible logic synthesis with Fredkin and Peres gates," *Journal of Emerging Technology Computing Systems*, vol. 4, no. 1, pp. 1–19, 2008.

[12] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2317–2330, Nov. 2006.

[13] D. Maslov and G. W. Dueck, "Complexity of reversible toffoli cascades and exor plas," in *Proceedings of the 12th International Workshop on Post-Binary ULSI Systems, Tokyo*, 2003, (downloaded Mar. 2007 from http://www.cs.unb.ca/profs/gdueckreversible/esop.pdf).

[14] 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)*, 2003, pp. 144–153.

[15] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive sum-of-products," in *Proceedings of the 5th International Reed-Muller Workshop*, 2001, pp. 242–250, August 2001, Starkville, Mississippi.

[16] J.E. Rice, K. Fazel, M. Thornton, and K. B. Kent, "Toffoli gate cascade generation using ESOP minimization and QMDD-based swapping," in *Proceedings of the Reed-Muller Workshop (RM2009)*, 2009, pp. 63–72.