On the Behavior of Substitution-based Reversible Circuit Synthesis Algorithms: Investigation and Improvement

Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi
Email: {msaeedi, szamani, msedighi}@ aut.ac.ir

Quantum Design Automation Group
Computer Engineering Department - Amirkabir University of Technology
Tehran, Iran

Presented by:
Mahtab Niknahad (Amirkabir University of Technology)

ANNUAL Symposium ON VLSI 2007
PORTO ALEGRE, BRAZIL

Outline

- Introduction
- Basic concept
- Previous work
- Search-based method
- Proposed method
  - DFS and BFS synthesis methods
  - Hybrid synthesis method
- Experimental results
- Conclusion
- References

QDA Group, Amirkabir University of Technology, Tehran, Iran

Introduction

- Reversible function
- Landauer's paper about energy dissipation
- Bennett's paper about the power dissipation of reversible gates
- The applications of reversible circuits
  - Low power CMOS design
  - Optical computing
  - Quantum computing
  - Synthesis

Basic Concept (1)

- Reversible Function
- A new notation for reversible function
- Reversible gate
- various reversible gates
  - NOT, CNOT, C2NOT, …
- $x_i(out)=x_i\ (i<n),\ x_n(out)=x_1x_2\ldots x_{n-1}\oplus x_n$.
- PPRM (Positive polarity Reed-Muller) expansion.

$$f(x_1, x_1, \ldots, x_n) = a_0 \oplus a_1x_1 \oplus \cdots \oplus a_nx_n \oplus a_{12}x_1x_2 \oplus \cdots \oplus a_{n,n-1}x_{n-1}x_n \oplus \ldots \oplus a_{12\ldots n}x_1x_2\cdots x_n$$

Basic Concept (2)

- Gate Complexity
  - The number of terms in PPRM expansion
  - The number of non-zero coefficients in previous equation
- gate cost
  - The number of elementary operations required to realize a gate
- Reducing the number of cascaded gates and the number of gate control lines are always preferred

Previous Work

- Two type of algorithms:
  - Transformation-based algorithms
  - Synthesis Algorithms (constructive algorithm)
- The size of a reversible circuit can be very large
  - A practical algorithm may become extremely difficult
- Search-based algorithms
  - Extensive exploration is required

Search-based method (1)

- PPRM expansion
- common sub-expressions
  - common sub-expressions between the PPRM expansions of multiple outputs are identified
- Primary objective
  - Minimize the number of gates (i.e. factors) needed to transform a PPRM expansion into the identity function.
- Secondary objective
  - Minimize the size of the individual gates (i.e. the number of literals in each factor) which is related to the number of control lines for each CNOT-based gate.

Search-based method (2)

- **While common sub-expressions seem to be good candidates for current substitution, there is no guarantee that the resulted PPRM expression contains fewer terms.**
  - **Example 1 (common sub-expression)**
  - $(a_{out}=a\oplus b,\ b_{out}=b\oplus bc\oplus ac,\ c_{out}=1\oplus c)$.
  - $b=b\oplus ac$ (common factor)
  - $(a_{out}=a\oplus b\oplus ac,\ b_{out}=b\oplus bc\oplus ac,\ c_{out}=1\oplus c)$
  - Not only this common factor dose not decrease the number of terms in its original expression (i.e. *b*), but also it increases the number of terms in the other expression (i.e. *a*).

Search-based method (3)

- Therefore, **greedy** common sub-expression selection may not result in a better expression
- greedy disregard of non-common factors
- Example 2 (non-common sub-expression):
- $a_{out}=1\oplus a\oplus bc\oplus ac$, $b_{out}=b\oplus bc\oplus ac$, $c_{out}=c$
- $b_{out}=b\oplus a$ (non-common factor)
- Using this factor will result in an optimized CNOT-based circuit with the cost of 8.
- The best report result is a circuit with the cost of 16.

Search-based method (4)

- An important result:
  - Based on the previous two examples, it can be concluded that the selection metric of common sub-expressions may result in poor results. Therefore, there is a clear advantage to use other factors as well as common factors to simplify a reversible circuit.

Search-based method (5)

- The only sub-expression that can increase the number of terms in PPRM expansions is $v_{i,out} = v_i \oplus 1$
- Any increase in the number of PPRM terms is discarded
- However, there is no direct correlation between local increase or decrease in the number of PPRM terms and the final synthesized result (The next slide)

Search-based method (6)

- $a_{out}=1\oplus a\oplus c\oplus ab\oplus ac$, $b_{out}=1\oplus b\oplus ab\oplus ac$, $c_{out}=a\oplus 1$
- $a_{out}=a\oplus c$ (Common factor)
- $a_{out}=1\oplus a\oplus c\oplus ab\oplus bc\oplus ac$, $b_{out}=1\oplus b\oplus c\oplus ab\oplus bc\oplus ac$, $c_{out}=1\oplus a\oplus c$
- The number of term eliminations is (-1,-2,-1)
- Using this factor as an initial substitution will result in better-synthesized result

- **The greedy substitution of factors, which results in fewer PPRM terms, may not lead to solution at all**

Proposed Method (1)

- DFS and BFS synthesis algorithm
- DFS algorithm
  - It considers the results of previously substituted factors before examining any new substitutions
- BFS Algorithm
  - It considers all new factors before using the results of previously substituted factors
- Deeper nodes are more likely to be close to a solution
- The authors of [12] used a DFS-based search method.
- Using a new previously unconsidered factor to synthesize a circuit

Proposed Method (2)

- **Example 4 (DFS vs. BFS synthesis methods)**
  - $a_{out}=1 \oplus ab \oplus bc \oplus ac$, $b_{out}=1 \oplus a \oplus c \oplus bc \oplus ac$, $c_{out}=1 \oplus a \oplus b \oplus ab \oplus bc \oplus ac$
- *BranchNo* is the maximum number of acceptable substitutions at each node
- *MaxDepth* is the maximum tree depth at which PPRM terms can still be increased.
- Please see the paper for more details.
- Testing more factors will always lead to better results
- Efficient method to synthesize

Hybrid synthesis method

- In the first *MaxDepth* levels, a BFS algorithm is used to evaluate all of the possible factors including common sub-expressions
- After that, a DFS algorithm is used to evaluate the previously considered nodes, as we believe that deeper nodes have more opportunities to lead a result.
- By using a hybrid DFS & BFS method, the benefits of both algorithm are used.
- Several notations and theorems are proposed in this paper which were omitted due to the lack of time.

Experimental results (1)

- The first eight examples are come from literature.
- The second eight examples are introduced for the first time.
- Please note that the second examples are not synthesized using the previously published paper (Search-based method)
- Pentium IV 3.0GHz computer with 1GB memory

Experimental results (2)

- *Cost* as the total cost of final circuit and *SNd* as the number of searched nodes

Table 1. The results of using our method on examples of [12]

Circuits [12]	DFS [12]		Our Method		Imp.(%)
	Cost	SNd	Cost	SNd	Cost
1	16	11	8	15	50
2	7	761	7	300	0
3	15	7	7	10	53
4	17	156	9	786	47
5	51	9515	27	8256	47
6	7	4	7	4	0
7	20	5	20	5	0
8	12	230	12	139	0
Average					24

Table 2. The results of using our method for eight new examples where the method of [12] can not produce any circuit

Our Circuits	No. of Gates	Cost	SNd
1	7	19	66
2	6	14	77
3	7	11	4387
4	7	19	352
5	15	51	678
6	14	62	9712
7	17	73	74521
8	16	72	85191

Experimental results (3)

Table 3. Our synthesized circuits for the attempted examples

Circuit#		Description	Our synthesized circuits
[12]	1	(4,5,6,1,0,7,2,3)	b=b⊕a, a=a⊕bc, b=b⊕a, a=a⊕1
	2	(7,6,4,5,0,1,2,3)	a=a⊕1, b=b⊕a, c=c⊕ab
	3	(0,1,2,5,4,3,6,7)	b=b⊕a, a=a⊕bc, b=b⊕a
	4	(0,6,2,3,4,5,1,7)	a=a⊕c, b=b⊕c, c=c⊕ab, b=b⊕c, a=a⊕c
	5	(0,14,2,3,4,5,6,7,8,9,10,11,12,13,1,15)	a=a⊕1, c=c⊕1, b=b⊕ad, d=d⊕abc, b=b⊕ad, a=a⊕1, c=c⊕1
	6	(4,5,6,7,2,3,1,0)	c=c⊕ab, b=b⊕a, a=a⊕1
	7	(8,9,10,11,12,13,14,15,4,5,6,7,2,3,1,0)	d=d⊕abc, c=c⊕ab, b=b⊕a, a=a⊕1
	8	(0,1,2,3,6,7,5,4,14,15,13,12,9,8,11,10)	d=d⊕ab, b=b⊕a, d=d⊕bc, c=c⊕b
Our Circuits	1	(6,7,2,0,3,4,5,1)	a=a⊕bc, b=b⊕ac, c=c⊕a, a=a⊕b, b=b⊕ac, a=a⊕1, b=b⊕a
	2	(4,3,7,0,2,6,5,1)	c=c⊕ab, b=b⊕ac, a=a⊕c, c=c⊕b, b=b⊕a, a=a⊕1
	3	(1,7,0,3,6,5,2,4)	a=a⊕b, c=c⊕a, b=b⊕c, c=c⊕ab, a=a⊕c, c=c⊕b, c=c⊕1
	4	(7,3,2,0,5,4,1,6)	c=c⊕1, a=a⊕1, b=b⊕a, a=a⊕bc, b=b⊕ac, c=c⊕ab, a=a⊕1
	5	(6,0,12,15,7,1,5,2,4,10,13,3,11,8,14,9)	c=c⊕1, c=c⊕ad, c=c⊕ad, c=c⊕a, b=b⊕cd, c=c⊕abd, d=d⊕ac, a=a⊕b, a=a⊕bc, a=a⊕c, c=c⊕1, d=d⊕b, b=b⊕1, b=b⊕ad, a=a⊕1
	6	(9,0,2,15,11,6,7,8,14,3,4,13,5,1,12,10)	b=b⊕a, a=a⊕bc, d=d⊕a, c=c⊕abd, c=c⊕ad, b=b⊕acd, a=a⊕1, d=d⊕c, c=c⊕b, b=b⊕c, b=b⊕acd, a=a⊕1, b=b⊕ad, d=d⊕1
	7	(6,15,9,5,13,12,3,7,2,10,1,11,0,14,4,8)	a=a⊕1, b=b⊕ac, a=a⊕1, c=c⊕abd, d=d⊕a, a=a⊕bcd, c=c⊕abd, b=b⊕a, d=d⊕bc, b=b⊕1, c=c⊕1, a=a⊕cd, c=c⊕b, d=d⊕bc, b=b⊕cd, d=d⊕c, c=c⊕a
	8	(11,3,9,2,7,13,15,14,8,1,4,10,0,12,6,5)	c=c⊕bd, c=c⊕b, c=c⊕abd , d=d⊕ac, b=b⊕acd, a=a⊕cd, d=d⊕a, d=d⊕abc, c=c⊕1, d=d⊕bc, c=c⊕1, a=a⊕1, d=d⊕ac, c=c⊕a, a=a⊕1, d=d⊕a

Conclusion

- A more efficient search-based synthesis method is proposed
- There is no direct correlation between local common sub-expression factors and the final circuit
- A new hybrid DFS/BFS reversible circuit synthesis algorithm is proposed.
- Experimental results demonstrated that using our method can lead to better synthesis circuits with respect to the total circuit cost as well as the probability of leading to a synthesized circuit.

References

- [1] R. Landauer, “Irreversibility and Heat Generation in the Computing Process,” IBM Journal, vol. 5, pp. 183-191, July 1961.
- [2] C. Bennett, “Logical Reversibility of Computation,” IBM Journal, vol. 17(6), pp. 525-532, November 1973.
- [5] M. Nielsen and I. Chuang, “Quantum Computation and Quantum Information” Cambridge University Press, 2000.
- [12] P. Gupta, A. Agrawal, and N. K Jha, “An Algorithm for Synthesis of Reversible Logic Circuits,” TCAD, November 2006.

Thanks for your attentions.

Any question?

(msaeedi@aut.ac.ir)