Spectral and Two-Place Decomposition Techniques in Reversible Logic

D. Michael Miller

Department of Computer Science
University of Victoria
Victoria, BC, Canada V8w 3P6
mmiller@csr.uvic.ca

ABSTRACT

A digital circuit is reversible if it maps each input vector into a unique output vector. Reversible circuits can lead to low-power CMOS implementations and are also of interest in optical and quantum computing. In this paper, we consider the synthesis of reversible logic assuming primitive reversible devices such as Feynman, Toffoli and Fredkin gates. In particular, we consider the use of Rademacher-Walsh spectral techniques and two-place decompositions of Boolean functions. Preliminary results are given for reversible and nonreversible functions and show that the approaches described do indeed show promise.

1. INTRODUCTION

A digital circuit is reversible if it maps each input pattern to a unique output pattern. Landauer [8] proved that traditional irreversible gates lead to power dissipation in a circuit regardless of its implementation. Bennett [1] showed that for power not to be dissipated it is necessary that the circuit be build from reversible gates. Reversible circuits are of interest because of their potential application in low-power CMOS design, quantum computation and optical computing.

Here we consider the synthesis of a reversible circuit as a composition of reversible logic gates. We do not elaborate on technology issues but do make the following assumptions: (i) fan-out is not permitted; (ii) loops are not permitted; and (iii) permutation of connections between gates is permitted. We consider the use of NOT, Feynman [3], Toffoli [15] and Fredkin [4] gates. The primary interest in this paper is to explore the use of Rademacher-Walsh spectral techniques [5][14], which are known to be quite good at disclosing exclusive-OR structures.

Synthesis of reversible logic is significantly different from conventional logic synthesis. Since loops are not permitted, a reversible logic circuit can be specified as a simple sequence of gates. Further since fan-out is not permitted, and assuming an appropriate technology, a reversible logic circuit can realize the inverse specification simply by applying the gates in the reverse order. Hence, synthesis can be carried out from the inputs toward the outputs or from the outputs toward the inputs. Indeed, as will be shown by example, it is advantageous to synthesize a circuit in both directions and take the simpler result.

2. PRELIMINARIES

2.1 Reversible Logic Gates

A reversible logic gate is a *k*-input, *k*-output (denoted *k***k*) device that maps each possible input pattern to a unique output pattern. For the gates considered, not only is the circuit reversible, the forward and reverse mappings are identical. Many reversible logic gates have been studied in the literature [2][3][4] [7][9][11][15]. Table 1 defines the gates considered here.

Gate Type	Functionality	Gate Notation
1*1 Not	$x' = \overline{x}$	NOT(*x*)
2*2 Feynman [3]	$x' = x$ $y' = x \oplus y$	FEY(*x*,*y*)
3*3 Toffoli [15]	$x' = x$ $y' = y$ $z' = xy \oplus z$	TOF3(*x*,*y*,*z*)
4*4 Toffoli [15]	$w' = w$ $x' = x$ $y' = y$ $z' = wxy \oplus z$	TOF4(*w*,*x*,*y*,*z*)
3*3 Fredkin [4]	$x' = x$ $y' = \overline{x}y \oplus xz$ $z' = \overline{x}z \oplus xy$	FRE(*x*,*y*,*z*)

Table 1: Reversible Logic Gates

The bi-directional nature of these gates is emphasized by using conventional quantum logic notation [10] where the same labels are used on both sides of the gate. One can readily verify each of these gates is reversible. Feynman and Toffoli gates transform a single variable while the Fredkin gate transform a pair

The obvious transformations apply regarding NOT and FEY. Futher:

$$\mathrm{FRE}(x,y,z) = \mathrm{FEY}(y,z)\,\mathrm{TOF3}(x,z,y)\,\mathrm{FEY}(y,z) \quad (1)$$

$$\mathrm{TOF3}(x,y,z) = \mathrm{FEY}(z,y)\,\mathrm{FRE}(x,z,y)\,\mathrm{FEY}(z,y) \quad (2)$$

$$\mathrm{TOF4}(w,x,y,z) = \mathrm{TOF3}(w,x,e)\ \mathrm{TOF3}(y,e,z) \quad (3)$$

where e is a constant 0 on the input side. The operations on the right-hand sides are applied in the order given.

2.2 Rademacher-Walsh Spectral Domain

A completely-specified Boolean function $\mathrm{f}(x_1 ... x_n)$ is defined by a column vector of 2^n 0's and 1's denoted $\mathbf{F}$. The Rademacher-Walsh spectrum [5] of the function is given by

$$\mathbf{R} = \mathbf{T}^n \mathbf{F} \tag{4}$$

where the transform matrix $\mathbf{T}^n$ is a Hadamard matrix defined as

$$\mathbf{T}^0 = [1]$$
$$\mathbf{T}^p = \begin{bmatrix} \mathbf{T}^{p-1} & \mathbf{T}^{p-1} \\ \mathbf{T}^{p-1} & -\mathbf{T}^{p-1} \end{bmatrix} \tag{5}$$

Taking +1 as logic 0 and –1 as logic 1, each row of the transform matrix can be seen to correspond to the truth vector of the exclusive-OR (EXOR) of a subset of $x_1, x_2 ... x_n$. Each spectral coefficient thus measures the correlation of $\mathbf{F}$ to a particular EXOR function and the spectral coefficients are identified by subscripts indicating the variables involved, *e.g.* for $n = 3$,

$$\mathbf{R} = \begin{bmatrix} r_0 & r_1 & r_2 & r_{1,2} & r_3 & r_{1,3} & r_{2,3} & r_{1,2,3} \end{bmatrix} \tag{6}$$

r_0 denotes the EXOR of no variables, *i.e.* the constant 0 function and can be seen to simply count the number of 1's in $\mathbf{F}$. All the other coefficients take values in the range -2^{n-1} denoting perfect agreement with the corresponding EXOR function and $+2^{n-1}$ denoting perfect agreement with the complement of that EXOR function.

Linear input translation [5] and linearization [6] have been studied in the spectral domain. They are implemented suing EXOR operations in Feynman gate structures.

Consider two functions, f and g with spectra $\mathbf{R}^{\mathrm{f}}$ and $\mathbf{R}^{\mathrm{g}}$, respectively. Boolean operations can be performed directly in the spectral domain [5] as shown in Table 1. Using these rules, we can perform the computations for the gates in Table 1 directly in the spectral domain.

NOT	$r_0^{\bar{\mathrm{f}}} = 2^n - r_0^{\mathrm{f}};\ r_v^{\bar{\mathrm{f}}} = -r_v^{\mathrm{f}}\ \forall v \neq 0$
AND	$r_v^{\mathrm{fg}} = \sum_{u=0}^{2^n-1} r_v^{\mathrm{f}} \times r_{v \oplus u}^{\mathrm{g}}\ \forall v$ ($v \oplus u$ is bit-wise $\oplus$)
OR	$\mathbf{R}^{\mathrm{f+g}} = \mathbf{R}^{\mathrm{f}} + \mathbf{R}^{\mathrm{g}} - \mathbf{R}^{\mathrm{fg}}$
EXOR	$\mathbf{R}^{\mathrm{f \oplus g}} = \mathbf{R}^{\mathrm{f}} + \mathbf{R}^{\mathrm{g}} - 2\mathbf{R}^{\mathrm{fg}}$

Table 2 Logic Computations in the Spectral Domain

2.3 Function Complexity

One simple measure of function complexity is a count of the number of adjacent 0's and adjacent 1's on its Karnaugh map [6]. It has been shown [5], that this count is given by

$$C(\mathrm{f}) = \frac{1}{2}\left(n2^n - \frac{1}{2^{n-2}} \sum_{v=0}^{2^n-1} \| v \| \mathbf{r}_v^2 \right) \tag{7}$$

where $\| v \|$ is the number of 1's in the binary expansion of v.

$NZ(\mathbf{R})$, the number of zero coefficients in $\mathbf{R}$, is a simple measure of the complexity of the spectrum of a function but is not a direct measure of function complexity since all EXOR functions, including single variables and their complements, have two nonzero coefficients. However, all such functions are equivalent under linear translation, and hence amenable to implementation using Feynman gates, so $NZ(\mathbf{R})$ is of interest.

In the synthesis method described below, we use the complexity metric

$$D(\mathrm{f}) = n2^n NZ(\mathbf{R}) + C(\mathrm{f}) \tag{8}$$

$D(\mathrm{f})$ gives a higher value the fewer the number of nonzero spectral coefficients, and when that measure is equal, to functions with higher adjacency count. Single variables and their complements yield the maximum value of $D(\mathrm{f})$ for a given n although they are not unique in that regard.

3. SYNTHESIS METHOD

Given a reversible logic specification expressed as a system of Boolean functions $\mathrm{f}_i(x_1, x_2, ..., x_n), 1 \leq i \leq n$, our method first transforms each function to the spectral domain giving the spectra $\mathbf{R}_i, 1 \leq i \leq n$.

The core of our method is the following procedure which identifies a single Feynman gate, or a single Toffoli gate possibly with NOT gates applied to certain inputs. Computations are carried out in the spectral domain using the rules in Table 2.

Procedure

Input: Spectra $\mathbf{R}_i, 1 \leq i \leq n$.

Output: One Feynman gate, or one Toffoli gate (possibly with input NOT gates), and transformed spectra $\mathbf{R}_i', 1 \leq i \leq n$.

Process:

(a) Examine the effect of each of the $n(n-1)$ possible FEY(x,y) gates and select the gate that results in the maximum positive change in $D(\mathrm{f}_i)$ for the variable it affects.

(b) If no gate is identified in (a), examine the effect of each of the $n(n-1)(n-2)/2$ possible TOF3(x,y,z) gates each with the four possible negation patterns for x,y and select the gate and negation pattern that results in the maximum positive change in $D(\mathrm{f}_i)$ for the variable it affects.

(c) If no gate is identified in (b), examine the effect of each of the $n(n-1)(n-2)(n-3)/6$ possible TOF4(*w,x,y,z*) gates each with the eight possible negation patterns for *w,x,y* and select the gate and negation pattern that results in the maximum positive change in $D(\mathrm{f}_i)$ for the variable it affects.

(d) If no gate is identified in (c), the procedure terminates in error. Otherwise, $\mathbf{R}'_j = \mathbf{R}_j, j \neq i$, where x_i is the variable affected by the selected gate. $\mathbf{R}'_i$ is computed based on the gate.

The above procedure is applied iteratively until the problem has been reduced to a set of n spectra each representing a unique variable or its complement. Each complemented variable requires a NOT gate be applied. The final result is a sequence of reversible logic gates which when applied in order transforms each of the initial n spectra to the spectrum of a unique variable. The method thus synthesizes the circuit from the outputs back to the inputs. No back-tracking is employed. As a final step, Fredkin gates can be inserted by applying (1).

Step (a) in the above procedure is in fact single gate linearization as applied in the two-place decomposition method described in [16] [17]. The search process described can thus be replaced by a direct selection based upon spectral coefficient values. We are currently investigating spectral conditions that could similarly replace the search process in steps (b) and (c).

4. EXAMPLES

For each example, the specification is given as an ordered set of decimal numbers which define the truth table specification of the reversible logic problem to be realized. To illustrate, the specification for Example 1 defines a Fredkin gate as specified in Table 3. The circuit is given as an ordered sequence of reversible gates. Read from left to right they transform the left side of the specification to the right side, and *vice versa*.

Example 1: Verification of realizing a Fredkin gate using two Feynman gates and a Toffoli gate.

Specification: [0,1,2,3,4,6,5,7] (corresponds to Table 3)
Circuit: FEY(b,c) TOF3(a,c,b) FEY(b,c)

a b c	a′b′c′
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 1 0
1 1 0	1 0 1
1 1 1	1 1 1

Table 3 Fredkin Gate Specification

Example 2: A second example of the interchange of two positions in the specification. Note the analogous structure to that of the Fredkin gate realization. The right two Feynman gates can be exchanged which better shows the expected symmetry of the solution. After that, the circuit given by our method is identical to a solution provided by Perkowski [12].

Specification: [0,1,2,4,3,5,6,7]
Circuit: FEY(a,c) FEY(a,b) TOF3(b,c,a) FEY(a,c) FEY(a,b)

Example 3: The four input extension of Example 2. Our method requires a TOF4 gate which as specified in (1) can be replaced by two TOF3 gates with a constant input added. The right three FEY gates can be permuted to better show the symmetry.

Specification: [0,1,2,3,4,5,6,8,7,9,10,11,12,13,14,15]
Circuit: FEY(a,d) FEY(a,c) FEY(a,b) TOF4(b,c,d,a) FEY(a,d) FEY(a,c) FEY(a,b)

Example 4: This is increment $\bmod 2^n$ for $n = 3$.

Specification: [1,2,3,4,5,6,7,0]
Circuit: TOF3(b,c,a) NOT(c) NOT(b) FEY(c,b)

Example 5: This is the 4 input extension of Example 4. Note the generalization to the structure of the solution for the $n = 3$ case. Further extension would require higher order TOF gates, or realizations using multiple lower order TOF gates and constant inputs.

Specification: [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0]
Circuit: TOF4(b,c,d,a) TOF3(c,d,b) NOT(d) NOT(c) FEY(d,c)

Example 6: This example is taken from [13]. Note that the inputs must be assigned as specified, *i.e.* input a is connected to a in the given realization, input b is connected to d, *etc*.

Specification: [3,11,2,10,0,7,1,6,15,8,14,9,13,5,12,4]
Assign inputs (a,b,c,d) to (a,d,b,c) in the realization below:
Synthesized circuit: NOT(d) NOT(b) TOF3(b,c,d) NOT(b) TOF3(a,c,d) NOT(c) TOF3(a,b,d) ***FEY(a,c) TOF3(b,c,a)*** FEY(b,c) ***FEY(a,c)*** FEY(a,b)

The synthesized solution can be transformed to a simpler solution by substitution of a Fredkin gate.

Transformed circuit: NOT(d) NOT(b) TOF3(b,c,d) NOT(b) TOF3(a,c,d) NOT(c) TOF3(a,b,d) ***FRE(b,c,a)*** FEY(b,c) FEY(a,b)

Example 7: This is the inverse to the specification in Example 6. The realization after substitution of a Fredkin gate is simpler than found in the opposite direction. This show the importance of applying our synthesis method to both a specification and the corresponding inverse (except when the specification is self-inverse) and choosing the best of the two results.

Specification: [4,6,2,0,15,13,7,5,9,11,3,1,14,12,10,8]
Assign inputs (a,b,c,d) to (b,c,d,a) in the realization below:
Synthesized circuit: NOT(c) NOT(b) NOT(a) TOF3(a,b,c) NOT(a) FEY(a,d) ***FEY(a,b) TOF3(b,d,a) FEY(a,b)***
Transformed circuit: NOT(c) NOT(b) NOT(a) TOF3(a,b,c) NOT(a) FEY(a,d) ***FRE(d,b,a)***

The solution from [13] requires 2 FEY, 1 TOF3, 1 FRE and 1 NOT so is quite comparable in gate count but has a longest signal path of 4 whereas our result's longest path is 5.

5. NON-REVERSIBLE LOGIC

An arbitrary combinational circuit composed of non-reversible gates, *e.g.* AND, OR gates, can be mapped to a reversible circuit typically with the addition of some number of constant inputs and 'garbage' outputs [10].

We are investigating an approach that uses the two-place decomposition method presented in [17][16]. This method decomposes a specification into two-input gates. Its particular advantages in this context are (i) that it makes use of spectral input translation, which corresponds to Feynman gates, and (ii) that it produces gate structures amenable to simple mapping to reversible gates.

For example, for a three-input two-output full adder, the two-place decomposition method from [17] gives:

t=EXOR(a,b); sum=EXOR(t,c); carry=OR(AND(a,b),AND(c,t))

This can be mapped to the reversible logic circuit

TOF3(a,b,d) FEY(a,b) TOF3(b,c,d) FEY(c,b)

where d=0 on input, b is the sum on output, and d is the carry on output, which is the solution given in [13]. This circuit has one constant input and two 'garbage' outputs which are however basic inputs.

Not all circuits are so amenable to this mapping method. One approach is to find a reversible circuit in this manner, use it to generate a reversible specification, and then apply the synthesis method above in the hopes of improving the circuit. This does not always lead to improvement. For example, applying this to the full adder example leads to the realization

TOF3(b,c,d) TOF3(a,c,d) TOF3(a,b,d) FEY(c,a) FEY(b,a)

which is more costly than the result from mapping the two-place decomposition result. Other examples, show an improvement.

6. CONCLUSION

Results to date show the spectral-based synthesis method does indeed have promise. More investigation of the mapping of circuits produced by two-place decomposition and the overall utility of that approach is required.

The preliminary implementation of the reversible logic synthesis method is in C and represents functions and their spectra as vectors of length 2^n although fast transform methods are used in the implementation rather than the matrix transformation method. Execution time on a PC is negligible for the examples shown. A decision diagram implementation is under development so the method can be applied to larger problems.

While our synthesis method does not require extensive look-ahead or back-tracking techniques, it does require the full consideration of the possible Toffoli gates at each stage of the synthesis. As noted above, we are currently investigating spectral conditions to eliminate such searching,. Finally, we are investigating in more depth the use of two-place decomposition techniques in the synthesis of reversible implementations for non-reversible specifications.

REFERENCES

[1] Bennett, C., "Logical Reversibility of Computation," *IBM Jour. of Research and Development*, **17**, 1973, pp. 525-532.

[2] De Vos, A., "Towards Reversible Digital Computers," *Proc. European Conf. Circuit Theory & Design*, 1997, pp. 923-931.

[3] Feynman, R., "Quantum Mechanical Computers," *Optics News*,**11**, 1985, pp. 11-20.

[4] Fredkin, E., and T. Toffoli, "Conservative Logic," *International Jour. Theoretical Physics*, 1982, pp. 219-253.

[5] Hurst, S. L., D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, 1985.

[6] Karpovsky, M. G., *Finite Orthogonal Series in the Design of Digital Devices*, John Wiley and Sons, 1976.

[7] Kerntopf, P., "On Efficiency of Reversible Logic (3,3) Gates," *Proc. 7th Intl. Conf. MIXDES*, 2000, pp. 185-190.

[8] Landauer, R., "Irreversibility and Heat Generation in the Computational Process," *IBM Journal of Research and Development*, **5**, 1961, pp. 183-191.

[9] Margolus, N., *Physics and Computation*, Ph. D. Thesis, Massachusetts Institute of Technology, 1988.

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

[11] Peres, A., "Reversible Logic and Quantum Computers," *Physical Review A*, **32**, 1985, pp. 3266-3276.

[12] Perkowski, M. Private communication.

[13] Perkowski, M., *et al.*, "A General Decomposition for Reversible Logic," *Proc. Fifth Reed-Muller Workshop*, 2001, pp. 119-138.

[14] Thornton, M. A., R. Drechsler and D. M. Miller, *Spectral Techniques in VLSI CAD*, Kluwer, 2002.

[15] Toffoli, T., "Reversible Computing," in *Automata, Languages and Programming*, Springer-Verlag, pp. 632-644, 1980.

[16] Tomczuk, R., and D. M. Miller, "Combinational Logic Synthesis by Two-Place Decomposition and Auto-correlation Techniques," *Proceedings Canadian Conference on VLSI*, 1992, pp. 139-146.

[17] Tomczuk, R., Autocorrelation and Decomposition Methods in Combinational Logic Design, Ph.D. Dissertation, University of Victoria, 1996.

Acknowledgements: This work was supported in part by Natural Sciences and Engineering Research Council of Canada Research Grant 39409-96. Discussions and e-mail exchanges with Prof. Marek Perkowski, Portland State University, regarding reversible logic synthesis are gratefully acknowledged.