TECHNICAL REPORT NO. 278

Probabilistic Analysis of Algorithms for Stuck-at Test Generation in PLAs.

by

John Franco and Kurt Keutzer

May 1989

COMPUTER SCIENCE DEPARTMENT

INDIANA UNIVERSITY

Bloomington, Indiana 47405–4101

Probabilistic Analysis of Algorithms for Stuck-at Test Generation in PLAs

John Franco
Department of Computer Science
Indiana University
Bloomington, Indiana 47405

Kurt Keutzer
ATT Bell Laboratories
Murray Hill, NJ

May 12, 1989

Abstract

A collection of algorithms for generating test vectors for PLAs is presented and analyzed. It is shown that, in some sense, complete sets of test vectors for almost all such circuits which are irredundant, primal, and non-tautological can be generated in polynomial time.

1 Introduction

It has been known for some time that logic optimization can produce circuits that are completely testable for all stuck-at faults. The relationship between testability and Boolean minimization for two-level combinational circuits dates back to the Quine-McCluskey algorithm [8]. The notions of primality and irredundancy were generalized for multi-level circuits in [1]. Recent work in synthesis for testability has been able to ensure complete multiple-fault testability for multi-level combinational logic circuits [5]. All of these results only show in varying ways that with unlimited computational resources test vectors could be generated for all stuck-at faults in a circuit; however, in practice the testability of a circuit is that which can be found with relatively modest computational resources. Are these two notions of testablity related? In other words, does *fully* testable imply *easily* testable? In general, knowledge of the existence of a solution to an NP-complete problem does not improve the chances of finding a solution [6]; however, it seems reasonable to expect that full testability should improve the expected time for generating tests.

In this paper we begin to formally evaluate the impact of complete testability of a circuit on the ease of finding tests. In particular we relate the primality and irredundancy of two-level circuits to the ease of finding test vectors for stuck-at faults in two-level circuits. Two-level circuits continue to be of interest because they are naturally implemented in programmable-logic arrays (PLA's). The AND/OR structure of a two-level circuit corresponds naturally to sum-of-products structure of disjunctive-normal-form (DNF). In such circuits all single and multiple stuck-at faults are testable if all single stuck-at-faults on the inputs and outputs of the AND gates are testable [7]. Finding a test vector for the output of AND gate is equivalent to finding a truth assignment that falsifies all clauses in the corresponding DNF expression, except for the clause that corresponds to the node under test, and sets that clause to *false* if a stuck-at-1 test is being generated or *true* if a stuck-at-0 test is being generated.

What we intend to show is that, in some sense, a large class of prime and irredundant circuit instances are provably testable in polynomial time. Thus, for a class of circuit instances complete testability implies easy testability with high probability. We do this by showing that test vectors can be obtained in polynomial time for almost all prime and irredundant instances generated by a standard probabilistic model.

It should be noted that there are some remaining problems in the direct application of this work to testing PLA's. The first is that the testing of PLA's in current technologies typically requires testing for shorts and cross-point faults, as well as stuck-at faults; furthermore, to address these additional faults specialized design for testability methods such as those described in [4] may be used. Secondly, the optimization approach used in some of the current heuristic two-level logic optimizers, such as [2], could, theoretically, generate stuck-at fault test vectors as a side effect, although in current practice test vectors are generated by auxillary tools. Despite these practical limitations we feel that this work is well motivated by the need to improve our understanding of the relationship between complete testability and ease of testability, and that it is natural to begin a study of this problem in the better understood domain of two-level circuits.

2 Preliminaries

A Boolean variable can take two values, *true* and *false*. A literal is either a negated or unnegated Boolean variable. If the literal is negated its value is opposite the value of the corresponding variable. Otherwise the literal has the same value as the corresponding variable. If v is a Boolean variable its negated literal is $\bar{v}$. The literals v and $\bar{v}$ are said to be complementary.

A DNF Boolean expression is a disjunction of conjunctions of literals. We represent DNF expressions as a collection of sets of literals. Each set of literals is called a clause. A clause is satisfied and has value *true* if one of its literals has value *true*. A clause is falsified and has value *false* if all its literals have value *false*.

Definition:

The *functionality* of an expression I containing r Boolean variables is a mapping $F_I : B_1 \times$

$B_2 ... B_r \rightarrow \{true, false\}$ where B_i, $1 \leq i \leq r$, is the Boolean value of variable v_i such that $F_I(t) = true$ if and only if at least one clause in I is satisfied by truth assignment t.

Definition:

A clause c in expression I is *irredundant* if removal of C from I changes the functionality of I.

Definition:

A literal l in clause c of expression I is *primal* if removal of l from c changes the functionality of I.

It is not the case that primality implies irredundancy. For example, each literal of each clause of $(a, \bar{b}), (\bar{b}, c), (\bar{a}, c)$ is primal but functionality is not changed if the clause $(\bar{b}, c)$ is dropped. It is also not the case that irredundancy implies primality. For example, each of the clauses of $(a, \bar{b}), (a, b)$ is irredundant but removal of neither literal b or $\bar{b}$ changes functionality.

We talk about testability of a DNF expression with the understanding that there is a direct mapping to the testability of the PLA based on that expression.

Definition:

A DNF expression is *completely testable* if, a) for every clause there exist two truth assignments which satisfy and falsify the clause and falsify all other clauses; b) for every literal l in a clause c there exist two assignments which cause all other literals in c to have value *true*, cause l to have value *true* and *false*, and falsify all other clauses.

A clause for which both truth assignments do not exist is said to be not testable. A literal for which both truth assignments do not exist is said to be not testable.

DNF expressions that are irredundant and primal are completely testable. For suppose there is a clause e that is not testable. Then either there is one value that e may take such that all assignments inducing this value satisfy at least one other clause or there is a value that e may never take. In the second case e is redundant so we only consider the first case. Setting e to the opposite value does not change functionality hence e is redundant. A similar argument holds for testing literals.

3 The Probabilistic Model

The probabilistic model we use generates DNF Boolean expressions consisting of n independent clauses, each of which is constructed as follows. Let $V = v_1, v_2, ..., v_r$ be a set of r Boolean variables. For all $1 \leq i \leq r$, with probability p the clause contains v_i, with probability p the clause contains

$\bar{v}_i$, and with probability $1-2p$ the clause contains neither. We call this model $M(n,r,p)$. An expression generated by the model will be called a random expression.

In a random expression it is possible for a clause to contain no literals. Such a clause will have no effect on the functionality of the expression. If p is set too low then all clauses contain no literals with probability tending to 1.

Lemma 1 *If* $\lim_{n,r\to\infty} pnr = 0$ *then a random expression contains only empty clauses with probability tending to 1.*

Proof:

The probability that a clause is empty is $(1-2p)^r$. The probability that all clauses are empty is $(1-2p)^{rn} = 1-2prn+O((prn)^2)$. The lemma follows. □

For a certain range of p there is at least one unit clause in a random expression.

Lemma 2 *If* $\lim_{n,r\to\infty} prn = \infty$ *and* $p < (1-\epsilon)\ln(n)/(2r)$ *for any* $\epsilon > 0$ *then the probability that there is a unit clause in a random expression tends to 1.*

Proof:

The probability that a particular clause is a unit clause is $\binom{r}{1}(2p)(1-2p)^{r-1}$. Hence the probability that a random expression contains no unit clauses is $f(p,r,n) = (1-2pr(1-2p)^{r-1})^n$. The expression $2pr(1-2p)^{r-1}$ has a maximum at $pr = 1$. We consider the endpoints $prn = k$ where k is large and $p = (1-\epsilon)\ln(n)/(2r)$ of the range specified by the hypothesis. If $prn = k$ then the probability that the expression contains no unit clauses is less than $e^{-2prn(1-2p)^r} < e^{-2k(1-k/n+O(1/n^2))}$. This tends to 0 as k increases. If $p = (1-\epsilon)\ln(n)/(2r)$ then the probability is less than $e^{-2prn(1-2p)^r} = e^{-(1-\epsilon)n\ln(n)(1-2p)^r} < n^{-(1-\epsilon)n^{1-(1-\epsilon)}} = n^{-(1-\epsilon)n^{\epsilon}}$. This tends to 0 as $n \to \infty$. Since $f(p,r,n) \to 0$ at the interval endpoints and $2pr(1-2p)^{r-1}$ increases from the low endoint to a maximum at $pr = 1$ and then decreases until the upper endpoint the lemma is proved. □

Over this range of p there are redundant clauses with high probability.

Lemma 3 *The probability that the literal of a unit clause is contained in another clause of a random expression tends to 1 if* $\lim_{n,r\to\infty} pn = \infty$ *and* $p < (1-\epsilon)\ln(n)/(2r)$ *for any* $\epsilon > 0$.

Proof:

From Lemma 2 the probability that there is a unit clause tends to 1. The probability that a literal which is the unit clause is not in any other clause is $(1-p)^{n-1}$. In the limit this is e^{-pn}. The lemma follows. □

From Lemmas 2 and 3, if $pn \to \infty$, and $p < (1-\epsilon)\ln(n)/(2r)$ then almost all random expressions have redundant clauses.

A tautology is not a reasonable PLA function. We restrict our attention to the range of p for which almost all random expressions are not tautologies.

Lemma 4 *The probability that a random truth assignment falsifies all clauses of an expression generated by $M(n,r,p)$ is greater than $(1-(1-p)^r)^n$.*

Proof:

The probability that a random truth assignment does not satisfy a clause is the probability that at least one of the literals in the clause is falsified or the clause contains no literals. This is greater than the probability that at least one of the literals in the clause is falsified which is one minus the probability that each of the r variables in V is either not in the clause or is in the clause but only as the literal that is made *true* by the truth assignment. This probability is $(1-(1-p)^r)$. The probability that all clauses are falsified is therefore greater than $(1-(1-p)^r)^n$. □

Lemma 5 *If $p > (1+\epsilon)\ln(n)/r$ for any $\epsilon > 0$ then a random expression is not tautological with probability tending to 1.*

Proof:

From Lemma 4 the probability that all clauses are falsified by a random input vector is greater than $(1-(1-p)^r)^n$. By comparing the Talyor series expansion of e^{-p} with $1-p$ we have $(1-(1-p)^r)^n \geq (1-e^{-pr})^n$. By hypothesis we have $(1-e^{-pr})^n \geq (1-e^{-(1+\epsilon)\ln(n)})^n = (1-n^{-(1+\epsilon)})^n$. But $\lim_{n,r\to\infty}(1-n^{-(1+\epsilon)})^n = e^{-n^{-\epsilon}} = 1$. Thus, the probability that a random truth assignment is a falsifying truth assignment, if $p > (1+\epsilon)\ln(n)/r$, is 1 in the limit. Hence, a random expression is not tautological with probability tending to 1 if $p > (1+\epsilon)\ln(n)/r$ for any $\epsilon > 0$. □

On the other hand,

Lemma 6 *If $p = c\ln(n)/r$ for any $c < 1$ and $\lim_{n,r\to\infty} n^{1-c}/r = \infty$ then a random instance is tautological with probability tending to 1. If $p = \alpha(n)\ln(n)/r$ where $\alpha(n) = o(1)$ and $\lim_{n,r\to\infty} n/r = \infty$ then a random instance is tautological with probability tending to 1.*

Proof:

The probability that a random truth assignment satisfies a clause is the probability that the clause has at least one literal and each variable in V is either not in the clause or is in the clause but only as the literal that is made *true* by the truth assignment. The probability

that a clause contains at least one literal is $(1-(1-2p)^r)$. The probability that a clause is satisfied given it contains at least one literal is no less than $(1-p)^r$. Hence the probability that a random truth assignment satisfies a clause is no less than $(1-p)^r(1-(1-2p)^r)$ and the probability that a random truth assignment does not satisfy a clause is no greater than $(1-(1-p)^r(1-(1-2p)^r))$. The probability that a random truth assignment does not satisfy every clause is no greater than $(1-(1-p)^r(1-(1-2p)^r))^n$. The average number of truth assignments that do not satisfy every clause is no greater than $2^r(1-(1-p)^r(1-(1-2p)^r))^n$. Finally, the average number of truth assignments that do not satisfy every clause is an upper bound on the probability that there exists such a truth assignment. If $p = c\ln(n)/r$, where $c < 1$, then the average number of truth assignments that do not satisfy every clause is no greater than $e^{r\ln(2)-n^{1-c}}$. But $\lim_{n,r\to\infty} e^{r\ln(2)-n^{1-c}} = 0$ if $\lim_{n,r\to\infty} n^{1-c}/r = \infty$. If $p = \alpha(n)\ln(n)/r$ then the average number of non-satisfying truth assignments tends to 0 if $\lim_{n,r\to\infty} n/r = \infty$. □

In this section we have found conditions under which random expressions are redundant or tautological with probability tending to 1. To summarize, if $p < (1-\epsilon)\ln(n)/(2r)$, $\epsilon > 0$, and $pnr \to 0$ then almost all random expressions only have empty clauses. If $p < (1-\epsilon)\ln(n)/(2r), \epsilon > 0$, and $pn \to \infty$ then almost all random expressions have redundant clauses. If $p = \alpha(n)\ln(n)/(2r)$, $\alpha(n) = o(1)$, and $n/r \to \infty$, then almost all random expressions are tautologies. If $p = c\ln(n)/r$ for any $c < 1$ and $n^{1-c}/r \to \infty$ then almost all random expressions are tautologies.

In the next section we consider testability in the remaining parameter space.

4 Algorithms For Testability and Analysis

In this section we consider the performance of algorithms for the testability of random DNF expressions.

If $p > c\ln(n)/r$ for every fixed c then testability is very easy. All we need to do for each clause is set its literals to *true* to test for stuck-at-0 (separately, set one literal to *false* to test for stuck-at-1) and then randomly select the remaining truth assignments. We call this the *random-method.*

Lemma 7 *If $p = \alpha(n)\ln(n)/r$ where $\alpha(n)$ is any growing function of n then the random-method finds a complete set of test vectors for a DNF expression generated according to $M(n,r,p)$ with probability $n^{-O(\alpha(n))}$.*

Proof:

Since clauses are independent, the assignment is random with respect to all clauses but the one under test. Hence the probability that all clauses but the tested one are falsified by the assignment is greater than $(1-(1-p)^r)^n$. The average number of times such assignments fail to generate a valid test vector is less than $2n(1-(1-(1-p)^r)^n)$ since the total number

of tests is $2n$ (stuck-at-0 and stuck-at-1 for each clause). This is an upper bound on the probability that at least one truth assignment fails to generate a valid test vector. Let $\alpha(r)$ be any function that increases with r. Let $p = \alpha(r)\ln(n)/r$. Then the probability that at least one truth assignment fails to generate a valid test vector is bounded from above by $2n(1 - e^{-e^{-\alpha(r)\ln(n)}n}) = 2n(1 - e^{-n^{1-\alpha(r)}}) = n^{-O(\alpha(r))}$. □

Since the random-method needs to be used $2rn+2n$ times to generate all tests, the average number of times the method fails to produce a test is less than $rn^{-O(\alpha(r))}$. This is an upper bound on the probability of failure and tends to 0 as $n, r \to \infty$ if $n > r^{\epsilon}$ for any $\epsilon > 0$.

In the previous section we showed that random expressions are redundant if $pn \to \infty$ and $c < 1/2$. If $\lim_{n,r\to\infty} pn \neq \infty$, $c < 1/2$, and $\lim_{n,r\to\infty} n^{2-4c}/r = \infty$ then with probability tending to 1 the number of unit clauses in an expression is greater than $\sqrt{r^{1+\epsilon}}$ for any $\epsilon > 0$. If the number of unit clauses is greater than $\sqrt{r^{1+\epsilon}}$ then with probability tending to 1 there is at least one pair of unit clauses that is identical or one pair that is complementary. Then the expression is redundant or tautological.

Lemma 8 *If the number of unit clauses in a random expression e is greater than* $\sqrt{r^{1+\epsilon}}$ *for any* $\epsilon > 0$ *then with probability tending to 1 either there is an identical pair of unit clauses in e or else there is a complementary pair in e.*

Let m be the number of unit clauses The probability that there are no two clauses that are identical or complementary is $r!r^{-m}/(r-m)!$. By applying Stirling's approximation for factorials this is less than $(1 - m/r)^{m-r}e^{m}$. In the limit this is $e^{-m^2/r}$. The lemma follows. □

Lemma 9 *If* $p = \alpha(n)\ln(n)/r$ *where* $\alpha(n) < 1/2$, *and* $\lim_{n,r\to\infty} n^{2-4\alpha(n)}/r^{1+2\delta} = \infty$ *for any* $\delta > 0$ *then with probability tending to 1 the number of unit clauses in a random expression is at least equal to* $\sqrt{r^{1+2\delta}}$.

The number of unit clauses is binomially distributed with mean $2pr(1-2p)^{r-1}n$. By the Chernoff bound for the binomial distribution the probability that the number of unit clauses is less than $1 - r^{-\delta/2}$ times the mean is less than $e^{-r^{-\delta}2pr(1-2p)^{r-1}n}/3 = \delta'(n,r)$. Let the mean be greater than $\sqrt{r^{1+2\delta}}$. Then the probability that the number of unit clauses is less than $\sqrt{r^{1+\delta}}$ is bounded from above by $\delta'(n,r) \to 0$. The mean is greater than $\sqrt{r^{1+2\delta}}$ if $\lim_{n,r\to\infty} n^{1-2\alpha(n)} > \sqrt{r^{1+2\delta}}$. The lemma follows. □

On the other hand, if $\lim_{n,r\to\infty} pn \neq \infty$, $c < 1/2$, and $\lim_{n,r\to\infty} n^{2-2c}/r = 0$ then variables are in so few clauses that with probability tending to 1 every clause contains either a literal that is not present in any other clause or contains a literal l such that every other clause that contains l or its complement also contains a literal that is not present in any other clause. We call this property of instances property P. A literal that appears in only one clause is said to be solitary.

An instance with property P, is either redundant or testable in polynomial time. To generate a stuck-at-1 test for a particular clause h in an irredundant expression, set all solitary literals to *false* and their complements to *true* then set all unset literals satisfying property P in clauses not yet falsified to *false* and their complements to *true*. As a result, all clauses will have value *false*. To generate a stuck-at-0 test for h, set all literals in h to *true* and their complements to *false*. Then set all solitary literals, excluding those of h, to *false* and their complements to *true*. Next, set all unset literals satisfying property P in clauses not yet falsified to *false* and their complements to *true*. All the clauses that have not yet been falsified contain literals that satisfy property P and have value *true* (from h); and/or do not satisfy property P. For such clauses, if there are no literals of the latter kind then the instance is redundant; if there are such literals, pick one that is unset and set it to *false* and its complement to *true*. If some clause other than h is still not falsified then h is redundant. Otherwise a test has been generated.

We now show that almost all random instances have property P under the conditions stated.

Lemma 10 *If $p = \alpha(n)\ln(n)/r$ where $\alpha(n) < 1/2$, and for any $\epsilon > 0$, $\lim_{n,r\to\infty} n^{2-2\alpha(n)}/r^{1+\epsilon} = 0$ then a random expression has property P with probability tending to 1.*

Consider any clause a and suppose it has x literals. We shall say that a is semi-valid if a contains no literals or or there is another literal in a which does not appear in any other clause. We find the result only for semi-valid clauses; this will bound the result stated in the hypothesis.

The probability that a has x literals is $\binom{r}{x}(2p)^x(1-2p)^{r-x}$. The probability that a variable does not exist in a clause other than a is $(1-2p)^{n-1}$. Hence, the probability that at least one literal of a is not in any other clause given a has x literals is $1-(1-(1-2p)^n)^x$. The probability that a is semi-valid

$$\begin{aligned}
&\sum_{x=1}^{r}\binom{r}{x}(2p)^x(1-2p)^{r-x}(1-(1-(1-2p)^n)^x)+(1-2p)^r\\
=\;& 1-\sum_{x=1}^{r}\binom{r}{x}(2p)^x(1-2p)^{r-x}(1-(1-2p)^n)^x\\
=\;& 1-(1-2p+2p(1-(1-2p)^n))^r+(1-2p)^r\\
=\;& 1-(1-2p(1-2p)^n)^r+(1-2p)^r.
\end{aligned}$$

Applying the binomial theorem and summing gives

$$\begin{aligned}
1-(1-2p(1-2p)^n)^r+(1-2p)^r &= 1-pn(1+O(pn))\sum_{i=1}^{r}\binom{r}{i}2i(-2p)^i\\
&= 1-4p^2nr(1-2p)^r(1+O(pn)).
\end{aligned}$$

Thus, the probability that a is not semi-valid is $4p^2nr(1-2p)^r(1+O(pn))$. The average number of such clauses is $4p^2n^2r(1-2p)^r$. This is an upper bound on the probability that some clause in a random expression is not semi-valid. By the conditions of the hypothesis this is less than $4c^2(\ln(n))^2n^{2-2\alpha(n)}/r$. The lemma follows. □

From Lemmas 8,9,10 and the results of Section 2 we conclude that either there exist algorithms which find complete test sets for random expressions with probability tending to 1 or such expressions are redundant or tautological with probability tending to 1 if $p < \ln(n)/(2r)$. We now concentrate on the last remaining range of p.

If $p = c\ln(n)/r$, $c > 1/2$, and $\lim_{n,r\to\infty} n^{1-c}/r = 0$ then random instances are not tautological and do not possess property P with probability tending to 1. In this case we can use a variant of an algorithm called *UNSAT-FINDER* to generate complete tests with probability tending to 1. *UNSAT-FINDER*, takes a DNF Boolean expression I as input and either finds a truth assignment which does not satisfy I or gives up. Let V be the set of Boolean variables from which a random expressione I is constructed. Let L denote the set of literals associated with V. Let $var : L \to V$ be a mapping from literals to their associated variables. Let $T : V \to \{true, false, unset\}$ be a mapping from variables to Boolean values. T specifies a truth assignment to the variables in I; the value *unset* is treated as a *don't care*. It is constructed and returned by *UNSAT-FINDER* if the algorithm does not give up.

UNSAT-FINDER(I):
For all $v \in V$, set $T(v) = unset$
Repeat
Let C_{min} denote the collection of clauses in I with the least number of literals
Randomly select literal l from C_{min}
If l is positive set $T(var(l)) = false$, Otherwise set $T(var(l)) = true$
set $I = \{c - comp(l) | c \in I, l \notin c\}$
Until there exists a null clause or $I = \Phi$
If $I = \Phi$ then return T, Otherwise give up

It should be clear that if *UNSAT-FINDER* returns a truth assignment, that truth assignment falsifies the original expression I. It should also be clear that *UNSAT-FINDER* runs in time bounded by a polynomial in the length of I. The conditions for which the probability that *UNSAT-FINDER* gives up tends to 0 are also the conditions for which random instances are not tautological. We wish to find those conditions when $p < \ln(n)/r$ (since we already know that instances are not tautological in probability when $p > \ln(n)/r$). The result is similar to a dual result on CNF Boolean instances and a "unit-clause" algorithm that was presented in [3]. However, that result is not strong enough to be applied here. Therefore, in what follows we strengthen the result of [3] and adapt it to *UNSAT-FINDER* while lifting some lemmas directly from that paper.

For the sake of simplifying the analysis we suppose that *UNSAT-FINDER* selects a literal from a smallest clause only if there exist clauses of size less than $\ln(2pr)$ (recall that $2pr$ is the average size of clauses). Otherwise, *UNSAT-FINDER* selects a literal randomly from the set of unset literals. In the analysis below references to *UNSAT-FINDER* are to this modified version.

We need to find the probability that no null clauses are generated by *UNSAT-FINDER*. We do so by modeling the flow of clauses through various states during execution of the algorithm. During each iteration of *UNSAT-FINDER* clauses containing the chosen literal are removed from I (because they are clauses that become falsified by the assignment $T(var(l))$) and occurrences in

I of the complement of the chosen literal are removed (because they represent literals that have value *true* as a result of the assignment $T(var(l))$). Let $C_i(j)$ denote the collection of clauses in I containing exactly i literals at the start of the $j+1^{st}$ iteration. Let $J(x,y,z)$ denote the probability distribution over DNF Boolean expressions where x clauses are chosen uniformly and independently from the set of all possible z-literal clauses that can be constructed from y Boolean variables.

Theorem 1 *Given $|C_i(j)| = n_i(j)$, for all $1 \leq i \leq r-j$, the clauses in $C_i(j)$ are distributed according to $J(n_i(j), r-j, i)$ independently of the clauses in $C_l(j)$, $l \neq i$.*

Proof:

This is certainly true for the case $j = 0$. Suppose it is true for all $0 \geq j \geq m$. There are two ways the $m+1^{st}$ selected literal is chosen: from the set of smallest clauses $C_k(m)$ if there exists a $k < \ln(2pr)$ such that $C_k(m) \neq \Phi$, and randomly from the set of unassigned literals otherwise. Consider the second case. By hypothesis, if h_i clauses of $C_i(m)$ contain the selected literal or its complement, the remaining $n_i(m) - h_i$ clauses of $C_i(m)$ are distributed according to $J(n_i(m)-h_i, r-m-1, i)$. If g_{i+1} clauses of $C_{i+1}(m)$ contain the complement of the selected literal, stripping occurrences of that literal from those clauses results in a collection of g_{i+1} clauses distributed according to $J(g_{i+1}, r-m-1, i)$. Combining the two collections results in $n_i(m) - h_i + g_{i+1}$ clauses distributed according to $J(n_i(m) - h_i + g_{i+1}, r-m-1, i) = J(n_i(m+1), r-(m+1), i)$. Now consider the case that a literal is selected from $C_k(m)$, $k < \ln(2pr)$. Except for the clause from which this literal was selected, the clauses of I are independent of the selected literal. Hence the previous argument applies. □

Let X be a random variable and let $E\{X\}$ denote the expectation of X. Let $w_i(j)$ denote the number of clauses entering $C_i(j+1)$ as a result of selecting the $j+1^{st}$ literal (there is no dependence on instance given here since we will soon take expectations). Let $z_i(j)$ denote the number of clauses leaving $C_i(j)$ as a result of selecting the $j+1^{st}$ literal. Let $n_i(j) = |C_i(j)|$. A set of recurrence relations for $E\{n_i(j)\}$, $1 \leq i, j \leq r$, in terms of $E\{w_i(j)\}$ and $E\{z_i(j)\}$ can be developed as follows:

$$E\{n_i(j+1)\} = E\{n_i(j)\} + E\{w_i(j)\} - E\{z_i(j)\}. \tag{1}$$

Because of Theorem 1, for all $\ln(2pr) \leq i \leq r$

$$E\{z_i(j)\} = E\{E\{z_i(j)|n_i(j)\}\} = \sum_{l=0}^{\infty} \frac{i \cdot l}{r-j} Pr(n_i(j) = l) = \frac{i \cdot E\{n_i(j)\}}{r-j}.$$

Also, $E\{w_r(j)\} = 0$ and for all $\ln(2pr) \leq i < r$

$$E\{w_i(j)\} = E\{E\{w_i(j)|n_{i+1}(j)\}\} = \sum_{l=0}^{\infty} \frac{(i+1)l}{2(r-j)} Pr(n_{i+1}(j) = l) = \frac{(i+1)E\{n_i(j)\}}{2(r-j)}.$$

The recurrence relations for $1 \leq i < \ln(2pr)$ depend on $p_i(j)$, the probability that the selected literal l is chosen from $C_i(j)$. In this case we have

$$\begin{aligned} E\{z_i(j)\} &= E\{z_i(j)|l \notin C_i(j)\}(1-p_i(j)) + E\{z_i(j)|l \in C_i(j)\}p_i(j) \\ &= \frac{iE\{n_i(j)\}}{r-j} + p_i(j)\left(1 - \frac{i}{r-j}\right) \end{aligned}$$

and

$$\begin{aligned} E\{w_i(j)\} &= E\{w_i(j)|l \notin C_{i+1}(j)\}(1-p_{i+1}(j)) + E\{w_i(j)|l \in C_{i+1}(j)\}p_{i+1}(j) \\ &= \frac{(i+1)E\{n_{i+1}(j)\}}{2(r-j)} - p_{i+1}(j)\frac{i+1}{2(r-j)}. \end{aligned}$$

From Theorem 3.3 of [3] we have

Lemma 11 *For all* $\ln(2pr) \leq i \leq r$ *and* $1 \leq j \leq r$,

$$E\{n_i(j)\} = \binom{r-j}{i}(2p)^i(1-p)^j(1-2p)^{r-i-j}n. \tag{2}$$

Since the number of literals in a clause of a random expression is binomially distributed,

Lemma 12 *For all* $1 \leq i \leq r$

$$E\{n_i(0)\} = \binom{r}{i}(2p)^i(1-2p)^{r-i}n$$

From Lemma 10 the conditions which insure $E\{w_{\ln(2pr)-1}(j)\} < r^{-\epsilon}$ for any $\epsilon > 0$ can be obtained.

Lemma 13 *For any* $\epsilon > 0$, *if* $e^{\ln(2)+2+(p\ln(2)+\ln(pr))/(1-p)-pr}n/r < r^{-\epsilon}$ *then for all* $1 \leq j \leq r$, $E\{w_{\ln(2pr)-1}(j)\} < r^{-\epsilon}$.

Proof:

$$E\{w_{\ln(2pr)-1}(j)\} \leq \frac{\ln(2pr)E\{n_{\ln(2pr)}(j)\}}{2(r-j)}.$$

The maximum of $E\{n_{\ln(2pr)}(j)\}/(r-j)$ can be found by determining the value of j which makes

$$\frac{(r-j)E\{n_{\ln(2pr)}(j+1)\}}{(r-j-1)E\{n_{\ln(2pr)}(j)\}} = 1$$

and substituting that value in (2). Thus, we want j such that

$$\frac{(r-j)E\{n_{\ln(2pr)}(j+1)\}}{(r-j-1)E\{n_{\ln(2pr)}(j)\}} = \frac{(r-j)\binom{r-j-1}{\ln(2pr)}(2p)^{\ln(2pr)}(1-p)^{j+1}(1-2p)^{r-\ln(2pr)-j-1}n}{(r-j-1)\binom{r-j}{\ln(2pr)}(2p)^{\ln(2pr)}(1-p)^{j}(1-2p)^{r-\ln(2pr)-j}n}$$

This is satisfied by $j = j_{max} = r - (1-p)(\ln(2pr)-1)/p$. Making use of Stirling's approximation in the third step below we have,

$$\begin{aligned}
&\frac{E\{n_{\ln(2pr)}(j_{max})\}}{(r-j_{max})} \\
&= \frac{p}{(1-p)(\ln(2pr)-1)}\binom{(1-p)(\ln(2pr)-1)/p}{\ln(2pr)}(2p)^{\ln(2pr)}(1-p)^{r-(1-p)(\ln(2pr)-1)/p} \\
&\qquad \times(1-2p)^{(1-p)(\ln(2pr)-1)/p-\ln(2pr)}n \\
&\leq \frac{p}{(1-p)\ln(2pr)}\frac{(2\ln(2pr))^{\ln(2pr)}}{\ln(2pr)!}(1-p)^{r-(1-2p)\ln(2pr)/p}(1-2p)^{(1-2p)\ln(2pr)/p}\left(\frac{1-p}{1-2p}\right)^{(1-p)/p}n \\
&< \frac{p}{(1-p)\ln(2pr)}(2e)^{\ln(2pr)}(1-p)^{r}\left(\frac{1-2p}{1-p}\right)^{(1-2p)\ln(2pr)/p}e^{2}n \\
&< \frac{p}{(1-p)\ln(2pr)}(2e)^{\ln(2pr)}e^{-pr}e^{-(1-2p)\ln(2pr)/(1-p)}e^{2}n \\
&< \frac{p}{\ln(2pr)}e^{\ln(2)-pr+p\ln(2pr)/(1-p)+2}n.
\end{aligned}$$

Thus, for all $1 \leq j \leq r$

$$E\{w_{\ln(2pr)-1}(j)\} < pne^{\ln(2)+2+p\ln(2pr)/(1-p)-pr} < e^{\ln(2)+2+(p\ln(2)+\ln(pr))/(1-p)-pr}n/r.$$

The lemma follows. □

Lemma 5 says that non-tautological instances are produced with probability tending to 1 only if $\lim_{n,r\to\infty} e^{-pr}n/r = 0$. Lemma 12 says that $E\{w_{\ln(2pr)-1}(j)\} < r^{-\epsilon}$ for all $1 \leq j \leq r$ if $\lim_{n,r\to\infty} e^{-pr}n/r^{1+\epsilon} = 0$ for any $\epsilon > 0$. Thus, in some sense for almost all non-tautological instances, $E\{w_{\ln(2pr)-1}(j)\} < r^{-\epsilon}$ for $1 \leq j \leq r$. It will be shown that if $E\{w_{\ln(2pr)-1}(j)\} < r^{-\epsilon}$ for $1 \leq j \leq r$ and any $\epsilon > 0$ then with probability tending to 1 *UNSAT-FINDER* will find a truth assignment for each clause such that the clause has value *true (false)* and all other clauses have value false. Thus, with *UNSAT-FINDER* it is possible to generate a complete set of test vectors for a PLA with probability tending to 1.

The following theorem will make our work much easier.

Theorem 2 *The probability there exists a clause of size less than* $\ln(2pr)$ *in a random instance tends to 0 if* $p = c\ln(n)/r$ *where* $c > 1/2$.

Proof:

The probability that a clause has size no greater than $\ln(2pr)$ is

$$\sum_{l=0}^{\ln(2pr)} \binom{r}{l} (2p)^l (1-2p)^{r-\ln(2pr)}.$$

Since $\ln(2pr)$ is much less than the mean pr, this sum is less than $(2pre)^{\ln(2pr)} e^{-2pr}$. Thus, the probability that all clauses have size greater than $\ln(2pr)$ is at least

$$\left(1-(2pre)^{\ln(2pr)} e^{-2pr}\right)^n.$$

If $p = c\ln(n)/r$ this is

$$\left(1-(2pre)^{\ln(2pr)} n^{-2c}\right)^n > e^{-(4pre)^{\ln(2pr)} n^{1-2c}}.$$

But $(2pre)^{\ln(2pr)} = (2ec\ln(n))^{\ln(2c\ln(n))}$ cannot grow as fast as n^ϵ for any $\epsilon > 0$. Hence the exponential tends to 1 if $c > 1/2$ and the lemma is proved. □

Suppose after creating a random instance we set $n_i(j) = 0$ for all $1 \le i \le \ln(2pr) - 1$ (that is, we eliminate all clauses of length less than $\ln(2pr)$). With probability tending to 1 we do not have to do so. Hence the result will apply to almost all random expressions.

Lemma 14 *For all* $1 \le i \le \ln(2pr) - 1$, $\lim_{n\to\infty} p_i(j) \to E\{w_i(j)\} - E\{w_{i-1}(j)\}$

Proof:

$$p_i(j) = pr(C_l(j) = \phi, l = 1, 2, ..., i-1) - pr(C_l(j) = \phi, l = 1, 2, ..., i).$$

Since $Ew_{\ln(2pr)-1}(j) < r^{-\epsilon}$ the average number of iterations between successive occurrences of the event that a clause moves from C_i to C_{i-1} increases with increasing n and r. Hence, asymptotically, the probability that there is at least one clause in $C_l(j)$, $l = 1, 2, ..., i-1$, is the flow of clauses into $C_{i-1}(j)$. This is $E\{w_{i-1}(j)\}$. Similarly, the probability that there is at least one clause in $C_l(j)$, $l = 1, 2, ..., i$, is $E\{w_i(j)\}$. Hence, $p_i(j)$ tends to $(1 - E\{w_{i-1}(j)\}) - (1 - E\{w_i(j)\}) = E\{w_i(j)\} - E\{w_{i-1}(j)\}$. □

Then we can write for $1 \le i \le \ln(2pr) - 1$, $1 \le j \le r$

$$\begin{aligned}
E\{n_i(j+1)\} &= \left(1-\frac{i}{r-j}\right) E\{n_i(j)\} + \left(1-\frac{i}{r-j}\right) E\{w_{i-1}(j)\} + \frac{i}{r-j} E\{w_i(j)\} \\
&= \left(1-\frac{i}{r-j}\right)\left(1+\frac{i}{2(r-j)}\right) E\{n_i(j)\} + \frac{i}{r-j} E\{w_i(j)\} \\
&\le \left(1-\frac{i}{2(r-j)}\right) E\{n_i(j)\} + \frac{i(i+1)}{2(r-j)^2} E\{n_{i+1}(j)\}.
\end{aligned}$$

Theorem 3 *Let* $E\{n_i(0)\} = 0$ *for all* $1 \leq i < \ln(2pr)$. *For all* $1 \leq i < \ln(2pr)$, $1 \leq j \leq r$, $\epsilon > 0$

$$E\{n_i(j)\} < \frac{(\ln(2pr))^{\ln(2pr)}}{i^i(r-j)^{(k-i)}} \binom{r-j}{\ln(2pr)} (2p)^{\ln(2pr)}(1-2p)^{r-\ln(2pr)-j}n$$

if $\lim_{n,r\to\infty} e^{-pr}n/r^{1+\epsilon} = 0$.

Proof:

By double induction. The basis step at each i holds since $E\{n_i(0)\} = 0$. The induction step proceeds as follows:

$$\begin{aligned}
E\{n_i(j+1)\} &= \left(1 - \frac{i}{2(r-j)}\right) E\{n_i(j)\} + \frac{i(i+1)}{2(r-j)^2} E\{n_{i+1}(j)\} \\
&< \frac{(2(r-j)-i)(\ln(2pr))^{\ln(2pr)}\binom{r-j}{\ln(2pr)}(2p)^{\ln(2pr)}(1-2p)^{(r-\ln(2pr)-j)}n}{2(r-j)i^i(r-j)^{(\ln(2pr)-i)}} \\
&\quad + \frac{i(i+1)(\ln(2pr))^{\ln(2pr)}\binom{r-j}{\ln(2pr)}(2p)^{\ln(2pr)}(1-2p)^{r-\ln(2pr)-j}n}{(i+1)^{(i+1)}(r-j)^{(\ln(2pr)-i-1)}2(r-j)^2} \\
&= \frac{(\ln(2pr))^{\ln(2pr)}\binom{r-j}{\ln(2pr)}(2p)^{\ln(2pr)}(1-2p)^{r-\ln(2pr)-j}n}{2(r-j)^{(\ln(2pr)-i+1)}} \left(\frac{i}{(i+1)^i} + \frac{2(r-j)-i}{i^i}\right) \\
&= \frac{(\ln(2pr))^{\ln(2pr)}\binom{r-j}{\ln(2pr)}(2p)^{\ln(2pr)}(1-2p)^{(r-\ln(2pr)-j)}n}{i^i 2(r-j)^{(\ln(2pr)-i)}} \left(1 - \frac{i(1-e^{-1})}{2(r-j)}\right) \\
&\leq \frac{(\ln(2pr))^{\ln(2pr)}\binom{r-j}{\ln(2pr)}(2p)^{\ln(2pr)}(1-2p)^{(r-\ln(2pr)-j-1)}n}{i^i 2(r-j)^{(\ln(2pr)-i)}} \cdot \frac{(r-j)^{\ln(2pr)-i}}{(r-j-1)^{\ln(2pr)-i}}.
\end{aligned}$$

This proves the theorem. □

Theorem 4 *Let* $E\{n_i(0)\} = 0$ *for all* $1 \leq i < \ln(2pr)$. *For any* $\epsilon > 0$, *all* $p < c\ln(n)/r$, *and* $1 \leq j \leq r$,

$$E\{w_1(j)\} \leq \frac{(\ln(2pr))^{\ln(2pr)}n}{r^{\ln(2pr)}}$$

if $\lim_{n,r\to\infty} e^{-pr}n/r^{1+\epsilon} = 0$.

Proof:

From the previous theorem

$$E\{w_1(j)\} \leq (\ln(2pr))^{\ln(2pr)} \binom{r-j}{\ln(2pr)} (2p)^{\ln(2pr)}(1-2p)^{(r-\ln(2pr)-j)}n.$$

Using the technique of Lemma 13 we find that this expression has a maximum at $j = r - 1/2p$. Substituting back into the expression gives the result. □

Theorem 5 *The probability that UNSAT-FINDER does not find a falsifying truth assignment is bounded from above by* $\ln(n)\ln(r)E^*\{w_1\}$ *where* $E^*\{w_1\}$ *is the maximum flow of clauses into* C_1.

Proof:

Similar to the proof of Theorem 3.4 in [3]. □

From Theorems 4 and 5 we have the probability that *UNSAT-FINDER* does not find a falsifying truth assignment is bounded from above by

$$\frac{\ln(n)\ln(r)(\ln(2pr))^{\ln(2pr)}n}{r^{\ln(2pr)}} \tag{3}$$

if $p = c\ln(n)/r$, $c > 1/2$, and $\lim_{n,r\to\infty} n^{1-c}/r = 0$. This is less than $1/(r^{1+\epsilon}n)$ if $n < 2^r$. If $n > 2^r$ falsifiability may be found in polynomial time by exhaustive search.

We make use of *UNSAT-FINDER* to generate tests in the same way that we made use of the random-method except that we use *UNSAT-FINDER* for generating the truth assignments. Since we have to make $2rn + n$ tests the average number of failed tests is less than $O(\ln(n)n^2/r^{\ln\ln(n)})$ from (3). This is an upper bound on the probability that a test fails and tends to 0 as $n, r \to \infty$ under the conditions stated.

5 Conclusions

In this paper we have examined the impact of primality and irredundancy on the ease of testability of two-level circuits. We have shown that for a large class of prime and irredundant circuits that with probability tending toward 1, the circuit may be fully tested in polynomial time. Thus for this restricted class of circuits we have shown that full testability does indeed imply easy testability with high probability .

References

[1] K. Bartlett *et al.*, Multilevel logic minimization using implicit don't cares, *IEEE Transactions on Computer-Aided Design of Integrate Circuits and Systems*, 17, 6 , pp. 723-740, June , 1988 .

[2] R. K. Brayton and C. McMullen and G. D. Hachtel and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, 1984 .

[3] J. Franco, and Y. C. Ho, Probabilistic performance of a heuristic for the Satisfiability problem, *Discrete Applied Mathematics*, 22, pp. 35-51, 1988/89 .

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

[5] G. D. Hachtel and R. M. Jacoby and K. Keutzer and C. R. Morrison, On the relationship between area optimization and multifault testabilty of multilevel logic, *Proceedings of the International Workshop on Logic Synthesis*, June, 1989.

[6] D. S. Johnson, The NP-Completeness column: an ongoing guide, *Journal of Algorithms*, **6**, pp. 291-305, 1985 .

[7] I. Kohavi and Z. Kohavi, Detection of multiple faults in combinational logic networks, *IEEE Transactions on Computers*, **C21** , 6 , pp. 556-568, June , 1972.

[8] E. J. McCluskey, Minimization of Boolean functions, *Bell Lab. Technical Journal*, Bell Lab., **35**, pp. 1417-1444, November, 1956.