Redundancy Removal in Combinational Circuits

Felipe Marques, Marcelo Lubaszewski, André Reis
felipem@inf.ufrgs.br
luba@iee.ufrgs.br
andreis@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul – UFRGS
Instituto de Informática
Cx. Postal 15064 - Av. Bento Gonçalves, 9500 - Campus do Vale - Bloco IV
Bairro Agronomia - Porto Alegre - RS - Brasil - 91501-970

Abstract:

This paper discusses the differences between two methods [KEU 91][REI 00] for redundancy removal in combinational circuits, exploring the relationship between performance optimization and testability. Performance optimizations aimed at reducing logic depth, may introduce stuck-at fault redundancies in the circuit. Old redundancy removal methods like [BRA 83] can result in slower circuits. Both methods [KEU 91] [REI 00] return an irredundant circuit without delay penalty.

1 Introduction

Defects can be introduced during the fabrication process of an integrated circuit. Due to this reason, circuits that are produced must be tested, to avoid that bad products arrive to the consumer market.

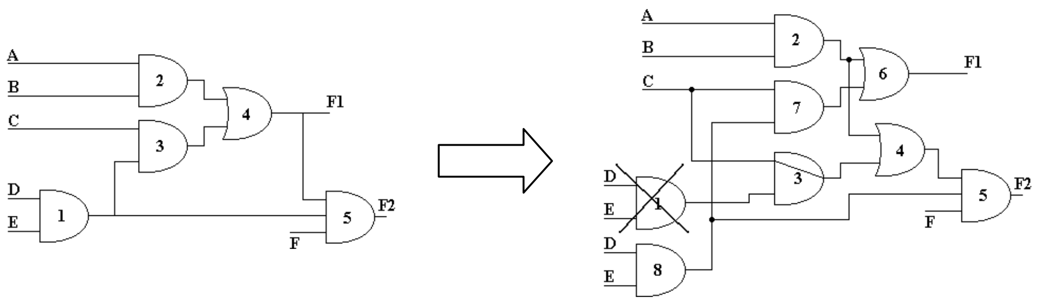
These tests are based on theoretical fault models. One of the most popular models is the stuck-at fault model. The occurrence of non-testable stuck-at faults can harm the performance of the circuit. The circuit can operate more slowly when the faults occur in redundant connections (non-testable faults). On the other hand, the logical behavior changes when faults occur in non-redundant connections. Many faults do not modify the logical behavior of the circuit. In this case, the static tests do not detect the all existent faults. The non-testable faults can have three origins: 1) fault tolerant designs, 2) timing optimizations 3) design not optimized. Redundancy removal is not desired in the first case. In the two last cases, it is possible to apply a redundancy removal method. Due to this reason, it is desirable to create methods to eliminate stuck-at faults of combinational circuits without introducing delay penalties.

Some logical synthesis methods are used to prevent the occurrence of those faults. These processes attempt to optimize three criteria: area, performance and testability. We are concerned with the relationship between performance and testability. Timing optimizations aimed at reducing logic depth (performance improvement) may introduce stuck-at fault redundancies in the design. In almost all cases the removal of these redundancies does not affect the speed of the circuit. In this paper, we compare two redundancy removal methods where, given a combinational circuit, the algorithms return a circuit without redundancies and with better or equal timing performance.

2 Method 1 – KMS Algorithm

This redundancy removal method is particularly dedicated to the case of redundancies introduced due to performance optimizations. The KMS algorithm [KEU 91] derives a circuit without redundancies from a redundant circuit. The process guarantees that the speed of the circuit will not be decreased. The algorithm is not concerned with the area of the circuit, i.e., it may either increase or decrease the final area. All the operations of this process are realized in a combinational circuit that contains only basic primitives (basic logic gates).

Let us consider the longest path of the circuit, *P*. If *P* is not statically sensitizable, the KMS algorithm invokes a process for redundancy removal to derive a circuit without redundancies. This process can be applied in any path *P* that is not statically sensitilizable. If the longest path is not statically sensitizable, the algorithm chooses another path, P_i and attempts to remove a redundant fault on the first edge of P_i (circuit input). If P_i is fanout-free, i.e., each gate along P_i has *fanout = 1*, then such a redundant connection always exists. If P_i is not fanout-free, duplication of some gates along of P_i (duplicated gates include those between the first edge of P_i and the last gate along P_i with *fanout > 1*) is done to obtain a corresponding path P_i' that is fanout-free. The first edge of P_i' is now redundant and is removed. It is proven in [KEU 91] that redundancy removal on the first edge of a longest path does not increase the delay of the circuit. Therefore, fault elimination does not increase circuit delay.



a) redundant circuit b) irredundant circuit

FIGURE 1 – The result of KMS algorithm.

The longest path of the circuit of figure 1a is *P (1, 3, 4, 5)*. That implementation contain four logical levels. Applying the KMS algorithm will duplicate the all gates of *P* with *fanout > 1*. Observing the figure 1b, we can notice that three gates are added in the circuit (*6*, *7* and *8*). The circuit of the figure 1b is an irredundant circuit. The performance of that circuit is better than the circuit of the figure 1a. The redundancy removal also improves the testability of the circuit.

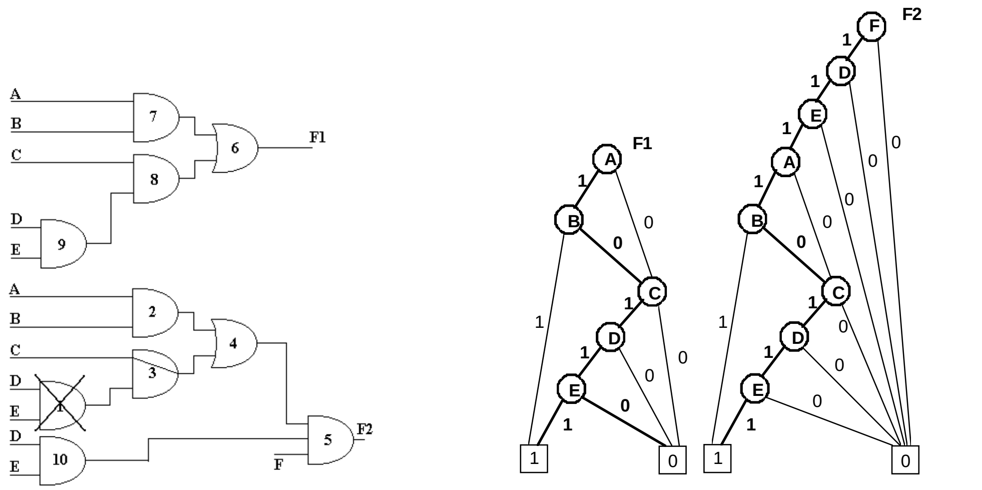
An expected drawback of the KMS algorithm is the number of iterations that are performed before the algorithm terminates. Hence, it does not complete on circuits with large numbers of false paths.

Another problem of this method is the dependence on external processes for fault identifications. The ATPG process is necessary to identify non-testable faults. Afterwards, the non-testable fault list is used in the redundancy removal process.

3 Method 2 – RPL Algorithm

This method use BDDs (Binary Decision Diagrams) to perform redundancy removal. These data structures represent a Boolean network. In the same way that KMS algorithm, the RPL algorithm receives a redundant combinational circuit and returns a circuit without redundancies and without introducing delay penalties. The main advantage of this algorithm is the coupling between the synthesis and the ATPG process over the same data structure.

The RPL algorithm [REI 00] uses a kind of BDD called Vertex Precedent BDDs in the fault identification process. These data structures preserve circuit testability properties. Thus the algorithm may analyze the VPBDDs to identify non-testable stuck-at faults. This process is similar to the KMS algorithm. However, the algorithm duplicate the all gates of the circuit with *fanout > 1*. This way, the process will have computational problems.



a) final circuit b) BDD of *F1* c) BDD of *F2*

FIGURE 2: Final circuit with all duplicated gates and BDDs representation.

For the circuit of figure 1a, the RPL method returns the circuit of figure 2a. Five gates were added (6, 7, 8, 9 and 10) after duplication process. At this point, we can evidence that the number of gates duplicated is almost the double of the number of gates duplicated by the KMS algorithm.

The next step is redundancy removal. The BDD of *F1* (figure 2b) does not present redundancies. However the BDD of *F2* (figure 2c) has some redundancies. The vertex *C* is only reachable if the inputs *F*, *D* and *E* are equals to 1 and A or B are equals to 0. Therefore, if *C* is equal to 1 then *F2* is equal to 1. The last two vertexes of this BDD are not necessary and they can be eliminated (the gate 1 of the figure 2a).

The high memory consumption is a great problem of the RPL method. The duplication process uses a significant memory amount. Hence, the RPL algorithm is only used to re-synthesize small circuits. On the other hand, the use of BDDs makes the process of fault identification easier.

4 Conclusions

This paper compares two methods for redundancy removal in a combinational circuit. Both methods exploit the relationship between testability and performance. Those methods use a great amount of memory to compute a circuit and usually exhaust the available memory due to logic duplication.

A proposal to improve the KMS algorithm is presented in [SAL 94]. The new algorithm performs efficient removal of long false paths in a circuit. However, this algorithm has a dependency on an external process for fault identification. The RPL algorithm introduces a method that uses Vertex Precedent BDDs. The use of a BDD data structure results in a straightforward process to identify faults during the synthesis. This is a significant advantage over the KMS algorithm. However the process for false path removal in the RPL algorithm has serious limitations due to execution time and memory consumption.

Future work will be centered on improving the RPL algorithm for obtaining better performance for execution time and memory consumption.

5 References

[KEU 91] KEUTZER, K.; MALIK, S.; SALDANHA, A. Is Redundancy Necessary to Reduce Delay? ***California: IEEE Transactions on Computer Aided Design***, vol. 10, n. 4, April 1991.

[REI 00] REIS, A.; PRADO, A.; LUBASZEWSKI, M. Testability Properties of Vertex Precedent BDDs, ***Amazonas: SBCCI2000 XIII Symposium on Integrated Circuits and System Design***, September 2000.

[SAL 94] SALDANHA, A.; BRAYTON, R.; SANGIOVANNI, A. Circuit Structure Relations to Redundancy and Delay. ***California: IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems***, vol. 13, n. 7, July 1994.

[BRA 83] BRAND, D.; Redundancy and Don’t Carer in Logic Synthesis. ***California: IEEE Transactions on Computer Aided Design***, vol. c-32, n. 10, October 1983.