Partially Redundant Logic Detection Using Symbolic Equivalence Checking in Reversible and Irreversible Logic Circuits

David Y. Feinstein and Mitchell A. Thornton
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX, USA
dfeinste,mitch@engr.smu.edu

D. Michael Miller
Department of Computer Science
University of Victoria
Victoria, BC, Canada
mmiller@cs.uvic.ca

Abstract

This paper investigates partially redundant logic detection and gate modification coverage in both reversible and irreversible (classical) logic circuits. Our methodology is to repeatedly compare a benchmark circuit with a modified copy of itself using an equivalence checker. We have found many instances in the irreversible logic ISCAS85 benchmarks where single gate replacements were not detected, indicating no change in functionality after gate replacement. In contrast, we demonstrate that the Maslov reversible and quantum logic benchmarks exhibit very high gate modification fault coverage, in line with the expectation that reversible circuits, which implement bijective functions, have maximal information content.

1. Introduction

In this paper, we contrast the ability of reversible logic and irreversible classical logic verification tools to detect partial redundancies by circuit modifications. Generally, one would expect that a given circuit would not be equivalent with a version of itself having undergone some structural modification. And yet, it is well known that for irreversible (classical) logic circuits, partially redundant logic or internal don't-care cones allow for such structural modifications to be undetected since such replacements do not affect overall functionality due to internal don't-care conditions. With the recent development of reversible logic simulation tools such as the Quantum Multiple-Valued Decision Diagrams (*QMDD*) package [7,8], we were motivated to investigate whether similar partially redundant logic instances are likely to be found with emerging quantum and reversible logic technologies.

Detection of partially redundant logic within any design, reversible or irreversible, has ramifications for logic synthesis, for design verification, and for design for test (*DFT*) issues. In logic synthesis, partially redundant logic may use valuable circuit resources and its removal can provide more efficient circuits.

While several researchers have recently dealt with the testability issues of reversible logic [3], to the best of our knowledge, this paper is the first investigation of reversible logic verification for structural modification detection using equivalence checking. To facilitate our research, we have developed similar symbolic equivalence checking programs for irreversible classical logic (CMBtest) and reversible logic (DQMMceq) that automatically compare a benchmark file with a controlled modified version.

The paper is organized as follows. In Section 2 we discuss previous work that deals with partially redundant logic detection in irreversible logic. We also introduce some of the properties of reversible logic. Our circuit modification strategy for both reversible and irreversible logic is outlined in Section 3. In Section 4 we discuss our preliminary experimental results. Conclusions and suggestions for further research appear in Section 5.

2. Background and Preliminary Information

For classical irreversible logic, a gate or segment of logic is redundant if it can be removed from a circuit without affecting the functionality of the circuit. A gate or segment of logic is partially redundant if it can be modified without affecting the functionality of the circuit. Partial redundancy arises from don't-care conditions internal to the circuit.

Simulation based functional validation uses a set of vectors that are applied to the design and compared to the expected results. Since a design containing partially redundant logic may respond correctly to a set of test vectors, various approaches have been suggested that use verification tools to pinpoint and remove such logic. Ratchev *et al.*, use the technique of forced error detection which is based on allowing the synthesis tool to invert one randomly selected logic signal in the netlist [10]. Huang *et al.* developed a tool that can detect don't-care logic while performing static property checking using ATPG and binary decision diagram (BDD) techniques [4].

A reversible gate/circuit maps each input pattern to a unique output pattern and does not therefore result in the erasure of any information. Pioneering work by Bennett, Feynman, Fredkin, and Toffoli investigated the potential of reversible logic to create circuits that theoretically achieve zero internal power dissipation. Reversible logic circuits are formed by combining gates in a cascade fashion using various synthesis methods that aim to

minimize garbage outputs and ancillary inputs [6]. A simple cascade of a 4 variable circuit composed of 4 gates is shown in Fig. 1. Each vertical line represents a *t*-variable Toffoli gate with *t*-1 control lines (filled circles) and one target line (open circle). For each gate, the value on the target line is negated if, and only if, all the *t*-1 control lines are set at '1'. We denote a *t*-variable Toffoli gate as TOF(x_1,x_2,…,x_{i-1},$x_{i+1,\ldots}$,x_n;x_i), where the target line x_i is separated by a semicolon from the control lines.

Each gate is part of a stage of the cascade which must include the gate and all the unconnected lines running through it. The overall transformation matrix *C* of this cascade is obtained by multiplying the transformation matrices of the 4 gates in reverse order. Thus,

$$C = G4 \times G3 \times G2 \times G1 \quad (1)$$

Definition 1 The *trivial identity gate* is a reversible gate that implements the identity transformation matrix *I* (a diagonal matrix with all non-zero entries equal 1). Any reversible gate with a transformation matrix that differs from the identity matrix is a *non-trivial* gate.

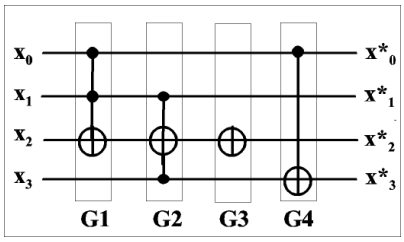

Figure 1. A cascade of 4 gates with 4 variables

Identity gates are trivial in the sense that they do not change the circuit transformation matrix and can be ignored in a functional sense. We consider only non-trivial gates when discussing reversible cascades.

Agrawal noted in the early 80's that fault detection is improved when the output information content of the tested circuit is maximized [1]. Reversible circuits, which implement bijective functions, are clearly maximized. Accordingly, Patel *et al*. showed that reversible circuits require fewer test vectors for multiple faults based on the stuck-at model compared to classical circuits [9].

Miller and Thornton proposed the QMDD structure to specify and simulate reversible and quantum logic circuits in a compact form [7]. QMDD sizes were further reduced using sifting-based variable reordering by Miller *et al.* [8].

With radix r=2, a matrix *M* of dimension [$2^n \times 2^n$] can be decomposed into four matrices $M_0,..,M_3$ such that:

$$M = \begin{bmatrix} M_0 & M_1 \\ M_2 & M_3 \end{bmatrix} \quad (2)$$

A QMDD applies this partitioning in the same way a reduced ordered binary decision diagram (*ROBDD*) [2] recursively applies Shannon decompositions. In a similar manner to ROBDDs, a QMDD adheres to a fixed variable ordering and common substructures (submatrices) are shared. A QMDD has a single terminal vertex with value 1, and each edge in the QMDD, including the edge pointing to the start vertex, has an associated complex-valued weight.

Theorem 1 *An $r^n \times r^n$ complex valued matrix M has a unique (up to variable reordering or relabeling) QMDD representation.*

Proof: A proof by induction based on the iterative construction of a QMDD and the normalization of edge weights that is performed during that construction is detailed in [7].

In this paper we distinguish between identical and equivalent circuits in accordance with the following definition.

Definition 2 Two circuits are *identical* if they comprise exactly the same gates and, correspondingly, the same netlist. Two circuits are functionally *equivalent* if they realize the same function. For reversible logic circuits, we further note that identical and equivalent circuits are represented a single unique transformation matrix.

Corollary 1 Two *$r^n \times r^n$ complex valued matrices M_1 and M_2 are identical if, and only if, they have the same start edge with the same weight in their QMDD representations.*

Proof: The proof follows directly from the canonicity of the QMDD (Theorem 1).

By Corollary 1, the QMDD package enables highly efficient equivalence checking of two reversible circuits.

3. Circuit Modification Strategy

We have developed two tools for detecting partially redundant logic. The "QMDDceq" for reversible logic uses a QMDD-based simulation tool, while the "CMBtest" tool for irreversible logic is based on the well-known CUDD package [11].

Both tools are used to perform repeated equivalence checking between a benchmark and a copy of the benchmark with controlled circuit modifications. Exhaustive structural tests that scan all the gates of the benchmark are possible only for small circuits. Larger circuits require random selection approaches. Fig. 2 shows the operational flow of our tools.

The gate modifications are quite different for irreversible and reversible logic gates and will be described separately.

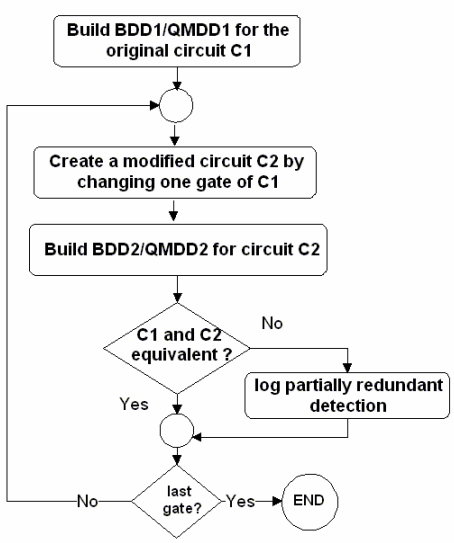

Figure 2. Detection of partially redundant logic

3.1. Gate Replacement for Irreversible Logic

We have defined various gate replacements for the ISCAS85 combinational benchmarks. These replacements are combined in three groups:

Negation - AND↔NAND; OR↔NOR; XOR↔XNOR.

Moderate1 - AND→NOR; NAND→XOR; OR→NAND; NOR→XNOR; XOR→OR; XNOR→AND.

Moderate2 - AND→OR; NAND→NOR; OR→XOR; NOR→AND; XOR→NAND; NOR→NOR.

Clearly, the negation replacements are the most severe. We expect the negated gates to be more easily detected by the equivalence checking tools.

3.2. Gate Replacement for Reversible Logic

We now prove several lemmas that illustrate the major differences between reversible and irreversible circuits.

Lemma 1 *Two reversible circuits (cascades) that are different by exactly one (non-trivial) gate cannot be equivalent.*

Proof: Let us assume that the two cascades C_1 and C_2 each consist of n gates, represented by the transformation matrices $G_1, G_2, ..., \boldsymbol{G_k}, .., G_n$ and $G_1, G_2, ..., \boldsymbol{G^*_k}, .., G_n$ respectively. Let $\boldsymbol{G_k} \neq \boldsymbol{G^*_k}$ be the only different gates in these circuits. Then by (1),

$$C_1 = G_n x G_{n-1} x ... x G_{k+1} x G_k x G_{k-1} x ... x G_2 x G_1 \text{ and}$$

$$C_2 = G_n x G_{n-1} x ... x G_{k+1} x G^*_k x G_{k-1} x ... x G_2 x G_1 .$$

The inequality $C1 \neq C2$ follows from the associative rule of matrix multiplication.

Lemma 1 immediately illustrates a major difference of reversible logic from irreversible logic – modifying a single gate is guaranteed to be detected. This requires that QMDDceq must modify two or more gates during each iteration in order for partially redundant logic to be created.

Lemma 2 *The removal of a single gate from one cascade of a pair of identical reversible cascades ensures their non-equivalence.*

Proof: The proof follows directly from Lemma 1 by substituting the removed gate with the identity gate.

Lemma 3 If a pair of equivalent cascades C_1 and C_2 remain equivalent after the removal of a contiguous set of l gates from C_2, then the removed set comprises a cascade representing the identity matrix.

Proof: It is easy to see that after removing l gates from C_2, the remaining equivalence condition $C1=C2$, namely

$$C_1 = G_n x ... x G_k x ... G_{k-l+1} x ... x G_1 = G_n x ... x G_{k+1} x G_{k-l} x ... x G_1 = C_2$$

can be satisfied only if $G_k x G_{k-1} x ... x G_{k-l+1} = I$

due to the properties of matrix multiplication.

The synthesis tools for reversible logic often employ template matching for circuit reductions [6,10]. Obviously, the condition of Lemma 3 should be easily detected and removed from the benchmarks by such a synthesis tool. Nevertheless, we can now prove Theorem 2 that sets the limit for partially redundant logic in reversible circuits.

Theorem 2 *A reversible logic cascade C of n non-trivial gates may contain redundant logic comprising 2 to n gates.*

Proof: By Lemma 1, the redundant logic cannot comprise a single non-trivial gate. By Lemma 3, we can provide an example of a redundant logic section of arbitrary size.

It is the prime motivation of this research to find hidden replacements in reversible logic when non-contiguous sets of gates are modified. A severe gate modification is achieved when we replace the type of gate altogether (e.g. TOF(x_0,x_1;x_2) ↔ V$^+$(x_0;x_2)). The deletion of a gate is essentially the replacement of the current gate type with the identity gate. A more moderate gate modification is achieved when changing the structure of a controlled gate with multiple lines like the Toffoli. For example, TOF($a,b,c;d$)≠TOF($a,d,c;b$) represents a change since the target line has been modified, as has one of the control lines.

4. Experimental Results

We have run the various gate replacements of the CMBtest tool on a number of ISCAS85 benchmarks as shown in Table 1. The table lists how many hidden replacements were detected using the Negation, Moderate1 and Moderate2 replacement rules. We can see that only one negation replacement is hidden in the benchmark C2670. On the other hand, there are many hidden replacements for the moderate gate replacements.

Some of these may indicate the presence of partially redundant logic.

Table 1. Hidden replacements in irreversible logic

Benchmark	gates/levels	Using Negation	Using Moderate1	Using Moderate2
C432	159 / 7	0	24	18
C880	382/ 12	0	57	20
C1355	545/ 15	0	416	2
C1908	879 / 17	0	327	0
C2670	1192 / 15	1	271	84
C5315	2306 / 25	0	475	184
C7552	3511 / 24	0	1059	244

We have tested the reversible logic benchmarks from D. Maslov's website [5] on our QMDDceq tool. These benchmarks include quantum as well as classical reversible logic benchmarks of generally "well synthesized" circuits. As expected, all our gate changes were ***promptly detected***, in sharp contrast to the irreversible logic case. Exhaustive gate modifications for sets of 2 to 6 gates were performed on smaller files.

Although we were unable so far to detect any partially redundant logic in these benchmarks, we should stress that our random gate selections tackle only a small portion of the search space. We cannot guarantee that some other gate selection changes will not reveal partially redundant logic in these benchmarks. Yet our efforts so far demonstrate that redundant logic in reversible circuits is less likely to occur than in irreversible classical logic.

Following Lemma 3, we were curious to determine if potential redundant logic in reversible circuits can appear in the form of non-continuous gates. For that purpose, we have generated several random reversible logic circuits **without** using template matching (as described in [6]) to optimize our design. Fig. 3 demonstrates one of our findings with a circuit we call "random4".

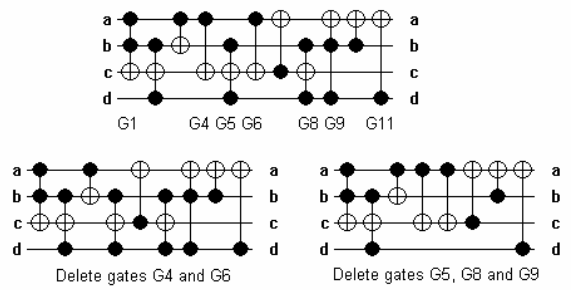

Figure 3. Results with file "random4"

This reversible circuit (shown in Fig. 3 using Maslov's RCviewer tool [5]) of 11 gates captures hidden replacements using the exhaustive gate deletion. We found that deleting gates {G4,G6} or {G5,G8,G9} does not alter the functionality of the circuit.

5. Conclusions

This paper has considered partially redundant logic detection and related gate replacement for reversible and irreversible logic. We have implemented automated tools that search for partially redundant logic using exhaustive and random selections along with symbolic equivalence checkers. This topic is very important with respect to the development of synthesis optimization tools, test–set generators, and the understanding of functional equivalence tools.

The experimental results demonstrate that reversible logic is less likely to exhibit redundant logic than irreversible logic, since reversible logic cascades are maximally connected. In contrast, irreversible logic tends to contain don't-care cones that are more prone to contain redundant logic.

The theoretical discussion shows that redundant logic may still appear in reversible logic. While so far we have detected redundant reversible logic in randomly generated circuits, we continue to enhance the gate replacement types of the QMDDceq tool and attempt to discover the detection of redundant logic in the benchmark reversible circuits.

References

[1] V. D. Agrawal, "An information theoretic approach to digital fault testing,", In *IEEE Trans. Comp,*. Vol. 30, pp. 582-587, 1981.

[2] R.E. Bryant. "Graph-Based Algorithms for Boolean Function Manipulation*"*. In *IEEE Trans. Comp*, C-35(8):677-691, 1986.

[3] D. Y. Feinstein, V.S.S. Nair, and M. A. Thornton, "Advances in Quantum Computing Fault Tolerance and Testing", In *Proc of High Assurance System Engineering Symposium,*. pp. 369-370, Nov. 2007.

[4] C-Y. Huang, B. Yang, H-C Tsai and K-T Cheng, "Static Property Checking Using ATPG v.s. BDD Techniques", *2000 Proceeding of International Test Conference*, pages 309-316. 2000.

[5] D. Maslov. Reversible logic synthesis benchmarks page. Online: http://www.cs.uvic.ca/~dmaslov/, Nov. 15, 2005.

[6] D**.** M. Miller**,** D. Maslov, and G. W. Dueck. "A transformation based algorithm for reversible logic synthesis". In *DAC'03*, pp. 318-323, 2003.

[7] D.M. Miller and M.A. Thornton, "QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits", In *ISMVL'06*, on CD, May 17-20, 2006.

[8] D. M. Miller, D. Y. Feinstein, and M. A. Thornton, "QMDD Minimization using Sifting for Variable Reordering", In *Journal of Multiple-valued Logic and Soft Computing*. pp. 537-552., 2007.

[9] K.N. Patel, J.P. Hayes, and I.L. Markov, "Fault testing for reversible logic", *In IEEE Trans. on CAD of IC and Systems*, Vol. 23, No. 8, pp. 1220-1230, August 2004.

[10] B. Ratchev, M. Hutton, G. Baeckler and B. van Antwerpen, "Logic synthesis and mapping: Verifying the correctness of FPGA logic synthesis algorithms*"*, In *Proc ACM/SIGDA 11th international symposium on Field programmable gate arrays***,** Feb. 2003

[11] F. Somenzi, *"The CUDD Package"*, University of Colorado at Boulder, Version 3.2.1 available at: http://vlsi.colorado.edu/~fabio/.