Reversible Logic Synthesis by Iterative Compositions

Portland Quantum Logic Group

Andrei B. Khlopotine *
andrei@pdx.edu

Marek Perkowski *
mperkows@ece.pdx.edu

Pawel Kerntopf **
pke@ii.pw.edu.pl

***** Department of Electrical and Computer Engineering, Portland State University
****** Institute of Computer Science, Warsaw University of Technology

ABSTRACT

A reversible circuit maps each output vector into a unique input vector, and vice versa. CMOS reversible / adiabatic circuits are currently the most important approaches to power optimization. This paper introduces an approach to synthesize generalized multi-rail reversible cascades for single-output Boolean functions. Minimizing the "garbage bits" is the main challenge of reversible logic synthesis. Experimental results over a set of single output functions (derived from Espresso PLAs) will be presented at IWLS 2002.

1. INTRODUCTION

Bennett and Landauer [2] proved that losing information in a circuit causes losing power. Information is lost when the input vector cannot be uniquely recovered from the output vector of a combinational circuit. The gate that does not lose information is called reversible. For instance, the so-called Feynman gate described by equations P = A, Q =A ⊕ B is reversible, as it can be easily seen in its truth table, because for each combination of output signals P, Q there is exactly one combination of input signals A, B.

The energy lost in a logic circuit has two components; one is related to non-ideality of switches and other technological factors and another to the information loss. While the first component is decreased with time by inventing new technologies and design principles such as adiabatic design, the second is related to information and can be decreased (to zero) only using reversible design methodologies. So far, the second component is much smaller in year 2002. According to [2], it is a necessary condition to use only reversible gates to build a circuit that will not loose energy during (internal) calculations. (However, energy may be lost for input and output operations.)

It was shown that reversible gates can be built in DNA, optical, quantum and other technologies but here we will concentrate on CMOS [1,4,5,18]. Developing systematic logic synthesis algorithms for reversible logic is still very immature, but some methods have been proposed [8,13,14]. In addition to the Feynman gate defined above, most papers discuss design using Toffoli and Fredkin [6] gates, as well as how to build these gates in several existing and future technologies. Analysis of three-input three-output gate families has been done [7]. This analysis created several new types of reversible gates. Most of reversible gates in literature are three-input three-output (3*3) or four-input four-output (4*4) gates, the exceptions are [4,5,8,13,15]. In this research we will stay with reversible gates with maximum size of 4*4 (including 1*1, 2*2, 3*3). To our knowledge no systematic method for synthesis using any gates with n>3 was ever published.

To avoid energy loss, the approach of Fredkin and Toffoli [6] has been used. It creates a "basic circuit" from reversible gates with garbage outputs (can be easily synthesized using the technology mapping). Next, this approach applies a "spy gate" for every primary output. The spy gate is a Feynman gate with B=0 which copies the output signal of the basic circuit. Next a mirror circuit is added with inputs from the second outputs of spy gates and from the garbage outputs of the basic circuit. The mirror circuit is the reverse of the basic circuit and has as many gates (that are inverses to gates of the basic circuit) as the basic circuit has gates. This solution leads to the duplication of the circuit's delay and cost of gates. The delay is 2n+1 where n is the delay of basic

circuit, and the gate cost is 2m + k where m is the number of gates and k is the number of primary outputs. Our approach reduces garbage and does not require mirror circuit at all.

The main differences of synthesizing a circuit with reversible gates, as compared to synthesizing a standard binary circuit, are the following:

1. The number of outputs of a logic gate is equal to the number of inputs. It is easy to find solutions sacrificing one or more gate outputs for garbage, but such solutions are of less value (they are still used, however, because better methods are not yet known [8,12,13]).
2. Every gate output that is not used as input to other gate or as a primary output is called garbage. A heavy price is paid for every garbage bit, if the garbage bits are left unattended, or if the mirror circuit and spy gates are added.
3. In reversible logic, fan-out of any gate output is not allowed; every output can be used only once. Feynman gates can be used as "copying circuits", the same way as in "spy circuits", to increase the fan-out. However, for every fan-out of two a Feynman gate is used. Obviously, this increases the cost and delay.
4. Several authors assume that there should be no loops of gates and we follow this assumption here (in general, this requirement is not mandatory for all technologies).

Concluding, the main rules for efficient reversible logic synthesis are the following: (1) use as many outputs of every gate as possible, and thus minimize garbage outputs. (2) do not create more constant inputs to gates than is absolutely necessary. (3) avoid leading output signals of gates to more than one input, because each such fan out of two requires adding one copying circuit.

The rest of the paper is organized as follows. Section 2 introduces the new family of reversible gates that was assumed as the library of gates in this work. Section 3 talks about the researched and simulated compositional synthesis methods for reversible logic. Section 4 outlines a pseudo code for the proposed synthesis method. Section 5 refers to the experimental results. Section 6 concludes the paper. The references are listed in section 7.

2. GENERALIZED FAMILIES OF REVERSIBLE GATES

Fig. 1 presents a generalized Feynman gate (the symbol of the gate at the bottom is EXOR), where f_1 denotes an arbitrary Boolean function of one variable. Similarly, the generalized Toffoli, Fredkin and Kerntopf gates are presented in Figs. 2, 3, and 4, respectively, where f_2 denotes an arbitrary boolean function of two variables. It can be easily verified from truth tables that all these gates are reversible. All the three above mentioned families can be extended to gates with an arbitrary number of control inputs. For example, the generalized Kerntopf gate with an arbitrary number n of inputs is defined as follows:

$P_1 = A_1, P_2 = A_2, \ldots, P_{n-2} = A_{n-2},$
$P_{n-1} = \text{MUX}(f_{n-2}, A_{n-1}, A_n),$
$P_n = \text{DAVIO}(f_{n-2}, A_{n-1}, A_n),$

where MUX (x,y,z) = x'y + xz, DAVIO (x,y,z) = x'z + y, f_{n-2} is an arbitrary function of n-2 variables being the control variable of the multiplexer.

Fig. 1. Generalized Feynman Gate

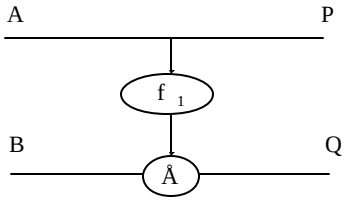


Fig. 2. Generalized Toffoli Gate

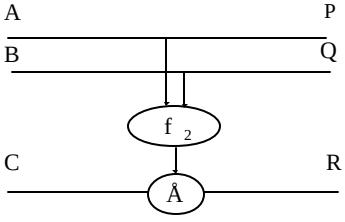


Fig. 3. Generalized Fredkin Gate

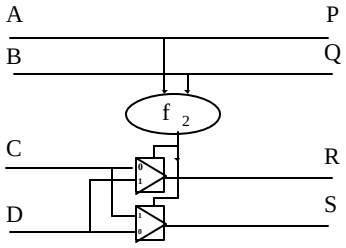
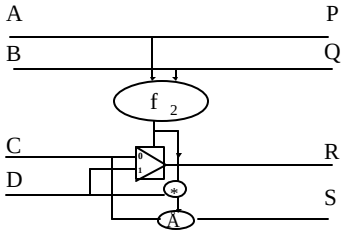


Fig. 4. Generalized Kerntopf Gate

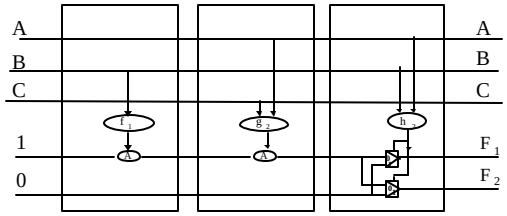


There are other available families of reversible gates and this is a valid topic for future research. In our work we stay withing the framework of the above presented family of reversible gates.

3. COMPOSITIONAL SYNTHESIS METHODS FOR REVERSIBLE LOGIC CASCADES AND ADAPTATION OF EXOR METHODS TO CASCADES

Compositional synthesis methods for reversible logic have been presented in [14]. Observe that the simplest structure for composition are cascades, because they have the same number of intermediate signals at every level. Cascade examples are presented in Figs. 5-7 and discussed in [8]. As we see there is only one signal added (constant 0 in example in Fig. 5) and this signal essentially becomes a function realized with (generalized) reversible gates.

Fig. 5. General Cascade of Feynman, Toffoli and Fredkin Family Gates

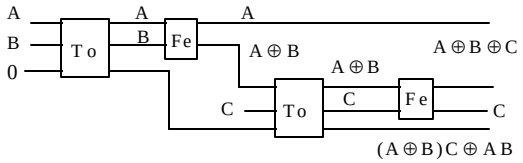


Example 1. Fig. 6 illustrates a more general case, realization of a Full Adder using Toffoli (To) and Feynman (Fe) gates, synthesized using the method introduced here. Let us discuss how it is created. It was first found that the original 3-input, 2-output function of the adder is not reversible and that it cannot be made reversible by adding one output signal (the reader can check it using Kmaps from the definition of reversibility). Thus one more constant input is added and it is assumed that the width of the circuit is 4 (see Figure 6). We have now two primary outputs, two potential garbage outputs, three primary inputs and one input constant. Not increasing the width is first assumed, and gates are selected to realize all primary output functions and to not generate garbage (which would require mirror and spy circuits). Whenever a solution cannot be found given these assumptions and the selected set of reversible gates, a backtrack is executed. Observe that even an algorithm with no heuristic cost function but based on depth search limit of four would find ultimately the solution from Fig 6. Because the number of gates and wire permutations is high, such approach would be exhaustive. Thus we need a heuristic cost function to be minimized. We use a maximization of combination of coincidence count and minimization of entropy of EXOR of the intermediate function in the wire of the new level and the primary output function, calculated for all pairs of wire and not yet realized output functions. Observe, that entropy-based cost function has small values for functions with many zeros (or many ones) but has high values for functions with approximately the same number of ones and zeros in their Kmaps, Decreasing entropy leads obviously to convergence of algorithm, but functions such as a variable or an EXOR of variables have simple realizations and the highest entropy. So, we treat such functions in a special way in the cost function. Similarly, all functions being products and sums of literals are treated specially in the cost functions [13]. After applying the first Toffoli gate from the left in Fig. 6, intermediate function AB is created which has high correlation with primary output $AB \oplus AC \oplus BC$. Functions A, B, AB, C are sufficient to realize all primary outputs, so next level is now composed. Function $A \oplus B$ is created as having high correlation (small value of cost function) with respect to primary output $A \oplus B \oplus C$. The variables after two input levels are now A, $A \oplus B$, C and AB. Toffoli gate is selected which realizes directly the majority function. The variables are now A, $A \oplus B$, C and $AB \oplus AC \oplus$

BC. Only one target output exists at this stage. It can be checked that Feynman gate is the best choice since it realizes A ⊕ B ⊕ C and primary input C (no garbage). Previous levels created only function A as potential garbage, the function has no garbage because all other outputs than primary outputs are primary inputs – so that energy taken from the power supply through A, B, C will be returned to outputs and primary inputs A and C (power supply). This circuit is optimal with respect to information and energy loss.

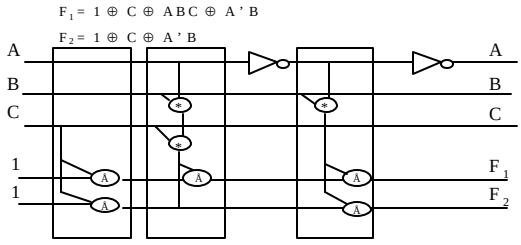
Example 2. As mentioned above, many well-known standard logic methods can be adapted to reversible cascades introduced here. For instance, Fig. 7 shows how an ESOP can be realized using such gates.

Fig. 6. Full Adder realized using Composition



Observe that when the generalized Toffoli gates are used, the upper part of the cascade delivers the primary inputs to reversible gates of the cascade, while the lower part allows to swap and negate wires or skip some of the gates.

Fig. 7. Example of multi-output ESOP cascade of Toffoli family gates



We have previously tried another approach based on building the given function by injecting the available reversible gates and then reducing the set of support variables. By doing this, we were hoping to get the given function with minimized number of intermediate signals. This methods is known as resubstitution. However, this approach brought about the situation when the number of intermediate signals was even increasing with every injection from some point and thus more garbage signals were created than if would be the case if cascading was implemented. Therefore, the cascading is the best performing method out of the two given. Cascading is the method used for the experimental results.

4. PSEUDOCODE OF THE ALGORITHM

The main contribution of this paper is an introduction of new reversible gate families and convergent synthesis algorithms for multi-rail reversible cascades for single-output functions. The method uses information-theory-based cost function, called coincidence count.. To compute the cost, we count the number of coinciding minterms in the Boolean space of the two functions to be compared.

We used a one-step look-ahead algorithm and performed branch and bound over all paths to the N best solutions. We assess the effectiveness of each of N best paths one step ahead before picking the one to follow. The main loop pseudocode:

```
iterative_compositions (func orig_func,  int n_best_gates)
{
 composed_func = false;
 best_gates = new func [n_best_gates];

 find_best_gates (orig_func, best_gates);
 while (composed_func != orig_func)
 {
   for (i = 0; i < n_best_gates; i ++)
   {
     t_func = create_func  (composed_func, best_gates [i]);
     best_gate = find_best_gates (t_func, n_best_gates);
   }
   inject_gate (best_gate, composed_funct, operator);
}
```

We first find N best gates for the current cascade. The heuristics is a coincidence count of the current gate injected into the composed function and the original function. Then, we look one step ahead and predict which one out of N will be the best one. We do this by getting N best gates (next cascade) for each of N best gates in the current cascade. We chose a gate from the current cascade that would give us the most effect in the future (in the next cascade). We inject the best gate from the current cascade into

the composed function (composed_func) and buffer already obtained N best gates for the next cascade and loop again with these N best gates as a current cascade. We keep looping until the composed_func is equal to the original function (orig_func).

5. EXPERIMENTAL RESULTS

No experimental results are achieved at this point. The software for the simulation is being developed and optimized for the performance**.**

Empirical results on the application of the proposed approach will be presented at the IWLS 2002.

6. CONCLUSIONS

The generalized Feynman, Toffoli, Fredkin and Kerntopf gates can be realized in CMOS technology. This allows for the use of the presented method in CMOS circuit synthesis too.

Current and future research involves: (1) characterizing new families of n-input n-output reversible gates for being used in regular structures and developing logic synthesis methods for them; (2) designing of reversible / adiabatic CMOS circuits for these families; (3) improving algorithm presented in this paper to achieve a more efficient synthesis; (4) improving software developed here to work on larger circuits. (5) developing software for multi-output functions.

7. REFERENCES

[1] W.C. Athas & L."J." Svensson , “Reversible Logic Issues in Adiabatic CMOS”, IEEE Workshop on Physics and Computation, 1994.

[2] C. Bennett, "Logical reversibility of computation", *I.B.M. J. Res. Dev.*, 17 (1973), pp. 525-532.

[3] BuDDy - A Binary Decision Diagram Package, http://www.itu.dk/research/buddy/index.html

[4] A. De Vos, B. Desoete, A. Adamski, P. Pietrzak, M. Sibinski, T. Widerski, ``Design of reversible logic circuits by means of control gates", Proc. 10th Int’l Workshop on Power and Timing Modeling, Optimization and Simulation, 2000, pp. 255-264.

[5] A. De Vos, B. Desoete, F. Janiak, A. Nogawski, ``Control gates as building blocks for reversible computers", Proc. 11th Int’l Workshop on Power and Timing Modeling, Optimization and Simulation, 2000, 2001.

[6] E. Fredkin, T. Toffoli, "Conservative Logic", *Int. Journal of Theor. Phys.*, 21 (1982), pp. 219-253.

[7] P. Kerntopf, “A Comparison of Logical Efficiency of Reversible and Conventional Gates,” 9th IEEE Workshop on Logic Synthesis, 2000, pp. 261-269.

[8] A. Mishchenko and M. Perkowski, “Logic Synthesis of Reversible Wave Cascades”, 11th IEEE/ACM Workshop on Logic and Synthesis, 2002.

[9] M. A. Perkowski, A. Sarabi, F. R. Beyl, "Universal XOR Canonical Forms of Switching Functions," *Proc.* Int’l Workshop on Applications of Reed-Muller Expansion in Circuit Design, 1993, pp. 2732.

[10] M. A. Perkowski, "A Fundamental Theorem for EXOR Circuits," *ibid*. pp. 52 – 60.

[11] M. Perkowski, A. Sarabi, and F. R. Beyl, "Fundamental Theorems and Families of Linearly Independent Forms for Binary and Multiple Valued", Proc. Int’l Workshop on Applications of Reed-Muller Expansion in Circuit Design, 1995, pp. 288-299.

[12] M. Perkowski, A. Sarabi, and F. R. Beyl, "Universal XOR Canonical Forms of Boolean Functions and its Subset Family of AND/OR/XOR Canonical Forms", IEEE Workshop on Logic Synthesis, 1995.

[13] M. Perkowski, “Generalization of reversible gates to n*n gates and their use in cascade synthesis. *Report PSU,* 2001.

[14] M. Perkowski, L. Jozwiak, P. Kerntopf, A. Mishchenko, A. Al-Rabadi, A. Coppola, A. Buller, X. Song, Md. M. Khan, S. Yanushkevich, V. Shmerko, and M. Chrzanowska-Jeske, “A General Decomposition for Reversible Logic”, Proc. Int’l Workshop on Applications of Reed-Muller Expansion in Circuit Design, 2001.

[15] J.Preskill, Lecture notes in quantum computing: *http://www. Theory.caltech.edu/~preskill/ph229*

[16] A. Sarabi, N. Song, M. Chrzanowska-Jeske, M. A. Perkowski, "A Comprehensive Approach to Logic Synthesis and Physical Design for Two-Dimensional Logic Arrays," *Proc. DAC'94,* San Diego, June 1994, pp. 321 - 326.

[17] N. Song, M. Perkowski, "Minimization of Exclusive Sum of Products Expressions for Multi-Output Multiple-Valued Input, Incompletely Specified Functions," *IEEE Trans. CAD*, Vol. 15, No. 4, April 1996, pp. 385-395.

[18] S.G. Younis, "Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic, Ph.D. Thesis, MIT, June 1994.