Efficient Algorithm for Positive-polarity Reed-muller Expansions of reversible circuits

Jing Hu
Department of Computer Science &Technology, Harbin Engineering University, Harbin Heilongjiang,150001,China.
(E-mail: hjlyh@vip.sina.com)

GuangSheng Ma
Department of Computer Science &Technology, Harbin Engineering University, Harbin Heilongjiang,150001,China.
(E-mail:maguangsheng@hrbeu.edu.cn)

Gang Feng
Department of Computer Science &Technology, Harbin Engineering University, Harbin Heilongjiang,150001,China.
(E-mail: fenggang@hrbeu.edu.cn)

***Abstract*—In this paper, we build mathematical modeling for reversible circuits and derive required parameters of cost function from this modeling. The heuristic algorithm plots out reversible circuit into some partitions, uses a priority queue based on search tree and explores candidate components at each partition in order of utilization ratio. We demonstrate that using this heuristic, path delay can be reduced by 8% compared to existing synthesis method. The improvements increase for strict delay constraints making synthesis especially important for high performance and a large number of inputs and outputs designs.**

***Index Terms*—reversible circuits, synthesis, reed-muller expansion, Boolean polynomial**

I. INTRODUCTION

ENERGY loss is an important consideration in digital design. Landauer's principle states that logic computations that are not reversible, necessarily generate heat for information that is lost [1]. As a result, reversible circuits play an important role on low power design. It becomes new perspectives to build almost energy loss-less, ultra-small, and ultra-fast quantum computers. Moreover, reversible computing is applied to 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.

The reversible synthesis differs significantly from the conventional synthesis because that no fan-outs and no feed-backs restrictions are applied to reversible gates. Synthesis approaches are not well developed for reversible circuits even for small numbers of inputs and outputs. Several heuristics methods presented in [2]-[5] optimized reversible circuits by template matching. Unfortunately, these heuristics do not scale well and require extensive use of template matching. They are not avoidable for extensive searching and feasible for synthesizing reversible circuits with a large number of inputs. P. Kerntopf, in [6], proposed a heuristic algorithm to reversible logic synthesis using a new complexity measure based on shared binary decision diagrams with complemented edges. But, it is only fit for small numbers of inputs and outputs. It may generate exponential explosion with increasing number of them. D. Maslov , in [7], showed that $|Q| = n \cdot 3^{n-1}$, if Q is the set of all possible gates with n inputs. Given the model for function implementation, the problem of synthesis problems, we use an XOR sum of products expression of the output function to synthesize the circuit. Use of such a Reed-Muller expansion of the function was also suggested in [8]. However, the above-mentioned approaches have only one constraint, area. It has neglected the impact of path delay. Moreover, it may be highly complicated as it is applied to functions with a large number of inputs. Finally, it claims that it does not require output permutation or extra garbage lines (such lines are required to equalize the number of inputs and outputs). Unfortunately, in some cases garbage is unavoidable.

Our algorithm has several key characteristics in this contribution. Firstly, reversible circuit is mapped onto matrix model so as to make it easier to extract the required factors of cost function and inputs and outputs of each point in circuit. Secondly, it minimizes the number of gates and path delay as the primary objective and the size of the gates as the secondary objective. Thirdly, we act the reversible logic synthesis problem as multi-output logic synthesis. In the algorithm, reversible circuit is split into some partitions so that it makes sure that target function in each partition is single output function and simplifies the search space and algorithm complexity, and therefore the method has greater potential to be extended to functions with more than just a few inputs and outputs. Fourthly, we formulate the reversible logic synthesis problem via symbolic algebra analysis [9].

The organization of the paper is as follows. Mathematical model is built in section II. The considered ideas for reversible logic synthesis are discussed and the correlative conceptions are presented in section III. The detailed descriptions of our algorithm to synthesize reversible circuits are given in section IV. Experiment results when run on a set of benchmark circuits are shown in Section V. The paper concludes with section VI.

This work is supported by the NSFC No. 60273081 and the HEUF047088.

Jing Hu, Guangsheng Ma and Gang Feng are with the Department of Computer Science&Technology at the Harbin Engineering University, Harbin Heilongjiang, 150001, China. (e-mail: hjlyh@vip.sina.com)

1-4244-0765-6/06/$20.00 ©2006 IEEE

II. MATHEMATICAL MODEL

Any Boolean function can be described as an XOR sum of products. The positive-polarity Reed-Muller (PPRM) expansion uses only uncomplemented variables and can be derived easily from the function's sum of products expansion. The PPRM of a function is unique and of the form, as in

$$F(x_1,x_2,\ldots,x_n)=a_0\oplus a_1x_1\oplus a_2x_2\oplus\ldots\oplus a_nx_n\oplus a_{12}x_1x_2\oplus a_{13}x_1x_3\oplus \ldots\oplus a_{n-1,n}x_{n-1}x_n\oplus\ldots\oplus a_{1,2,\ldots,n}x_1x_2\ldots x_n \quad (1)$$

Where $a_i\in\{0,1\}$, x_i are all uncomplemented (positive polarity) and $\oplus$ denotes XOR Boolean operation [8].

We can find that some redundant components may exist, if an input is used two or more components. A simple case in point is function $ab\oplus ad$ for signal a, b and d, which is described as $a(b\oplus d)$, $a(b\oplus ad)$ or $a(ab\oplus d)$. Function $ab\oplus ad$ can be expressed with a form, $a(b\oplus d)$, if gate $b\oplus d$ has existed at other place or it makes cost decreasing.

On the other hand, such as function a$\oplus$ b$\oplus$ c, it can be expressed as $a\oplus(b\oplus c)$, $(a\oplus b)\oplus c$ or $b\oplus(a\oplus c)$. Under this condition, how should we select components to build a reversible circuit? In this work we present a multi-output function, $F(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta)$, to describe a reversible circuit, where $x_1,x_2,\ldots,x_n$ are original inputs, $f,g,\ldots,h$ are required outputs of the reversible circuit, and δ is the number of spanned components from original input to designated component. To begin with, adjust reversible circuits so as to insure that relative components are adjacent (see Fig. 1).

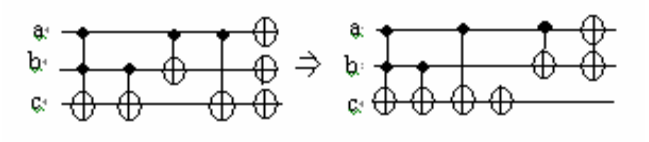

Fig. 1 (a) Architecture before adjusting (b) Architecture after adjusting

Because of the special structure of reversible circuit, cascade, it fits in well with mapping the multi-output function onto a matrix M,

$$\begin{array}{c} \\ i \\ \\ j \\ \\ f \\ \end{array}\begin{array}{c} \begin{array}{c}\delta_{i,\ldots j}\end{array} \\ \begin{bmatrix} \ldots & 0 & \ldots \\ \ldots & 0 & \ldots \\ \ldots & x_i,\ldots,x_j(i,\ \delta_{i,\ldots j}) & \ldots \\ \ldots & \ldots & \ldots \\ \ldots & x_i,\ldots,x_j(j,\ \delta_{i,\ldots j}) & \ldots \\ \ldots & 0 & \ldots \\ \ldots & f_{i,\ldots j} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix} \end{array}.$$

The spatial correlations of circuit are modeled by partitioning the reversible circuit into n column grids. Let any column be a component and its elements be 0,• or ⊕ symbols denoting NOTHING , AND and XOR Boolean operation respectively. The model cannot only clearly represent the relation among elements of each column. More important, it can also efficiently represent required factors of cost function and input and output values of each component. For example, we model the inputs and outputs of $\delta_{i,\ldots j}$ th column using $F(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta_{i,\ldots j}-1)$ and $F(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta_{i,\ldots j})$, respectively. It is assumed that the parameter δ_{max} denotes the maximum of No.column, namely the above-mentioned n. Let us consider the parameter δ_{max} and L as two factors of the cost function, where parameter L that is the number of maximum row denotes the number of inputs and outputs. In ordering to count cost function, we need do following definition.

Definition 1(don't care columns) : Two neighboring columns are called don't care columns, if all nonzero elements of them are at different rows or nonzero elements are same rows and the previous one is • .

By our definition, parameter δ_{max} expresses the number of components that can be utilized to estimate the area of one. For simplicity, we assume the delays of the generalized NOT, CNOT and Toffoli gates are 1, 2 and 3, respectively. The overall delay distribution of the reversible circuit can be computed very straightforwardly by merging don't care columns into a single column. As a simple example, a reversible circuit is fig. 2(a), and fig. 2(b) is its matrix expression, where δmax=3 and δmax'=3 and L=3.

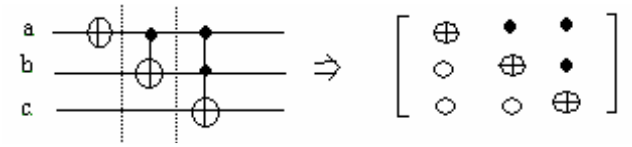

Fig. 2 (a) Cascade architecture (b) Its matrix model

III. DETAILED ANALYSIS

To simplify synthesis of reversible circuits, we give the following definition.

Definition 2(partition) : If all the involved inputs are applied to same output in successive columns, this section is called a partition.

That is, a partition includes successive gates whose target line is at same output or whose last target line is at one. As a result, it makes sure that target output function in each partition is single output function. And according to the number of maximal column in partition, we define the serial number of the partition.

To illustrate the solution of the procedure, consider the following example. A partition on f is from $f(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta_{i,\ldots j})$ to $f(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta_{k,\ldots,l})$. Its input function $F(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta_{i,\ldots j}-1)$ can be represented as the outputs of original input, $x_1,x_2,\ldots,x_n,f,g,\ldots,h$, at $(\delta_{i,\ldots j}-1)$th column, and its output function is $F(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta_{k,\ldots,l})$. Its target output function is $f(x_1,x_2,\ldots,x_n,f,g,\ldots,h,\delta_{k,\ldots,l})$ and serial number is m. Without loss of generality, we can find the relative information of any partition by the described model. It is remarkable that partitions may lap over because of having

different partition with respect to different output.

The multi-output function of the partition in last example can be transformed into PPRM form, $f(x_1', x_2', \ldots, x_n', f', g', \ldots, h')$. Its target output is f, and its inputs are $x_1', x_2', \ldots, x_n', f', g', \ldots, h'$, which are the output values of $(\delta_{i,\ldots j}\text{-}1)$ th column. Although we reduce the search space by partition, the one for a good single Toffoli gate substitution with a quantum circuit is so big that a heuristic is needed. On second thoughts, XOR and AND Boolean operations of the PPRM form are mapped into the additive and multiplicative ones of Boolean polynomials that are similar to ones of real-valued polynomials, only a constraint is required, i.e. Boolean variable $x \bullet x = x$. Consider applying the decomposition algorithm [9] based on Gröbner basis to getting candidate gates for Boolean polynomials. Its basic idea is to decompose Boolean polynomials into available factors. The detail of this relaxation procedure are being omitted for lack of space, but can be found in [9]. Using the algorithm, we can get its equivalent transform forms. Then check whether the gates exist in matrix M. If exist, make corresponding adjustment without changing the output of the target function; Otherwise, add them to the circuit and do repeatedly the above process. If adjustment is successful and the solution found is better than any previous solution, we can perform the adjustment; Otherwise, the adjustment is not allowed.

The abovementioned question, however, is how to solve some special forms that may have two or more transforms. As a simple example, consider synthesizing a function, a⊕ b⊕ c. It can be transformed into some forms, $a\oplus(b\oplus c)$, $(a\oplus b)\oplus c$ or $b\oplus(a\oplus c)$. To solve this sort of problem, a guideline is necessary to fix the priority of candidate gates. Such guideline can make sure that the result is oriental.

Algorithm Subroutine performing budget of candidate components

```
Function budget(M, Par[n])
# Given a reversible circuit M, its partition Par[n] and a set L of reversible gates(only include NOT, CNOT and Toffoli gates) as universal library.
for m=1 to n
#m-D is data-path delay of the m-th partition.
Candidates components(Par[m]) ← GuidedDecomposition(Par[m], m-D, L)
# Note that function GuidedDecomposition(Par[m], m-D,L) return the solution-tree rather than the best solution[11].
candidates ← candidates components(Par [m])
end for
priority candidate gates ← priority(candidates)
return (candidate components(Par [n]), priority candidate gates)
```

After initializing some partitions for reversible circuit, explore candidate gates of each partition. According to utilized times, candidate gates is endow with priority. You need set it on the highest priority if the candidate gate has existed in the reversible circuit. The algorithm gives preference to the gates with more utilization ratio as all things being they are more likely to be close to the solution. In terms of sort ascending of the serial number of partition, we recursively apply the same steps to each partition. For each partition, pop gate with highest priority from candidate gates queue of the partition. It is noticeable that, taking the relativity among components into consideration, we should check whether some components might be merged into (deleted from) other partitions after synthesizing a partition. And adjust responding number of the partition.

IV. THE ALGORITHM

Applying the above ideas to a reversible function yields an optimized reversible function. A basic flowchart of algorithm is shown in Fig. 3. To simplify quantum realizations of reversible circuits, we consider expressing reversible circuits with the generalized NOT, CNOT and Toffoli gates so that we have an optimal starting point. Thus, it is easier to come up with a good procedure for substituting these gates in the circuit to yield the most simplification. Having done that, to substitute special gates for SWAP and Fredkin gates.

Consider, any cascade structure of a reversible function as input of the algorithm and an optimized reversible logic as output of the one. To begin with, perform the preprocessing: Firstly, adjust circuit so that it makes the gates with same output adjacent to the best of one's abilities. And, get rid of useless gates that are at non-required output lines or not used the required output as follow Fig. 4. Secondly, mapping the reversible circuit onto a matrix M. Thirdly, plot out some partitions and budget candidate components by the two subroutines. Moreover, the algorithm enters a loop where it pops a candidate gate from the priority queue that will bring us closer to the desired solution. It is necessary to consider the cost of the entire circuit as the overall cost can be affected by a

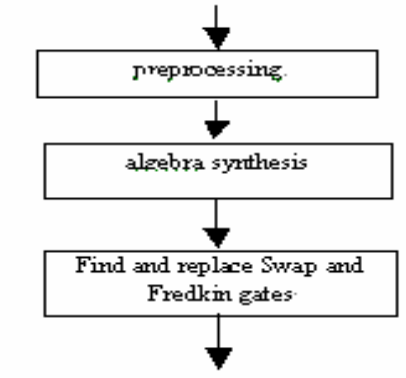

Fig. 3 Flowchart of algorithm

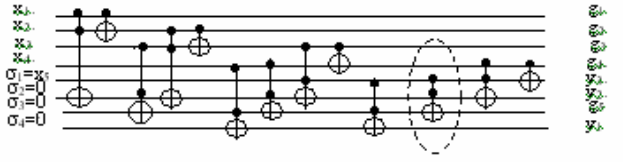

Fig. 4 An example of eliminating useless gates

local optimization. The parameter δ_{max} and δ_{max}' is used to measure adjustment's effectiveness. Compare original performance with adjustment's and save which one is better. Taking these factors into consideration, we define cost function CF, as in (2).

$$CF=\alpha\delta_{max}+\beta D+\gamma\phi \quad , \qquad (2)$$

where the ϕ is quantum cost. The weights α, β and γ add up to 1 and their values are decide by the emphasis of different design. If cost CF' of adjusted circuit is less than or equal to the original CF, we can perform the adjustment and choose the best position of each component to achieve performance optimization. Otherwise, we restore the previous valid configuration and decide whether other candidate gates present an attractive path to follow for synthesis.

V. Experimental Results

To test the effectiveness of the proposed approach, we implemented the described algorithm in C programming language. We took several benchmarking circuits and compared their quantum realization costs before and after applying the above approach.

TABLE I depicts the results of applying various algorithms to simplifying some reversible circuits composed of NOT, CNOT and Toffoli gates. The "Prior" and "Our" columns indicate the best published quantum cost in previous literatures (Maslov, Dueck, Miller)[10] and our synthesized quantum cost respectively.

TABLE I
BENCHMARKING RESULTS FOR SYNTHESIS

examples	our			prior		
	#gates	delay	#lines	#gates	delay	#lines
4mod5	5	5	5	9	8	5
9sym	28	15	12	28	21	12
rd53#1	30	16	7	30	19	7
rd53#2	12	12	7	12	12	7
rd53#3	12	9	8	12	10	8
rd53#4	20	14	7	20	16	7
rd53#5	16	15	7	16	16	7
rd73	20	12	10	20	15	10
rd84	28	14	15	28	22	15
ham7	23	20	7	23	23	7
ham15	130	120	15	132	132	15
hwb8	634	625	8	637	633	8

The first two reversible circuits in TABLE I are single output functions, 4mod5 and 9sym. The next five ones are function rd53 that is the 5-input 3-output symmetric function. The functions rd73 and rd84 are 7-inputs 3-outputs and 8-inputs 4-outputs. Functions ham7 and ham15 are the size 7 and 15 Hamming optimal coding function. Function hwb8 is the size 8 hidden weighted bit function. Its output equals its input shifted on the number of positions equal to the number of ones in the input pattern. Hidden weighted bit function is known to have an exponential size BDD for any variable ordering.

In the "our" set of results of TABLE I, we assume that the weights $\alpha = 0.5$, $\beta = 0.3$ and $\gamma = 0.2$. The #gates column shows the numbers of gates in the reversible circuit. The delay reported is the cumulative delay of gates on the critical path. The #lines reported is the number of the number of inputs and outputs. Experimental results on a set of benchmarks show that our algorithm is indeed effective in solving synthesis problem for reversible circuits. Using our heuristic, we can achieve the best published area in previous literatures and an average delay improvement of 8%. The reduction in circuit delay comes with no area penalty by choosing appropriate weights.

VI. Conclusion

The paper introduces the approach of multiple purpose synthesis of reversible logic that bases on matrix model. It is also applicable to multi-output binary functions. We have presented an algorithm, which sets up some partitions and uses a Boolean polynomial decomposition of each partition to get candidate gates. The algorithm searches the candidate factors in priority order to try to find the best possible solution that increases the likelihood of meeting tight delay objectives while meeting area constraints. The presented heuristic algorithm has been tested on many examples and proved very promising. The heuristic effectively complements synthesis of reversible. It is more suitable for reversible functions with a large number of inputs than existing methods.

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process", *IBM J. Res.*, 5, pp.183- 191, 1961. as the first . . ."

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

[3] A. Khlopotine, M. Perkowski, and P. Kerntopf, "Reversible logic synthesis by iterative composi- tions", *in Proc. Int. Wkshp. Logic Synthesis*, pp. 261- 266, June 2002.

[4] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits", *in Proc. Design Automation Conf.*, pp. 419-424, June 2002.

[5] D. Maslov, C. Young, D. M. Miller, G. W. Dueck, "Quantum Circuit Simplification Using Templates", *in Proc. of the Design, Automation and Test in Europe Conference and Exhibition* (DATE'05), pp.1208-1213.

[6] P. Kerntopf, "A New Heuristic Algorithm for Reversible Logic Synthesis", *in Proc. Design Automation Conf.*, June 7–11, 2004, San Diego, California, USA.

[7] G. W. Dueck and D. Maslov, "Reversible function synthesis with minimum garbage outputs", *in 6th International Symposium on Representations and Methodology of Future Computing Technologies*, pp. 154-161, March 2003.

[8] A. Agrawal and N. K. Jha, "Synthesis of Reversible Logic", *in Proc. of the Design, Automation and Test in Europe Conference and Exhibition* (DATE'04).

[9] A. Peymandoust and G.De Micheli, "Application of symbolic computer Algebra in High-level Data- flow Synthesis", *IEEE Trans. On computer-aided design of integrated circuits and systems*, vol.22, pp.1154-1165, 2003.

[10] http://www.cs.uvic.ca/~dmaslov/