Synthesis of Robust Delay-Fault Testable Circuits: Theory

Srinivas Devadas
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology, Cambridge

Kurt Keutzer
Computing Systems Research Laboratory
AT&T Bell Laboratories, Murray Hill

Abstract

Correct operation of synchronous digital circuits requires propagation delays of all sensitizable paths in the circuit to be smaller than a specified limit. Physical defects and processing variations in integrated circuits can affect the temporal behavior of a circuit without altering the logical behavior. These defects are called *delay faults*.

In order to design, and especially to synthesize, highly or completely delay-fault testable circuits, one has to fully understand the sources of untestability. Ideally this requires the determination of necessary and sufficient conditions for delay-fault testability. Synthesis procedures that produce testable logic realizations by satisfying sufficient conditions for delay-fault testability can then be based on this understanding. There is often an area/performance overhead associated with these procedures, but an understanding of both the sufficiency conditions and the techniques of logic optimization can alleviate the overhead by minimally constraining the optimization to achieve full testability.

In this paper we give a comprehensive theoretical framework for the analysis and synthesis of delay-fault testable combinational logic circuits. In particular, we provide necessary and sufficient conditions for a combinational logic network to be completely robustly path-delay-fault testable. We begin with two-level circuits and give a necessary and sufficient condition for robust path-delay-fault testability in terms of the well known concepts of primality, irredundancy and essential vertices. We then apply these notions to multilevel circuits to arrive at a necessary and sufficient condition for multilevel implementations. We also generalize these results to multiple-output functions. We show that *algebraic factorization* retains the robust path-delay-fault testability of a two-level circuit and this naturally suggests a synthesis procedure for multilevel robustly path-delay-fault testable circuits.

We switch focus to a less restrictive though common model of delay faults, namely, gate-delay-faults. We give a necessary and sufficient condition for two-level and multilevel functions to be completely robustly gate-delay-fault testable. We show that *constrained algebraic factorization* in a multilevel synthesis procedure retains the complete robust gate-delay-fault testability of a two-level network.

We present preliminary experimental results using these synthesis techniques.

1 Introduction

Correct operation of synchronous digital circuits requires propagation delays of all sensitizable paths in the circuit to be smaller than a specified limit – usually the clock interval. Physical defects in integrated circuits can sometimes degrade circuit performance without altering the logic functionality. These faults are called *delay faults*.

There are two main delay fault models, namely, the gate-delay model [20] [34] [28] and the path-delay-model [31] [23]. Under the gate-delay model, the fault delays a transition at a single node in the circuit. Under the path-delay-model, a fault is said to have occurred if the propagation delay along a path in the circuit is greater than the specified limit. The two models each have advantages and drawbacks. In the gate-delay model, delay faults resulting from the sum of several small incremental delay defects may not be detected. A drawback of the path-delay model is that if complete path-delay-fault testing is not performed, delay faults belonging to gates not in the set of tested paths are not detected. Furthermore, most circuits are not completely robustly path delay-fault testable.

Automatic test pattern generators have been proposed for both kinds of delay fault models (*e.g.* [22], [24], [23], [15], [12]). In order to have a high degree of confidence in delay tests, it is also valuable that the tests be valid under arbitrary delays in the circuit, and not be invalidated due to hazards (causing glitches in the output). While a high degree of such robust delay-fault coverage is desirable, a number of problems are associated with testing robustly in delay fault models. Due to these relatively stringent design requirements, it appears that an unacceptably large fraction of faults in circuits being designed today are *not* robustly testable [23], and the primary focus of this paper is to address this problem.

An obstacle to path-delay-fault testing is the potentially large vector sets required to test each path in robust manner. To address this problem a subset of paths may be chosen. These may include delay critical paths [31] or paths which cover all gates or nets in the circuit [21].

A remaining important barrier in the practical application of robust delay-fault testing is that robust testing currently requires a more expensive (non-standard) scan structure, such as that described in [24], and this means that the advantages of robust delay-fault testing must be weighed against the additional cost. Nevertheless, for many applications, quality and reliability are rising in importance and the relative importance of area, particularly in pad-limited designs, is decreasing. Furthermore, research is in progress to develop techniques which will allow the application of robust tests via a standard scan structure.

In order to develop efficient tests for delay faults in a circuit necessary and sufficient conditions for testability must be derived and automatic test-pattern generation procedures based on these conditions must be developed. Similarly, in order to design, and especially to synthesize, highly or fully delay-fault testable circuits, one has to completely understand the sources of untestability. Ideally, this also entails the determination of necessary and sufficient conditions for delay-fault testability. Synthesis procedures that produce testable logic realizations by satisfying sufficient conditions for delay-fault testability can then be based on this understanding. There is often an area/performance overhead associated with these procedures, but an understanding of both the sufficiency conditions and the techniques of logic optimization can minimize the overhead by minimally constraining the optimization to achieve full testability.

In this paper we give a comprehensive theoretical framework for the analysis and synthesis of delay-fault testable combinational logic circuits. We provide necessary and sufficient conditions for a combinational logic network to be completely robustly path-delay-fault testable. We begin with two-level circuits and give a necessary and sufficient condition for robust path-delay-fault testability in terms of the well known concepts of primality, irredundancy and essential vertices. We then apply these notions to multilevel circuits to arrive at a similar condition for multilevel implementations. We generalize these results to multiple-output functions.

Necessary and sufficient conditions for robust path-delay-fault testability were given in [23]. These conditions were motivated by the desire to develop automatic test-pattern generation procedures for robust path-delay-faults and were entirely adequate for that purpose. On the other hand these conditions do not bring to light any global circuit properties latent in robust path-delay-fault testable circuits and as a result using these conditions we found it difficult to show how robust path-delay-fault testability is retained by non-trivial circuit transformations. We show that *algebraic factorization* [2] retains the path-delay-fault testability of a two-level circuit.

We then switch our focus to the less stringent gate-delay-fault model. We give a necessary and sufficient condition for two-level and multilevel functions to be completely robustly gate-delay-fault testable. We show that *constrained algebraic factorization* is required to retain complete gate-delay-fault testability beginning from a two-level network.

In summary, for each of the common models of delay-fault testability, robust gate-delay-faults and robust path-delay-faults, we provide necessary and sufficient conditions for complete testability under that model for two-level circuits. We describe these in terminology common to two-level minimization and show their relationship to properties produced by two-level minimizers. We then show similar conditions for multilevel networks. Where convenient we deal with the synthesis issues involved in realizing the delay-fault testable networks, but in general synthesizing completely robust path-delay-fault testable for both two-level and multilevel circuits, and robust gate-delay fault testability for multilevel circuits, requires a sophisticated orchestration of logic optimization and technology mapping techniques. We therefore defer a full treatment of these to a companion paper [9] and a preliminary version of the results of that paper has been published [7].

Basic definitions are presented in Section 3. In Sections 4 and 5, we deal with path and gate-delay-fault testability of two-level circuits, respectively. Conditions for path and gate-delay-fault testability in multilevel implementations are the subject of Sections 6 and 7, respectively. Preliminary experimental results are presented in Section 8.

2 Related Work

Because of the sense of growing importance of delay-fault testing there has been a great deal of recent research activity on the delay-fault model and we will only discuss that work that directly relates to synthesis procedures. In [4] conditions for robust testing in the stuck-open fault model are examined and circuit transformations that enhance testability in that model are given. A synthesis procedure for ensuring multifault testability is given in [14]. The determination of necessary and sufficient conditions for *nonrobust, adjacency* gate-delay-fault testability are given in [13], *i.e.* conditions for gate-delay-fault testability with the constraint that the applied vector pair differ in a single bit.

The path-delay-fault model was introduced in [31] and a number of properties relating to path-delay-faults are stated there. Design procedures that guarantee completely robustly path-delay-fault testable, multilevel circuits using unate decomposition techniques from two-level circuits were presented in [18, 19]. A heuristic procedure targeting the removal of reconvergent fanout structures to enhance path-delay-fault testability was given in [29] and a more deterministic procedure for removing untestable path-delay-faults was presented in [27]. Preliminary versions of this work were presented in [10] and in [6]. A similar line of work was independently pursued in [26] where necessary and sufficient conditions for *general* robust path-delay-fault testability are derived. Using these derived conditions an *extended* algebraic factorization procedure is shown to retain general robust path-delay-fault testability. The general robust path-delay-fault model is less restrictive than the robust path-delay-fault model because it does not require single event sensitization.

3 Definitions and Notation

Analysis and synthesis of delay-fault testable circuits requires spanning the fields of logic optimization, testing and timing analysis. In this section we try to provide a minimal amount of terminology from these fields.

3.1 Logic Synthesis

A **Boolean function** F of N variables is a mapping from $B^N = \{0,1\}^N \rightarrow \{0,1\}$. We model B^N as a Binary N-cube. A vertex $v \in B^N$ for which $F(v) = 1$ is a member of the ON-set. If $F(v) = 0$ then v is a member of the OFF-set.

A **literal** is a Boolean variable or its complement. We formally define a **cube** as a set of literals and we interpret the cube as a product of literals. For example $\{a, b, c\}$ is a cube, interpreted as $a \cdot b \cdot c$ which may be abbreviated to abc. A **minterm** is a cube in which every variable in the Boolean function appears. We may interpret the minterm as a **vertex** in the N-cube.

Minterms and cubes may be used to represent the values of a set of input variables: *e.g.* $x\overline{y}z$ is shorthand for $x = 1,\ y = 0$, and $z = 1$. In this way there is a natural correspondence between an input vector or input stimulus, a minterm and a vertex in the N-cube. This correspondence may be extended to cubes where unspecified values in the function are assumed to be undefined values. Thus, if a circuit C has inputs $v,\ w,\ x,\ y,\ z$ then applying the cube $x\overline{y}z$ to C is shorthand for applying $v = X,\ w = X,\ x = 1,\ y = 0$, and $z = 1$. (Here X denotes an unknown value.) We define the distance δ between a cube q and a cube r as the cardinality of the set: $\{l | (l \in q) \wedge (\overline{l} \in r)\}$. For example, the distance between abc and $ab\overline{c}$ is 1; the distance between ab and abc is 0; and the distance between $b\overline{c}def$ and $abcf$ is 1.

Following the historical usage we define a **cover** as a set of cubes and we interpret the cover as a sum-of-products expression. For example $\{\{a, b, c\}, \{d, e, f\}\}$ is a cover, interpreted as $abc + def$. We say that a cube q **covers** a cube (or vertex) r if $q \subseteq r$. If a cube q covers only ON-set vertices of a Boolean function F then q is called an **implicant** of F. A **relatively essential vertex** of a cube q in a cover C is a vertex that is covered by q and is not contained in any other cube in C.

3.2 Testing

A Combinational Logic Circuit is represented as a labeled, directed, acyclic graph (DAG) $G = (V, E)$ with each vertex v labeled with the name of a primitive gate such as AND, OR or NOT, or with the name of a primary input or output. In general, a combinational logic circuit can be created from gates of arbitrary complexity, but to simplify the discussion we assume that the circuit is expressed in terms of primitive gates. There is an edge $< u,\ v >$ in V between two vertices if the output of the gate associated with u is an input to gate v. The members of V that have no fan-in are the only vertices that may be labeled with the name of a primary input. The members of V that have no fan-out are the only vertices that may be labeled with the name of a primary output.

If the output of a gate, g_1, is connected to an input of gate, g_2, g_1 is a **fanin** of g_2. Gate g_2 is a **fanout** of gate g_1.

A **two-level circuit** is a circuit directly implementing a *cover* by implementing each product term by an AND gate, the sum term by an OR gate, adding inverters on inputs as needed. We consider only non-trivial two-level circuits in which no AND gate is fed both by a signal and its complement (*e.g.* $a \cdot \overline{a}$).

A single-output combinational logic circuit computes a Boolean function in the obvious way. An N input M output combinational logic circuit computes a vector-Boolean function from $B^N \rightarrow B^M$. As we will be dealing primarily with combinational logic circuits in this paper, we will use circuit to mean combinational logic circuit.

A circuit C is **completely single-stuck-at-fault testable** if for each stuck-at fault f there exists an input stimulus t such that the output of the circuit on applying t in the presence of f is different from the output of C without the fault f. **Completely multiple-stuck-at-fault-testable** can be defined similarly.

If a single fault is stuck-at-1 untestable then the edge (net) in the circuit associated with that fault can be replaced by a constant 1. Similarly if a single fault is stuck-at-0 untestable then the edge (net) in the circuit associated with that fault can be replaced by a constant 0.

For a fuller treatment of testing terminology, see [3] [11].

3.3 Timing/Delay Testing

In this section we introduce terminology that will allow us to discuss timing issues as well as fault models for temporal behavior. We will introduce a number of terms, but hopefully the resultant clarity regarding the concepts will be worth the price. A **path** in a combinational circuit is an alternating sequence of vertices and edges, $\{g_0, e_0, ..., g_n, e_n, g_{n+1}\}$, where edge e_i, $1 \leq i \leq n$, connects the output of vertex g_i to an input of vertex g_{i+1}. For $1 \leq i \leq n$, g_i is a gate; g_0 is a primary input and g_{n+1} is a primary output. Each e_i is a net.

An **event** is a transition $0 \rightarrow 1$ or $1 \rightarrow 0$ at a gate. Consider a sequence of events, $\{r_0,\ r_1,\ ...,\ r_n\}$ occurring at gates $\{g_0,\ g_1,\ ...,\ g_n\}$ along a path, such that r_i occurs as a result of event r_{i-1}. The event r_0 is said to propagate along the path. If there exists an vector pair such that under *appropriate* delays in the circuit, an event could propagate along a path, then the path is said to be **event sensitizable**. If there exists an input vector pair such that under *arbitrary* delays in the circuit, an event propagates along a path, then the path is said to be **single event sensitizable**.

A **controlling value** at a gate input is the value that determines the value at the output of the gate independent of the other inputs. For example, 0 is a controlling value for an AND gate. A **non-controlling value** at a gate input is the value which is not a controlling value for the gate. For example, 1 is a non-controlling value for an AND gate. We say a gate g has a **controlled value** if one of its inputs has a controlling value; otherwise, we say g has a **non-controlled value**. For the purposes of the discussion here, we restrict our attention to networks comprised of simple gates, since the notion of controlling and non-controlling values for such gates is intuitive. However, this notion can be easily generalized to networks containing complex gates.

Let $\pi = \{g_0, e_0, ..., g_n, e_n, g_{n+1}\}$ be a path. The inputs of g_i other than e_{i-1} are referred to as the **side-inputs** to π.

Let $\pi = \{g_0, e_0, ..., g_n, e_n, g_{n+1}\}$ be a path in a circuit C. We say that an input vector w **sensitizes to a 1** path π in C iff the value of g_{n+1} is 1 and for each g_i, $1 \leq i \leq n+1$, if g_i has a controlled value then the edge e_{i-1} presents a controlling value. Similarly, we say that an input vector w **sensitizes to a 0** path π in C iff the value of g_{n+1} is 0 and for each g_i, $1 \leq i \leq n+1$, if g_i has a controlled value then the edge e_{i-1} presents a controlling value.

We may now observe that for a path π in a circuit C to be *single event sensitizable* (to a 1) it is necessary that there exist a vector pair $< w_1, w_2 >$ such that w_1 *sensitizes to a 0* the path π and w_2 *sensitizes to a 1* the path π. Moreover, all side-inputs along π must have (the same) non-controlling values for both w_1 and w_2, and at each gate g_i along π if the value of g_i on w_1 was the controlled value then the value of g_i on w_2 is the non-controlled value and if the value of g_i on w_1 was the non-controlled value then the value of g_i on w_2 is the controlled value. Changes on the side-inputs would imply that the path is not single event sensitized, *i.e.* event sensitized under arbitrary delays.

A circuit has a **gate-delay-fault** iff there is one gate in the circuit such that the output of the circuit is slow to make a $0 \rightarrow 1$ (or $1 \rightarrow 0$) even when one or more of the gate's inputs change values. Each single gate-delay-fault is assumed to be so catastrophic as to cause a delay along any path to any output [20]. In this paper we call a **robust test** for a gate-delay-fault a test which is valid under arbitrary delays and is not invalidated by glitches, hazards or races.

A circuit has a **path-delay-fault** iff there exists a path from a primary input to a primary output via a set of gates and nets such that a primary input event is slow to propagate *along the path* to the primary output [31].

In this paper we call a **robust test** for a path-delay-fault on a path π, a test which *single event sensitizes* π and is therefore valid under arbitrary delays and is not invalidated by glitches, hazards or races. Such tests were called *restricted delay test pairs* in [30] and *single-path propagation hazard-free robust tests* in [26]. In the latter reference a complete taxonomy of path-delay-fault tests are given. Because we will only be dealing only with robust path-delay-faults we use this abbreviated term without ambiguity. Robust path-delay-fault testability is a most stringent testabilty requirement for a circuit, but it is for this very reason that it poses the greatest challenge, and therefore the greatest interest, for synthesis techniques.

Unless noted otherwise we will assume that a robust test for a delay fault in a circuit C is a vector pair $< v_1,\ v_2 >$ such that $C(v_1) = 0$ and $C(v_2) = 1$. Let the expected transition time on the vector pair be τ. The application of the vector pair is as follows: Vector v_1 is applied to C and the values on nets are allowed to settle for an arbitrary amount of time. Vector v_2 is then applied to C. At time τ the output of C is sampled; if the value is 1 then no fault is detected, otherwise a fault is detected. Next, the vector pair $< v_2, v_1 >$ is applied to propagate the opposite event along the path and detect faults corresponding to the $1 \rightarrow 0$ event.

4 Path-Delay-Fault Testable Circuits with Two-level Logic Implementations

4.1 Introduction

Of the two delay fault models that we consider, robust path-delay fault testing is the more comprehensive. In this section we focus on path-delay-fault testing in two-level logic circuits. We give necessary and sufficient conditions for robust path-delay-fault testability of single-output circuits, and then generalize these results to the multiple-output case.

4.2 Conditions for Single-Output Functions

Paths in two-level circuits are very simple. They begin at a primary input, may pass through an inverter, pass through an AND gate and finally an OR gate. Because of the correspondence between a two-level circuit and a cover, a primary input of a circuit and a literal in a cube, a gate in the circuit and a cube in the cover we will use phrases such as "Let C be a two-level single-output circuit and let π be a path in C that starts with literal l in cube q" as shorthand for: "Let C be a two-level single-output circuit implementing cover E and let π be a path in C that starts with primary input i and passes through gate g such that i corresponds to literal l in cover E and g corresponds to cube q in cover E".

For notational convenience, we denote the value of a circuit C on a vector v by $C(v)$. For example the circuit $abc + def$ evaluates to 1 on the vector $abcdef$. We denote the Boolean value of a cube q on a vector v by $q(v)$. For example given the cube $q = abc$, $q(abcdef) = 1$. Similarly, we denote the Boolean value of a literal l in a cube q on a vector l by $q_l(v)$. For example given the cube $q = abc$, $q_b(abcedef) = 1$.

We now proceed with a lemma that gives a necessary condition for a path in a two-level single-output circuit C to be robustly testable.

Lemma 4.1 : *Let C be a two-level single-output circuit and let π be a path in C that starts with literal l in cube q. Let $D = C - q$. If $< v_1, v_2 >$ is a delay-fault test for π in C then for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$.*

Proof: Suppose there does not exist any such literal m both in v_1 and v_2. Then there exists at least one cube e in D such that for every literal l in v_1 and v_2 either $e_l(v_1) = 1$ or $e_l(v_2) = 1$. Thus under the appropriate input delays, the cube e can evaluate to a 1 causing a glitch and invalidating a delay-fault test for π. ■

We now give a theorem describing necessary and sufficient conditions for a two-level single-output circuit to be robustly path-delay-fault testable.

Theorem 4.1 : *Let C be a two-level single-output circuit. Let π be a path in C that starts with l in cube q. Let $D = C - q$. There exists a delay-fault test for π in C iff there exists a vertex v_2 that is a relatively essential vertex of q and a vertex $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$ such that v_1 is in the OFF-set of C.*

Proof: **Necessity**: Suppose $< w_1, v_2 >$ is a delay-fault test for π. Suppose v_2 is not a relatively essential vertex of q. Then there exists a cube d in D such that $d(v_2) = 1$. This implies that the path π is not the only path sensitized to a 1 by v_2. This would imply that π is not the only gate that supplies a controlling value to the OR gate at the output of the circuit, and this contradicts the definition of robust path-delay-fault testability. Thus, v_2 is a relatively essential vertex of q.

We now wish to show that while the vector w_1 need not be distance-1 (in literal l) from v_2, we can always construct a vector v_1 such that v_1 is in the OFF-set distance-1 from v_2 in literal l and $< v_1, v_2 >$ is a delay-fault test for π. By Lemma 4.1 for every cube d in D there exists some literal m in both w_1 and v_2 such that $d_m(w_1) = 0$ and $d_m(v_2) = 0$. Let the set of all such literals be M. By definition, all literals in M are the same in w_1 and v_2 and $D(w_1) = D(v_2) = 0$. Let $v_1 = v_2 - \{l\} \cup \{\overline{l}\}$. The literal l is not in M because it cannot be the same in w_1 and v_2; the literal l must change values to propagate an event along π. So $M \subseteq v_2$ and $M \subseteq v_1$. Therefore $D(v_1) = 0$ and because v_1 contains $\overline{l}$ and $q_{\overline{l}}(v_1) = 0$, so v_1 is in the OFF-set of C. Thus the existence of vertices v_1 and v_2 is necessary for a delay-fault test to exist for π.

Sufficiency: To show that the existence of these vertices is sufficient for a delay-fault test to exist for π we simply show that these vertices, when applied as a vector pair, constitute a delay-fault test for π. After applying v_1 and waiting an arbitrarily long amount of time the circuit output settles to a 0. When v_2 is applied it first causes the input net associated with l to rise. The other inputs to cube q are set to non-controlling values by both v_1 and v_2 because only l changes in q in v_1 and v_2. So the event propagates through q to the output of C. Thus these vertices may be used as a delay-fault test for π. ■

The existence of a vector pair $< v_1,\ v_2 >$ as above was also shown to be a sufficiency condition for robustly testing pFET TSOP faults at the output gate of an irredundant two-level circuit in the proof of Lemma 2 in [17].

4.3 Conditions for Multiple-Output Functions

The conditions of the previous section can be generalized to multiple-output two-level circuits. For multiple-output circuits we have very stringent requirements because we must verify that *every* path from an input to an output is robustly delay-fault testable. We can contrast this with the ordinary testing requirement that a stuck-at-fault fault is testable at *some* output. In particular, we require that each literal in each cube of a multiple-output cover be prime in *every* output, and that the distance-1 requirement between the relatively essential vertex and OFF-set also be satisfied for every output.

Lemma 4.2 : *Let C be a two-level multiple-output circuit and let π be a path in C that starts with literal l in cube q and ends at primary output O. Let $D = C_O - q$, where C_O represents the set of cubes in C that feed output O. If $< v_1, v_2 >$ is a delay-fault test for π in C then for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$.*

Proof: Similar to the proof of Lemma 4.1. ∎

We now give a theorem describing necessary and sufficient conditions for a two-level multiple-output circuit to be robustly path-delay-fault testable.

Theorem 4.2 : *Let C be a two-level multiple-output circuit. Let π be a path in C that starts with literal l in cube q and ends at primary output O. Let $D = C_O - q$. There exists a delay-fault test for π in C iff there exists a vertex v_2 that is a relatively essential vertex of q with respect to output O and a vertex $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$ such that v_1 is in the OFF-set of O.*

Proof: We consider the single-output circuit corresponding to output O in C and apply Theorem 4.1 to obtain the above result. ∎

5 Gate-Delay-Fault Testable Circuits with Two-Level Logic Implementations

5.1 Introduction

There exist Boolean functions for which no two-level circuit implementation is robustly delay-fault testable [9]. Gate-delay-fault testability represents an alternative testability criterion for such functions. Independently of one's position regarding the relative merits of gate-delay and path-delay-fault testability, it is interesting to examine the gate-delay-fault model for the additional insight that it gives into circuit properties that are generated and maintained by logic optimization procedures. Therefore, in this section we consider the less restrictive model of gate-delay-faults and develop necessary and sufficient conditions for a two-level circuit to be robustly gate-delay-fault testable.

5.2 Conditions for Single-Output Functions

We begin with a simple lemma that gives sufficient conditions for robust gate-delay-fault testability.

Lemma 5.1 *: Let C be a two-level single-output circuit and let g be a gate in C. If a path π through g is robustly path-delay-fault testable then g is robustly gate-delay-fault testable.*

Proof: Suppose there is a path π through g that is path-delay-fault testable. If there is any delay fault associated with g, be it a $1 \rightarrow 0$ or a $0 \rightarrow 1$ transition, it is reflected in the delay of π for the appropriate event propagating along π. Thus any path-delay-fault test for π is a gate-delay-fault test for g. ∎

We now derive necessary and sufficient conditions directly in terms of the two-level circuit structure. As before, given the direct correspondence between the $0 \rightarrow 1$ and $1 \rightarrow 0$ transition gate-delay-fault tests, we will concentrate on a single transition fault per gate.

Theorem 5.1 *: Let C be a two-level single-output circuit and let g be an* AND *gate in C. The gate g is robustly gate-delay-fault testable iff there exists a vector pair $< v_1, v_2 >$ such that v_2 is a relatively essential vertex of the cube q associated with g, and there exists a literal l in q such that $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$ is in the OFF-set of C.*

Proof: **Necessity:** We could apply a similar argument to the one used in the proof of Theorem 4.1, but we thought it would be enlightening to see a different argument for the same property. Suppose $< w_1, w_2 >$ is a gate-delay-fault test for g. The distance of this pair may be great but we will proceed to construct a distance-1 vector pair as in the theorem.

Let S be the set of variables that change from w_1 to w_2. Let $x = \{l | \; (l \in w_2) \wedge (l, \bar{l} \; \notin \; s)\}$, *i.e.* let x be the cube resulting from the elimination of all literals in S from w_2. First, observe that by the arguments of Lemma 4.1 under the appropriate delays on inputs, any vertex covered by x could be reached by applying the pair $< w_1, w_2 >$. This implies that the intersection of x with each d in $D = C - q$ is empty. If not, then under appropriate delays some vertex covered by another cube in D could be reached and a glitch would occur that invalidates the gate-delay test for q. Hence, what is covered by x? The cube x covers exactly members of the ON-set covered only by q, *i.e.* relatively essential vertices of q, and members of the OFF-set.

We wish to find two adjacent vertices in the subspace determined by x, such that one is in the ON-set of C and the other is in the OFF-set. How can these vertices be located? The circuit values appearing on the inputs of C as the vector pair $< w_1, w_2 >$ is applied may be visualized as a sequence of vertices forming a path in the Boolean N-space from vertex w_1 and vertex w_2. Furthermore this path is restricted to be in the subspace determined by the cube x. As vertex w_1 is in the OFF-set and vertex w_2 is in the ON-set, there exist adjacent vertices v_1 and v_2 at some point on the path such that v_1 is in the OFF-set and v_2 is in the ON-set.

Sufficiency: Suppose there exists a vector pair v_1, v_2 such that v_2 is a relatively essential vertex of the cube q associated with g, and there exists a literal l in q such that $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$ is in the OFF-set. By Theorem 4.1 this constitutes a robust path-delay-fault test for the path from the input associated with l through g to the output of C. Thus by Lemma 5.1 this constitutes a robust gate-delay-fault test for g. ■

The above theorem did not explicitly consider the single OR gate in the two-level circuit. All paths have to pass through the OR gate, and hence a gate-delay-fault at the OR gate is also robustly testable, if at least one AND gate is robustly testable. This is however, merely a sufficiency condition, and not a necessary one. The more general results of Section 7 can be used to determine necessary and sufficient conditions for the OR gate to be robustly gate-delay-fault testable.

5.3 Conditions for Multiple-Output Functions

We generalize the results of the previous section to multiple-output circuits.

Theorem 5.2 *: Let C be a two-level multiple-output circuit and let g be an* AND *gate in C. The gate g is robustly gate-delay-fault testable iff there exists a vector pair v_1, v_2 such that v_2 is a relatively essential vertex of the cube q associated with g, for some output O, and there exists a literal l in q such that $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$ is in the OFF-set of the output O. If g is an* OR *gate in C, asserting an output O, it is robustly gate-delay-fault testable if there exists an* AND *gate g_2, that feeds output O, such that g_2 is robustly gate-delay-fault testable.*

Proof: Similar to the proof of Theorem 5.1. ■

It has been conjectured that primality and irredundancy of a cube in a cover, is sufficient for the corresponding gate to be robustly gate-delay-fault testable. This would imply that each prime in the cover contains an relatively essential vertex that is distance-1 from the OFF-set. However, Touati and Watanabe [32] have given a two-level circuit that is prime and irredundant which contains a prime whose only relatively essential vertex is, in fact, distance-2 from the OFF-set. This same function is given as Figure 2-4 in [4]. Thus, as was the case for robust path-delay-fault testability modified covering strategies that satisfy necessary conditions for gate-delay-fault testability are required. Such a procedure is presented in [7, 9].

6 Path-Delay-Fault Testable Circuits with Multilevel Logic Implementations

6.1 Introduction

While the results derived for two-level circuits were of interest, multilevel circuits are of much greater utility. We therefore wish to extend the results of Section 4 to multilevel implementations and derive necessary and sufficient conditions for a multilevel circuit to be robustly path-delay-fault testable. To aid in our analysis our approach is to use an alternative representation of a multilevel circuit developed by Armstrong [1] called the *equivalent normal form* or ENF. The ENF of a circuit is a two level representation in which the sensitization conditions for each path is represented by a cube with each literal in the cube annotated by information regarding its path from the primary input of the circuit to the primary output. Similar representations were used in [33] for the analysis of hazards in asynchronous circuits and in [25] for the analysis of transients in circuits. We will define ENF more formally below.

Given a combinational circuit C with inputs $i_1, \ldots, i_m$, outputs $o_1, \ldots, o_n$ and labeled gates $g_1, \ldots, g_p$, we can construct the ENF for each output o_k of C by traversing the circuit forward from the inputs and visiting each gate in topological order. Each time we reach a gate that fans out to more than one destination we duplicate that gate as many times as necessary until each gate has only one fanout. Each duplicated gate retains the label of the original. When the output is reached we have created a DAG in which only the inputs have fanout greater than 1. Such a DAG is called a leaf-DAG. Each complex gate is then translated into path equivalent AND gates, OR gates and inverters. The next step is to push all inverters back from the primary output of the subcircuit to the primary inputs. During this transformation AND gates may be converted to OR gates and *vice versa*. There is a one-to-one correspondence between each path in C and each path in the leaf-DAG circuit. Finally, the ENF expression itself is computed as follows. Each variable in an ENF expression is simply a primary input of C, such as i_j, and has a label consisting of a set of gates denoting the path this input has taken to reach the output. The ENF for a primary input i_j is simply i_j. The intermediate ENF expression corresponding to the output of an AND gate g with inputs a_α, b_β is a sum-of-products expression over tagged literals, $E_g = a_{\alpha \cup g} \cdot b_{\beta \cup g}$. The expression E_g is created by reducing the product to sum-of-products form using DeMorgan's laws and distributivity without making any Boolean reductions such as $(a + \overline{a} \equiv 1)$, $(a + a \equiv a)$, $(a \cdot a \equiv a)$ or $(a \cdot \overline{a} \equiv 0)$. Similarly, the intermediate ENF expression corresponding to the output of an OR gate h with inputs a_α, b_β is a sum-of-products expression over tagged literals, $E_h = a_{\alpha \cup h} + b_{\beta \cup h}$. As before, no Boolean reductions are made in reducing E_h to sum-of-products form.

In Figure 1 we show a circuit based on Figure 2 in [1]. The final ENF expression for primary output M is:

$M = e_{\{5,9\}} \cdot b_{\{5,9\}} \cdot \overline{a}_{\{1,4,5,9\}} + b_{\{5,9\}} \cdot \overline{c}_{\{2,3,4,5,9\}} \cdot d_{\{3,4,5,9\}} \cdot e_{\{5,9\}} + a_{\{1,4,6,8,9\}} \cdot c_{\{2,3,4,6,8,9\}} \cdot \overline{b}_{\{7,8,9\}} + a_{\{1,4,6,8,9\}} \cdot c_{\{2,3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}} + a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{b}_{\{7,8,9\}} + a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$.

This ENF has been derived using the rules for the AND and OR gates given above.

Generating the ENF of a multilevel circuit may require, in the worst case, time and space exponential in the number of inputs to the circuit; however, the primary motivation for using the ENF to describe the necessary and sufficient conditions for testability is for clarity of explication and for the ease of proving, in Section 6.4, that synthesis procedures retain testability. ENF analysis is only used in proofs, it does not have to be explicitly calculated on each circuit.

As an aside, it is interesting to note that explicit ENF computation and analysis is possible for combinational circuits of significant size. These include virtually all multilevel circuits that have been synthesized from two-level representations (*c.f.* Section 8). Furthermore, the synthesis methodology of [7, 9] which is based on the theory developed here, only requires ENF computation and analysis for small subcircuits. Finally, it is worth noting that the size of the ENF does not correspond directly to the size of a circuit but is more strongly related to the number of paths in the circuit. Nevertheless, when a set of test patterns for delay-fault tests is not arrived at as a by-product of the synthesis process, we suggest the use of traditional automatic test-pattern generation methods (*e.g.* [12, 23]) and not the use of ENF analysis for arriving at the desired test-pattern set.

6.2 Conditions for Single-Output Functions

Armstrong developed the ENF to describe the conditions under which a single stuck-at fault is propagated from the input of a circuit to its outputs. In particular, he showed that a test for a literal l_π in an ENF expression, "sensitizes", in a way that will be made clearer below, the path π in the original circuit.

Let us consider the ENF expression M above. The existence of literal $a_{\{1,4,6,8,9\}}$ in the cube $a_{\{1,4,6,8,9\}} \cdot c_{\{2,3,4,6,8,9\}} \cdot \overline{b}_{\{7,8,9\}}$ indicates that a 1 is needed on primary input a to *sensitize to a 1* the path from input a along $a_{\{1,4,6,8,9\}}$. (If the cube contained $\overline{a}_{\{1,4,6,8,9\}}$, then a 0 would be needed to sensitize to a 1 the path from input a). In addition, to sensitize to a 1 the path $a_{\{1,4,6,8,9\}}$, a non-controlling value is needed at the other input of gate 4 so we need a 1 on primary input c.

This is reflected by the literal $c_{\{2,3,4,6,8,9\}}$. Finally, we need a non-controlling value on the other input to gate 8 and this is reflected by the literal $\overline{b}_{\{7,8,9\}}$. A non-controlling value on the other input to gate 8 could also be accomplished by setting primary input d to a 0. This is reflected by the existence of literal $\overline{d}_{\{7,8,9\}}$ in the cube $a_{\{1,4,6,8,9\}} \cdot c_{\{2,3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$. One way to sensitize a 1 along the path $b_{\{7,8,9\}}$ is reflected by the cube $a_{\{1,4,6,8,9\}} \cdot c_{\{2,3,4,6,8,9\}} \cdot \overline{b}_{\{7,8,9\}}$, *i.e.* by the partial assignment $< a = 1, b = 0, c = 1 >$. A vector may sensitize more than one path to a 1. This partial assignment is consistent with the partial assignment $< a = 1, c = 1, d = 0 >$. Thus the input vector $< a = 1, b = 0, c = 1, d = 0, e = 0 >$ sensitizes to a 1 both the paths $b_{\{7,8,9\}}$ and $d_{\{7,8,9\}}$.

Given a circuit C with ENF expression E, and a tagged literal l_π, the **cube complex associated with** l_π is the set of all cubes q such that $l_\pi \in q$ and there is no other tagged literal $l_\rho \in q$. The reason for excluding cubes that contain more than one literal of the same variable with different tags in the cube complex is due to the fact that the paths corresponding to l_π and l_ρ should not both be sensitized by a test for π, and will be elaborated on below.

6.3 ENF and Robust Path-Delay-Fault Testability

We wish to describe the conditions for *single event sensitization* in terms of ENF expressions. As we have seen, more than one path may be sensitized to a 1 by a vector, and to ensure that we have a single event sensitization on the pair $< v_1, v_2 >$ we must make sure that only one path is event sensitized by the pair. To accomplish this all other side-inputs along π must have the same values on both v_1 and v_2. One thing that will help ensure this is that we will only change the value of a single variable in $< v_1, v_2 >$ [31]. But simply changing only one variable will not be sufficient: One particular type of cube in the ENF requires special attention. In the cube $a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$ in the ENF expression M, the literal $\overline{d}$ appears twice. This indicates that the paths $d_{\{3,4,6,8,9\}}$ and $d_{\{7,8,9\}}$ beginning with primary input d are both sensitized to a 1 by the vector $< a = 1, b = 1, c = 0, d = 0, e = 0 >$ and reconverge at gate 8. [1] An input assignment that sensitizes to a 0 the path corresponding to $\overline{d}$ as reflected in the cube $a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$ is $< a = 1, b = 1, c = 0, d = 1, e = 0 >$. However, the vector sequence $<< a = 1, b = 1, c = 0, d = 0, e = 0 >, < a = 1, b = 1, c = 0, d = 1, e = 0 >>$ fails to single event sensitize a $1 \rightarrow 0$ event at the output because whichever path, $d_{\{3,4,6,8,9\}}$ or $d_{\{7,8,9\}}$, that presents a 0 to the input of gate 8 causes a $1 \rightarrow 0$ event on the output of gate 8, and a delay on the $1 \rightarrow 0$ transition on the other path will not be detected. Thus, in the selection of robust path-delay-fault tests for $d_{\{3,4,6,8,9\}}$ and $d_{\{7,8,9\}}$, input assignments that sensitize paths as reflected in

[1] Reconvergent fanout is the cause for the occurrence of cubes with multiple literals of the same variable.

the cube $a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$ must be ignored, and this is the motivation for the elimination of such cubes in the definition of the *cube complex associated with the tagged literal* $\overline{d}_{\{3,4,6,8,9\}}$.

We now wish to build on our understanding of ENF expressions to develop necessary and sufficient conditions for a multilevel circuit to be path-delay-fault testable in terms of its ENF representation. Because of the one-to-many relationship between paths in the circuit C and cubes in the corresponding ENF E we cannot derive testability conditions for paths directly from the results of Section 4. Instead, we will provide a series of lemmas and theorems similar in structure to those provided for two-level circuits.

Lemma 6.1 *: Let C be a multilevel single-output circuit with ENF E. Let π be a path in C and let the tagged literal associated with π be l_π. Let the cube complex of l_π be L and let D be those cubes not containing l_π. If $< v_1, v_2 >$ is a robust path-delay-fault test for π in C then for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$.*

Proof: Suppose there does not exist any such literal m both in v_1 and v_2. Then there exists at least one cube q in D such that for every literal l in v_1 and v_2 either $q_l(v_1) = 1$ or $q_l(v_2) = 1$. Thus under the appropriate input delays, the cube q can evaluate to a 1. The cube q evaluates to a 1 exactly when some path ρ in C is event sensitized. The cube q represents sensitization conditions for some set of paths including ρ and none of these conditions depend on the the sensitization of π since l_π is not in q. Since ρ and π are different paths that can be simultaneously event sensitized, we can always choose the delays in the circuit such that an event propagates along ρ to the output before an event along π. This means that the cube q can evaluate to a 1, and ρ can temporarily control the output of the C, independent of π, and therefore invalidate the test. ■

So far in our discussion of ENF expressions we have ignored cubes of the form $b_\pi \cdot \overline{b_\rho} \cdot c$. Such cubes do not cover any vertices in the ON-set of a circuit and as a result they do not give information as to how a path can be sensitized to a 1. They are of no value in constructing a robust path-delay-fault test, but it is useful to explain how they can invalidate one. Let us consider the ENF expression $a \cdot b_\pi \cdot \overline{b_\rho} + b_\sigma \cdot c_\omega + b_\sigma \cdot \overline{b_\chi}$. Consider tests for those paths that do not begin with primary input b, such as the test for path c_ω: $<< 0, 1, 0 >, < 0, 1, 1 >>$. In this case, b has the same value in both v_1 and v_2; therefore, no event travels down either path b_π or b_ρ. Another interesting case is when we test for a path such as b_σ using the sequence $<< 0, 0, 1 >, < 0, 1, 1 >>$. In this case we might be concerned about the cube $b_\sigma \cdot \overline{b_\chi}$ producing a glitch at the output; however, the path b_σ is sensitized to a 0 on $< 0, 0, 1 >$, and whatever the delays on path b_χ, the cube $b_\sigma \cdot \overline{b_\chi}$ cannot be sensitized to a 1 until the path b_σ is sensitized to a 1. Thus this cube does not invalidate the robust path-delay-fault test for b_σ. Consider the behavior of cube $a \cdot b_\pi \cdot \overline{b_\rho}$ on the vector sequence $<< 0,0,1 >, < 0,1,1 >>$. Because literal a is 0 in both vectors the cube does not cause a glitch, but the vector sequence $<< 1,0,1 >, < 1,1,1 >>$ would not be a robust path-delay-fault test for b_σ because under the appropriate delays the cube $a \cdot b_\pi \cdot \overline{b_\rho}$ could go to a 1 invalidating the test. Thus the restriction that for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$ is necessary even for cubes which do not contribute to the ON-set of the circuit.

We now give a theorem describing necessary and sufficient conditions for a multilevel single-output circuit to be robustly path-delay-fault testable.

Theorem 6.1 *: Let C be a multilevel single-output circuit with ENF E. Let π be a path in C and let the tagged literal associated with π be l_π. Let the cube complex of l_π be L and let D be those cubes not containing l_π. There exists a robust delay-fault test for π in C iff*

1. *there exists a vertex v_2 such that $L(v_2) = 1$ and v_2 is not covered by any cube in D and*
2. *there exists a vertex $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$ such that v_1 is in the OFF-set of C and*
3. *for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$.*

Proof: Necessity: Suppose $< w_1, v_2 >$ is a robust delay-fault test for π. Suppose v_2 is covered by some cube d in D; thus $d(v_2) = 1$. The cube d gives the event sensitization conditions for some path $\rho \neq \pi$ in C and therefore ρ is event sensitized by v_2. Thus, both π and ρ race to cause an output event in C. Under chosen delays, this race could invalidate the path-delay-fault test for π, and therefore by definition $< v_1, v_2 >$ is not a robust test for π. This gives a contradiction and therefore v_2 is not covered by any cube of D.

We now wish to show that while the vector w_1 need not be distance-1 (in literal l) from v_2, we can always construct a vector v_1 such that v_1 is in the OFF-set distance-1 in literal l from v_2 and $< v_1, v_2 >$ is a robust delay-fault test for π. By Lemma 6.1 it was shown to be necessary that for every cube d in D there exists some literal m in both w_1 and v_2 such that $d_m(w_1) = 0$ and $d_m(v_2) = 0$. Let the set of these literals be M. By definition all literals in M are the same in w_1 and v_2 and $D(w_1) = D(v_2) = 0$. Let $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$. The literal l is not in M because it cannot be the same in w_1 and v_2; the literal l must change values to propagate an event along π. So because $M \subseteq v_2$, $M \subseteq v_1$. Therefore $D(v_1) = 0$ and because v_1 contains $\overline{l}$, $L(v_1) = 0$. So v_1 is in the OFF-set of C. Thus the existence of these vertices is necessary for a robust delay-fault test to exist for π.

Sufficiency: To show that the existence of these vertices is sufficient for a robust delay-fault test to exist for π we simply show that these vertices, when applied as a vector pair, constitute a delay-fault test for π. After applying v_1 and waiting an arbitrarily long amount of time the circuit output settles to a 0. When v_2 is applied it first causes the input net associated with l to rise. As all the sensitization conditions for the propagation of the event (precisely the set M in Lemma 6.1) are contained in both vectors the event propagates to the output of C. Finally, the restriction that for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$ ensures that neither ON-set cubes in E, nor cubes that cover no ON-set members, can cause a glitch at the output masking the event. Thus $< v_1, v_2 >$ is a robust path-delay-fault test for π. ∎

The ENF of a circuit contains information about *all* the paths in a circuit, and hence given the ENF a two-valued logic suffices to generate a robust test for a path in an arbitrary multilevel circuit. On the other hand, path sensitization algorithms like those described in [23] use a 5-valued logic to efficiently generate robust tests directly from a multilevel circuit.

6.4 Synthesis Considerations

The restructuring of logic equations in multilevel logic optimizers like MIS [2] is performed using techniques such as *cube extraction*, *kernel extraction* and *algebraic resubstitution*. If a circuit C' is created from a circuit C by repeated application of these restructuring operations, then we call C' an *algebraic factorization* of C.

The following definitions have been taken from [14]. **Two cubes are syntactically equivalent** iff they are identical as sets of literals. **Two covers are syntactically equivalent** iff they are identical as sets of cubes. Given a combinational logic circuit, **syntactic flattening** to a two-level cover involves the substitution of intermediate gates into their fanouts without the use of Boolean reductions such as $(a + \overline{a} \equiv 1)$, $(a + a \equiv a)$, $(a \cdot a \equiv a)$ or $(a \cdot \overline{a} \equiv 0)$. (as in the computation of the ENF). A multilevel circuit M is **syntactically identical under flattening** to a two-level cover C if the cover produced after syntactically flattening M, namely C', is syntactically identical to C.

We now prove an important theorem that has applications to multilevel synthesis; we show that if a two-level circuit is completely and robustly path-delay-fault testable, then algebraic factorization retains complete path-delay-fault testability.

Theorem 6.2 *: Let C be a two-level multiple-output circuit that is completely robustly path-delay-fault testable. Let A be a multi-level circuit resulting from an algebraic factorization of C. Then A is robustly path-delay-fault testable.*

Proof: The key observation is that the ENF expression for A, E_A is precisely C, with the addition of tags on the literals. This property of algebraic factorizations is called *syntactic identity under flattening* in [14] and is described in detail there. As C is robustly path-delay-fault testable, E_A obeys the sufficiency requirements of Theorem 6.1. Thus A is robustly-path-delay fault testable. ∎

For example, let $C = \overline{a}bc+\overline{a}bd+a\overline{b}c+a\overline{b}d$. The circuit C is robustly path-delay-fault testable. Let $A = (\overline{a}b+a\overline{b})(c+d)$. Given an ordering of the gates in the circuit associated with A, the ENF of A is $E_A = \overline{a}_{\{1,3,5\}}b_{\{1,3,5\}}c_{\{4,5\}} + \overline{a}_{\{1,3,5\}}b_{\{1,3,5\}}d_{\{4,5\}} + a_{\{2,3,5\}}\overline{b}_{\{2,3,5\}}c_{\{4,5\}} + a_{\{2,3,5\}}\overline{b}_{\{2,3,5\}}d_{\{4,5\}}$. Ignoring the tags, E_A is precisely the same expression as C and therefore A has the properties required by Theorem 6.1.

In the theorem above we have shown that algebraic factorization techniques retain robust path-delay-fault testability, when two-level minimization techniques have produced an initially robustly path-delay-fault testable circuit. This means that practical application of the results of this paper is restricted to register-bounded combinational sub-circuits that can be flattened to two levels and minimized. The applicability of this result depends on the types of designs and the design style. For instance random control structures which are naturally modeled as finite-state machines can typically be flattened to two levels, while regular arithmetic and logical structures typically cannot. The direct application of these techniques to a particular design will depend on the predominance of register bounded control structures, such as in many telecommunications circuits [16], over arithmetic structures. Techniques for the synthesis of non-flattenable regular structures, such as adders and comparators, are presented in [7].

6.5 Conditions for Multiple-Output Functions

The conditions in the previous section are easily generalized to multiple-output functions. We simply require that the conditions of Theorem 6.1 be satisfied for all the paths to *each* output in the network.

7 Gate-Delay-Fault Testable Circuits with Multilevel Logic Implementations

7.1 Introduction

We wish to extend the results of Section 5 to multilevel implementations and derive necessary and sufficient conditions for a multilevel circuit to be robustly gate-delay-fault testable. The conditions for robustly detecting a gate-delay-fault in a multilevel circuit are much easier to achieve than for robustly detecting a path-delay-fault. As was pointed out in Lemma 5.2, to detect a gate-delay-fault it is sufficient to detect a path delay fault for *any* path that goes through the gate. There is one more significant weakening of the conditions necessary for detection. For a path to be robustly tested it must be singly event sensitized; however, to robustly detect a gate-delay-fault it is not necessary that any single path through a gate be singly event sensitizable. This fault will be detected even if a number of events simultaneously propagate through the gate, because if a gate-delay-fault exists at the gate, each of these events will be delayed. We again use the ENF representation in our analysis.

The **gate-cube-complex** of a gate g, call it L_g, in a circuit C with ENF E is defined as the set of all cubes containing tagged literals associated with a path through g. More formally, $L_g = \{q \in E \mid (l_\pi \in q) \wedge (g \in \pi)\}$.

In the definition of cube-complex associated with a tagged literal, given in Section 6.2, cubes such as $a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$ were disqualified from appearing in the complex. This was required by the fact that when $< v_1, v_2 >$ is a robust test for a path-delay-fault on a $0 \rightarrow 1$ transition, $< v_2,\ v_1 >$ is also a robust test for the path on a $1 \rightarrow 0$ transition. In the case of cubes of the form above, if we are checking for a $0 \rightarrow 1$ transition along path $d_{\{3,4,6,8,9\}}$, then we have two $0 \rightarrow 1$ transitions at the inputs of the AND gate 8, and the slower of these is passed through. However, if we are dealing with the reverse $1 \rightarrow 0$ transition, then the faster of these transitions is passed through and the delay-fault test is invalidated.

However, in the case of a gate-delay-fault, if we have converging transitions at the inputs to the faulty gate, a test $< v_1, v_2 >$ and a fault corresponding to the $0 \rightarrow 1$ primary output transition, then the transitions at the gate are all delayed. Similarly when we use a test $< w_2,\ w_1 >$, we are dealing with the fault corresponding to the $1 \rightarrow 0$ primary output transition, and again we ensure that the transitions at the gate are all delayed.

7.2 Conditions for Single-Output Functions

The requirement that robust gate-delay-fault tests do not allow hazards, races and glitches gives the following condition.

Lemma 7.1 *: Let C be a multilevel single-output circuit with ENF E. Let g be a gate in C and let the gate-cube-complex of g be L_g. Let $D = E - L_g$. If $< v_1, v_2 >$ is a robust gate-delay-fault test for g in C then for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$.*

Proof: This can be proved by a similar argument to the one used in the proof of Lemma 6.1. ■

The introduction of a concept developed for the analysis of hazards in asynchronous circuits will aid in proving the following theorem. Given a vertex v_1 and a vertex v_2 we define the **transition cube** for this vector pair, call it x, as follows: Let S be the set of variables that change values from v_1 to v_2. Let $x = \{l | \ (l \in v_2) \wedge (l, \bar{l} \ \notin \ S)\}$, *i.e.* let x be the cube resulting from the elimination of all literals in S from v_2. The cube x covers all the vertices in the N-cube that may be reached under arbitrary delays as the inputs change from v_1 to v_2.

Theorem 7.1 *: Let C be a multilevel single-output combinational circuit with ENF E and let g be a gate in C. Let L_g be the gate-cube-complex associated with g and let $D = E - L_g$. There exists a robust gate-delay-fault test for gate g in C on a $0 \rightarrow 1$ primary output transition iff*

1. *there exists a cube q such that $q \in L_g$ and*
2. *there exists a vertex v_2 such that $q(v_2) = 1$ and v_2 is not covered by any cube in D and*
3. *there exists a literal $l_\pi \in q$ such that $g \in \pi$ and $v_1 \ = \ v_2 - l \cup \bar{l}$ is in the OFF-set of C and*
4. *for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$.*

Proof: Necessity: First note that the condition that v_2 must be a relatively essential vertex of the gate-cube-complex of g can be argued as in previous theorems.

Suppose $< w_1, \ w_2 >$ is a robust gate-delay-fault test for g on a $0 \rightarrow 1$ output transition in C. The Hamming distance of this vector pair may be great but we will proceed to construct a distance-1 vector pair as in the theorem. Note that in order to avoid glitches at the output, for every possible path in the Boolean N-space from w_1 to w_2, we require that the vertices encountered in the path change from being OFF-set vertices to ON-set vertices exactly once. Let x be the transition cube for the vector pair $< v_1,\ v_2 >$. As noted above, under the appropriate delays on inputs, any vertex covered by x could be reached by applying the pair $< w_1,\ w_2 >$. This implies that the intersection of x with each cube d in $D = E - L_g$ is empty. If not, then under appropriate delays some vertex covered by another cube in D could be reached and a glitch would occur that invalidates the gate-delay-test for g. Thus, the cube x covers exactly members of the ON-set covered only by cubes in L_g and members of the OFF-set.

We wish to find two adjacent vertices in the subspace determined by x, such that one is in the ON-set of C and the other is in the OFF-set. The circuit values appearing on the inputs of C as the vector pair $< w_1,\ w_2 >$ is applied may be visualized as a sequence of vertices forming a path in the Boolean N-space from vertex w_1 and vertex w_2. Furthermore, this path is restricted to be in the subspace determined by the cube x. As vertex w_1 is in the OFF-set and vertex w_2 is in the ON-set, there exist adjacent vertices v_1 and v_2 at some point on the path such that v_1 is in the OFF-set and v_2 is in the ON-set. This may easily be proved by induction on the length of the path from w_1 to w_2.

The argument above shows that it is necessary that there exist a vector pair $< v_1,\ v_2 >$ which is distance-1 in some literal l and that v_2 is a relatively essential vertex of L_g. We now wish to show that for each cube q such that $q(v_2) = 1$ there must exist one literal l in q that is tagged with a path π that passes through g. In other words for each cube $q \in L_g$ such that $q(v_2) = 1$, there is a literal l_π in q such that l_π is tagged with a path through g and it is the transition of l_π from 0 to 1 on $< v_1,\ v_2 >$ that makes the cube q make the transition from 0 to 1.

Suppose that there exists a cube q with no such l_π. We know that the literal l must be in q otherwise q would not make the transition from 0 to 1, but by supposition the literal $l \in q$ is not tagged with π such that $g \in \pi$ but instead with some σ such that g is not in σ. In this case we could apply the vector sequence $< w_1,\ w_2 >$ and under appropriate delays we could travel from w_1 to v_1 and from v_1 (through a primary input change in the variable associated with literal l) to v_2. Now, $q(v_1) = 0$, $q(v_2) = 1$ and literal l_σ causes the transition. But g is not in σ so when the vector pair $< w_1,\ w_2 >$ is applied, under the appropriate input delays the path that is sensitized to a 1 does not propagate an event through gate g. But this would contradict the assertion that $< w_1,\ w_2 >$ is a robust gate-delay-fault test for g. So there must exist such an $l_\pi \in q$ as in the theorem.

That for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$ was shown in Lemma 7.1.

Sufficiency: After applying v_1 the output of C settles to a 0. When the vector v_2 is applied the only paths that are sensitized to a 1 are those that pass through g. The restriction that for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$ ensures that neither ON-set cubes in E, nor cubes that cover no ON-set members can cause a glitch at the output masking the event. Thus if all transitions at g are delayed then the transition to a 1 at the output is delayed, and the pair $< v_1,\ v_2 >$ is a robust gate-delay-fault test for g. ■

For example, consider a gate-delay-fault test for gate 8 in Figure 1 consisting of $w_1 =< 1,1,0,1,0 >$ and $w_2 =< 1,0,0,0,0 >$. Then $S = \{b, d\}$ and $x = a \cdot \overline{c} \cdot \overline{e}$. Under the appropriate delays any vertex in the subspace covered by $a \cdot \overline{c} \cdot \overline{e}$ is reachable. This is a distance-2 test but we wish to find a distance-1 test as in the theorem. We can choose any path from w_1 to w_2 that we like; one route is $< 1,1,0,1,0 >,< 1,1,0,0,0 >,< 1,0,0,0,0 >$. The last vertex in the OFF-set in this sequence is $< 1,1,0,1,0 >$. The next vertex in the ON-set is $< 1,1,0,0,0 >$. Thus $v_1 =< 1,1,0,1,0 >$ and $v_2 =< 1,1,0,0,0 >$ is a distance-1 test with the variable d changing from v_1 to v_2. One cube q (in fact the only one) such that $q(v_2) = 1$ is $a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$. Both the tagged literals $\overline{d}_{\{3,4,6,8,9\}}$ and $\overline{d}_{\{7,8,9\}}$ have paths through gate 8 and they are both responsible for cube $a_{\{1,4,6,8,9\}} \cdot \overline{d}_{\{3,4,6,8,9\}} \cdot \overline{d}_{\{7,8,9\}}$ making the transition from $0 \rightarrow 1$. It is interesting to note that $<< 1,1,0,1,0 >, < 1,1,0,0,0 >>$ is not a robust path-delay-fault test because paths $d_{\{3,4,6,8,9\}}$ and $d_{\{d,7,8,9\}}$ are both sensitized to a 1, but as they both go through gate 8 they will both be delayed if gate 8 has a delay fault.

As mentioned previously, to avoid glitches at the output, we require that any robust gate-delay-fault test pair $< w_1,\ w_2 >$ be such that for every possible path in the Boolean N-space from w_1 to w_2, the vertices encountered in the path change from being OFF-set vertices to ON-set vertices exactly once. It can be argued that glitches that propagate through the gate under test do not invalidate a test even if they appear the output. Relaxing the requirement above regarding the relationship between $< w_1,\ w_2 >$, and analyzing the nature of the delay-fault tests under this relaxed relationship bears further investigation. An example is the test pair $w_1 =< 1,1,0,1,0 >$ and $w_2 =< 1,0,0,0,0 >$, for the circuit of Figure 1.

The conditions for robust gate-delay-fault testability are not exactly symmetric and the consideration of the $1 \rightarrow 0$ transition requires an additional definition. The **single-literal gate-cube complex** associated with a gate g and a *specific* literal l is defined to be the set of all cubes in L_g

such that at least one occurrence of l exists in each cube, and such that *all* occurrences of literal l contain gate g in their tag. Thus, $L_g(l) = \{q \in L_g \mid (l \in q) \wedge (g \in \pi \; \forall \; l_\pi \in q)\}$. The set of cubes defined by the single-literal gate-cube-complex is used in the following theorem to give the necessary and sufficient conditions for a multilevel single-output function to be robustly gate-delay-fault testable on a $1 \rightarrow 0$ primary output transition.

Theorem 7.2 *: Let C be a multilevel single-output combinational circuit with ENF E. Let g be a gate in C. There exists a robust gate-delay-fault test for gate g in C on a $1 \rightarrow 0$ primary output transition iff there exists some literal l such that*

1. *there exists a cube q such that $q \in L_g(l)$, where $L_g(l)$ is the single-literal gate-cube-complex associated with l and g and*
2. *there exists a vertex v_2 such that $q(v_2) = 1$ and v_2 is not covered by any cube in $D = E - L_g(l)$ and*
3. *there exists a vertex v_1 such that $v_1 = v_2 - \{l\} \cup \{\bar{l}\}$ is in the OFF-set of C and*
4. *for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$.*

Proof: Necessity: Suppose $< w_2,\; w_1 >$ is a robust gate-delay-fault test for g on a $1 \rightarrow 0$ output transition. The proof is very similar to that in Theorem 7.1 but having constructed the vector pair $< v_2,\; v_1 >$ as in the proof of that theorem, we need to show why each q that covers v_2 here needs to be a member of $L_g(l)$ and not simply L_g. Let q be a cube that covers v_2. If we apply the vector pair $< v_1,\; v_2 >$ then the cube q cannot make the transition from 0 to 1 until literal l_π makes the transition from 0 to 1. However, if we apply the vector pair $< v_2,\; v_1 >$ then the cube q can make the transition from 1 to 0 as soon as any literal $l_\sigma \in q$ makes the transition from 1 to 0. Thus, for each such q *every* tagged literal l_ρ associated with l in q must be tagged with a path ρ such that ρ passes through gate g. Therefore $q \in L_g(l)$. Otherwise the proof is as in Theorem 7.2.

Sufficiency: After applying v_2, the output of C settles to a 1. When the vector v_1 is applied the only paths that have a $1 \rightarrow 0$ event are those that pass through g. The restriction that for every cube d in D there exists some literal m in both v_1 and v_2 such that $d_m(v_1) = 0$ and $d_m(v_2) = 0$ ensures that neither ON-set cubes in E, nor cubes that cover no ON-set members can cause a glitch at the output masking the event. Thus if all transitions at g are delayed then the transition to a 1 at the output is delayed, and the pair $< v_2,\; v_1 >$ is a robust gate-delay-fault test for g. ∎

7.3 Synthesis Considerations

We wish to construct a procedure which synthesizes a robust gate-delay-fault testable multilevel circuit from a robustly gate-delay-fault testable two-level circuit.

Because robust path-delay-fault testability was such a strong property, it was possible to retain it using unconstrained algebraic factorization. However, a constrained algebraic factorization procedure is required to retain gate-delay-fault testability in a multilevel circuit. This is analogous to the situation with single-stuck-at-fault testability and multi-stuck-at-fault testability. As is shown in [14], if a combinational circuit is multifault testable then this property is retained through unconstrained algebraic factorization. But, if the property of the initial circuit is weaker, such as the sequential single-stuck-at-fault testability, then a constrained algebraic factorization procedure is required [5] [8].

Consider the two-level circuit shown in Figure 2(a). All four gates are robustly testable for gate delay faults. However, when the algebraic factor ab is extracted from the cube abc, the circuit is no longer robustly gate-delay-fault testable. The algebraically factored circuit is shown in Figure 2(b). In this circuit, gate 7 is untestable. In the original circuit, the only path through gate 3 which was testable was that originating at literal c. Thus there is no way to test the new gate corresponding to the factor ab when it is extracted from the cube abc in the original cover. A constrained factorization would be required in order to preserve testability of the gate-delay-faults in the circuit.

The essence of an algebraic factorization procedure that retains robust gate-delay-fault testability of the initial multilevel circuit is to ensure that in each cube of each factor (cube or kernel) there is at least one literal with the properties of literal l in Theorem 6.1. Of course if all literals in the initial two-level circuit have this property, as in the case of a robustly path-delay-fault testable two-level circuit, then no constraints are required. The theorem below gives a succinct sufficiency condition for the retainment of gate-delay-fault testability, beginning from a two-level circuit where a path through each gate is robustly testable.

Theorem 7.3 : *Given a two-level network C, where at least one path through each* AND *gate is robustly testable, let E_C be the ENF of circuit C. Let A be a multilevel circuit resulting from an algebraic factorization of C, with ENF E_A. If the tagged literals in E_C, which satisfy the conditions of Theorem 6.1 correspond to a set of tagged literals/paths in E_A/A that pass through all the gates in A, then A is completely robustly gate-delay-fault testable.*

Proof: By Theorem 6.2, there is a one-to-one correspondence between the literals of E_A and E_C.

Thus, if a tagged literal in E_C satisfies the conditions of Theorem 6.1, then the corresponding tagged literal in E_A does also. Since the set of paths in A, that correspond to tagged literals in E_A satisfying the conditions of Theorem 6.1, pass through all the gates in A, robustly testing these paths will robustly test all gate-delay-faults in A. ■

It is possible, as illustrated in Figure 2, that the robustly testable paths in the two-level circuit do not produce corresponding paths which pass through all the gates after algebraic factorization. Constraints have to be imposed on cube and kernel extraction in order to ensure that this condition is satisfied.

7.4 Conditions for Multiple-Output Functions

For multiple-output functions, we simply require that any gate in the multilevel circuit be robustly testable at *some* output, because in the common model of gate-delay-faults it is assumed that the gate-delay-fault is sufficiently catastrophic to produce a delay at every output. Using this observation Theorems 7.1 and 7.2 may easily be generalized.

8 Experimental Results

We present experimental results using an ENF-based program for analyzing the delay-fault testability of multilevel circuits, and preliminary experimental results obtained via algebraic factorization to retain robust path-delay-fault testability.

We chose several examples from the MCNC Benchmark set, whose statistics and path-delay-fault testabilities under a two-level implementation are summarized in Table 1. The testabilities for single-output minimized and multiple-output minimized two-level representations were determined using an ENF-based analysis program. While the single-output minimized covers are larger, they have significantly higher testabilities.

Multilevel logic circuits are of much greater utility. In Table 2, we compare literal counts and path-delay-fault testabilities obtained using algebraic factorization and unconstrained multilevel logic optimization on the different examples. Unconstrained multilevel optimization implies that the initial two-level network was multiple-output minimized and Boolean factorization was used during logic optimization. Algebraic factorization was used on the circuits produced by the single-output minimization (Table 2). A modified version of the program MIS [2] was used in both cases. As can be seen, the literal counts are comparable, but significantly higher testabilities can be achieved via algebraic factorization. In several cases, the use of Boolean factorization in a heuristic logic optimization strategy adversely affects literal counts. Of course, if one were unconcerned with testability issues, one would pick the best of the results produced by algebraic or Boolean factorization.

While the ENF-based path-delay-fault testability analysis program has a worst-case exponential time and space complexity, it can be used to analyze circuits of reasonable size, due to the fact that the ENF of an algebraically factored two-level circuit is the same size as the original two-level circuit.

More general and efficient synthesis procedures, based on the theory presented in this paper, that can be used for significantly larger circuits have been developed [7, 9]. These procedures have been used to synthesize non-flattenable circuits, with over 5000 gates, to be completely robustly path-delay-fault testable.

9 Conclusions

In this paper we gave a comprehensive theoretical framework for the analysis and synthesis of delay-fault testable combinational logic circuits. We have provided necessary and sufficient conditions for robust delay-fault testability for gate-delay-faults and path-delay-faults in two-level circuits. We described these conditions using terminology common to two-level minimization and showed their relationship to properties produced by two-level minimizers. Using ENF analysis we were then able to develop necessary and sufficient conditions for delay-fault testability in multilevel networks.

Where convenient we have dealt not only with the theoretical properties required for delay-fault testable circuits but also with the issues involved in synthesizing such delay-fault testable circuits. In particular we proved that algebraic factorization preserves the robust path-delay-fault testability of a two-level cover. While we feel that this body of theoretical work is in itself interesting for its contribution to the area of delay testing and for the insights it gives into properties of combinational logic circuits, we are especially excited by the potential for practical application of these results, and we have implemented procedures for synthesizing delay-fault testable circuits based on the theory developed in this paper [9].

10 Acknowledgements

Thanks to Robert Brayton, Sharad Malik, Alex Saldanha, Sudhakar Reddy and Alberto Sangiovanni-Vincentelli for interesting discussions on logic synthesis and testability. Thanks especially to Herve Touati and Yosinori Watanabe for disproving our earlier conjecture regarding sufficiency conditions for gate-delay-fault testability in two-level circuits, and Michael Bryan for correcting the necessary and sufficient conditions for multilevel gate-delay-fault testability. Additional thanks are due to Sudhakar Reddy for pointing us to the work described in [4]. This work was supported in part by the Defense Advanced Research Projects Agency under contract N00014-87-K-0825.

References

[1] D. B. Armstrong. On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets. *IEEE Transactions on Computers*, EC-15(2):66–73, February 1966.

[2] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. MIS: A Multiple-Level Logic Optimization System. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, CAD-6(6):1062–1081, November 1987.

[3] M. A. Breuer and A. D. Friedman. *Diagnosis and Reliable Design of Digital Systems*. Computer Science Press, 1976.

[4] G. L. Craig. Stuck-Open Fault Testability in Combinational Networks. In *Ph.D thesis, University of Wisconsin at Madison*, 1987.

[5] S. Devadas and K. Keutzer. Boolean Minimization and Algebraic Factorization Procedures for Fully Testable Sequential Machines. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 208–211, November 1989.

[6] S. Devadas and K. Keutzer. Necessary and Sufficient Conditions for Robust Delay-Fault Testability of Logic Circuits. In *Sixth MIT Conference on Advanced Research on VLSI*, pages 221–238, April 1990.

[7] S. Devadas and K. Keutzer. Synthesis and Optimization Procedures for Robustly Delay-Fault Testable Logic Circuits. In *Proceedings of the 27^{th} Design Automation Conference*, pages 221–227, June 1990.

[8] S. Devadas and K. Keutzer. A Unified Approach to the Synthesis of Fully Testable Sequential Machines. In *IEEE Transactions on Computer-Aided Design*, pages 39–50, January 1991.

[9] S. Devadas and K. Keutzer. Synthesis of Robust Delay-fault Testable Circuits: Practice. 11:277–300, March 1992.

[10] S. Devadas, K. Keutzer, and A. R. Newton. New Trends in Testing and Verification. In *Tutorial Notes: Int'l Conference on Computer-Aided Design*, November 1989.

[11] H. Fujiwara. *Logic Testing and Design for Testability.* MIT Press, Cambridge MA, 1985.

[12] C. T. Glover and M. R. Mercer. A Method of Delay Fault Test Generation. In *Proceedings of the* 25^{th} *Design Automation Conference*, pages 90–95, June 1988.

[13] C. T. Glover and M. R. Mercer. A Deterministic Approach to Adjacency Testing for Delay Faults. In *Proceedings of the* 26^{th} *Design Automation Conference*, pages 351–356, June 1989.

[14] G. D. Hachtel, R. M. Jacoby, K. Keutzer, and C. R. Morrison. On Properties of Algebraic Transformations and the Synthesis of Multifault-Irredundant Circuits. In *IEEE Transactions on Computer-Aided Design*, pages 313–321, March 1992.

[15] T. Hayashi, K. Hatayama, K. Sato, and T. Natabe. A Delay Test Generator for LSI Logic. In *Proceedings of the* 14^{th} *Fault Tolerant Computing Symposium*, pages 146–149, June 1984.

[16] K. Keutzer. Three Competing Design Methodologies for ASICS: Architectural Synthesis, Logic Synthesis and Module Generation. In *Proceedings of the* 26^{th} *Design Automation Conference*, pages 308–313. ACM/IEEE, June 1989.

[17] S. Kundu. Design of Multioutput CMOS Combinational Circuits for Robust Testability. *IEEE Transactions on Computer-Aided Design*, 8(11):1222–1226, November 1989.

[18] S. Kundu and S. M. Reddy. On the Design of Robust Testable CMOS Combinational Logic Circuits. In *Proceedings of the Fault Tolerant Computing Symposium*, pages 220–225, 1988.

[19] S. Kundu, S. M. Reddy, and N. K. Jha. On the Design of Robust Multiple Fault Testable CMOS Combinational Logic Circuits. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 240–243, November 1988.

[20] Y. Levendel and P. R. Menon. Transition Faults in Combinational Circuits: Input Transition Test Generation and Fault Simulation. In *Proceedings of the* 16^{th} *Fault Tolerant Computing Symposium*, pages 278–283, June 1985.

[21] W-N Li, S. M. Reddy, and S. Sahni. On Path Selection in Combinational Logic Circuits. In *IEEE Transactions on Computer-Aided Design*, pages 56–63, January 1989.

[22] C. C. Liaw, S. Y. Su, and Y. K. Malaiya. Test Generation for Delay Faults Using the Stuck-at Fault Test Set. In *Proceedings of the Int'l Test Conference*, pages 167–175, November 1980.

[23] C. J. Lin and S. M. Reddy. On Delay Fault Testing in Logic Circuits. In *IEEE Transactions on Computer-Aided Design*, pages 694–703, September 1987.

[24] Y. K. Malaiya and R. Narayanswamy. Testing for Timing Failures in Synchronous Sequential Integrated Circuits. In *Proceedings of the Int'l Test Conference*, pages 560–571, October 1983.

[25] E. J. McCluskey. Transients in Combinational Logic Circuits. In R. H. Willson and W. C. Mann, editors, *Redundancy Techniques for Computing Systems*, pages 9–46. Spartan Books, 1962.

[26] A. Pramanick and S. Reddy. On The Design of Path Delay Fault Testable Combinational Circuits. In *Proceedings of the* 20^{th} *Fault Tolerant Computing Symposium*, pages 374–381, June 1990.

[27] A. Pramanick, S. Reddy, and S. Sengupta. Synthesis of Combinational Logic Circuits for Path Delay Fault Testability. In *Int'l Symposium on Circuits and Systems*, pages 3105–3108, May 1990.

[28] J. Rajski and H. Cox. A Method of Test Generation and Fault Diagnosis in Very Large Circuits. In *Proceedings of the Int'l Test Conference*, pages 932–943, September 1987.

[29] K. Roy, K. De, J. A. Abraham, and S. Lusky. Synthesis of Delay Fault Testable Combinational Logic. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 418–421, November 1989.

[30] J. Savir and W. H. Anney. Random Pattern Testability of Delay Faults. In *Proceedings of the International Test Conference*, pages 263–273, October 1986.

[31] G. L. Smith. A Model for Delay Faults Based on Paths. In *Proceedings of the Int'l Test Conference*, pages 342–349, September 1985.

[32] H. Touati and Y. Watanabe. personal communication, 1989.

[33] S. H. Unger. *Asynchronous Sequential Switching Circuits*. Wiley, New York, 1969.

[34] J. A. Waicukauski, E. Lindbloom, B. Rosen, and V. Iyengar. Transition Fault Simulation by Parallel Pattern Single Fault Propagation. In *Proceedings of the Int'l Test Conference*, pages 542–549, September 1986.

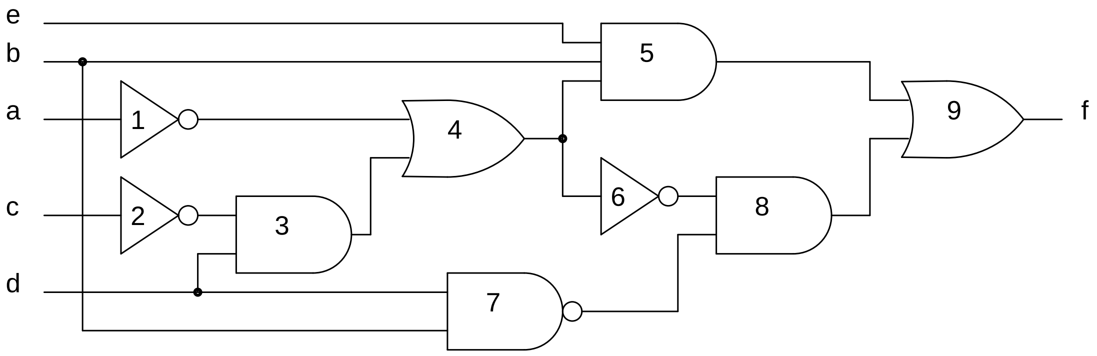
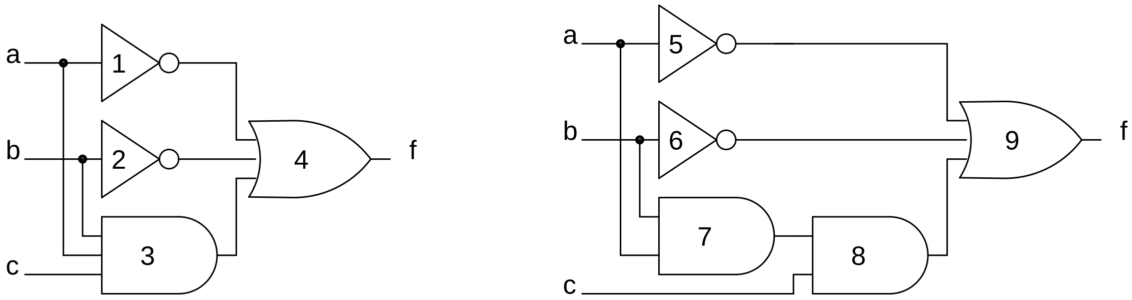


Figure 1: Example multilevel circuit to illustrate the ENF



(a) Two-level circuit

(b) Factored circuit

Figure 2: Algebraic factorization lowering gate-delay-fault testability

example	in	out	multiple-output		single-output	
			pro	path	pro	path
Z5xp1	7	10	65	70.9	74	98.3
Z9sym	9	1	86	95.3	86	95.3
add4	9	5	135	100.0	135	100.0
add6	12	7	355	100.0	355	100.0
addm4	9	8	200	88.0	213	95.9
adr4	8	5	75	100.0	75	100.0
al2	16	47	66	91.9	89	99.5
bc0	26	11	179	72.5	487	92.3
bcd.div3	4	4	9	88.6	11	100.0
dist	8	5	123	83.3	152	92.3
f51m	8	8	77	94.7	76	96.6
l8err	8	8	52	86.1	58	92.4
life	9	1	84	79.2	84	79.2
log8mod	8	5	38	84.4	52	92.8
m181	15	9	42	82.9	53	95.2
m2	8	16	47	45.9	106	95.2
m3	8	16	62	41.9	133	94.6
m4	8	16	105	40.5	211	93.3
max512	9	6	133	73.5	167	91.1
mlp4	8	8	128	89.0	144	97.1
par8	8	1	128	100.0	128	100.0
pla4	31	90	147	84.8	695	98.6
radd	8	5	75	100.0	75	100.0
rckl	32	7	32	61.1	97	100.0
rd53	5	3	31	83.8	31	100.0
rd73	7	3	127	88.4	141	94.8
root	8	5	57	71.8	71	90.2
sqr6	6	12	49	81.0	58	92.8
sym10	10	1	210	70.6	210	70.6
t3	12	8	33	99.1	33	99.1
x1dn	27	6	110	100.0	110	100.0
z4	7	4	59	100.0	59	100.0

Table 1: Testability and area comparisons of two-level implementations

example	algebraic		Boolean	
	lit	path	lit	path
Z5xp1	152	98.9	154	49.7
Z9sym	238	95.9	213	97.7
add4	56	100.0	76	74.3
add6	96	100.0	97	40.3
addm4	476	97.3	501	64.9
adr4	48	100.0	59	56.5
al2	155	99.6	155	78.5
bc0	1132	94.8	769	74.0
bcd.div3	24	100.0	24	65.1
dist	413	94.5	345	33.5
f51m	167	97.6	120	24.1
l8err	170	95.3	135	80.9
life	80	89.1	73	87.1
log8mod	152	95.6	120	23.1
m181	97	100.0	92	76.7
m2	253	96.4	290	43.8
m3	339	95.7	383	40.7
m4	514	94.7	604	41.7
max512	468	93.3	467	72.4
mlp4	353	98.2	384	91.2
par8	28	100.0	28	100.0
pla4	1444	99.0	1079	78.4
radd	48	100.0	57	44.0
rckl	195	100.0	197	61.4
rd53	59	100.0	36	33.2
rd73	137	93.4	65	79.5
root	169	94.1	158	78.2
sqr6	148	95.3	150	43.7
sym10	317	77.6	348	71.1
t3	98	100.0	86	48.1
x1dn	101	100.0	89	100.0
z4	42	100.0	51	83.4

Table 2: Testability and literal count comparisons of multilevel implementations