

EFFICIENT VLSI FAULT SIMULATION

JOHN H. REIF

Department of Computer Science

Duke University

Durham, NC 27706, U.S.A.

Abstract—Let C be an acyclic Boolean circuit with n gates and $\leq n$ inputs. A circuit manufacture error may result in a "Stuck-at" (S-A) fault in a circuit identical to C except a gate v only outputs a fixed Boolean value. The S-A fault simulation problem for C is to determine all possible (S-A) faults which can be detected (i.e., faults for which a faulty circuit and C would give distinct outputs) by a given test pattern input.

We consider the case where C is a tree (i.e., has fan-out 1.)

We give a practical algorithm for fault simulation which simultaneously determines all detectable S-A faults for every gate in the circuit tree C . Our algorithm required only the evaluation of a circuit $FS(C)$ which has $\leq 7n$ gates and has depth $\leq 3(d+1)$, when d is the depth of C . Thus the sequential time of our algorithm is $\leq 7n$, and the parallel time is $\leq 3(d+1)$. Furthermore, $FS(C)$ requires only a small constant factor more VLSI area than does the original circuit C .

We also extend our results to get efficient methods for fault simulation of oblivious VLSI circuits with feedback lines.

1. INTRODUCTION

1.1. Faults in VLSI Manufacture

In a very short period of time, integrated circuits have grown in complexity from a single semiconductor, to the now current Large Scale Integration (LSI) with up to $n \approx 10^4$ gates, and recently the advent of Very Large Scale Integration (VLSI) with feasibility of $n \approx 10^5$ gates or more.

Unfortunately, the manufacture of LSI and VLSI yields faulty circuits with a frequency that grows very quickly with the number n of gates. In theory, the yield rate decreases exponentially with the number of gates, and this is borne out in practice. For LSI with $n \approx 10^4$, a yield rate of 50% would be considered extraordinarily good, and for VLSI with $n \approx 10^5$, one can expect at best a yield rate in nonfaulty circuits of a few percentages. Since a large integrated circuit can have only a limited number of input/output pods, it is not practical to directly test the performance of each individual gate of such a circuit. Instead, a set of input vectors called test patterns are input to the possibly faulty circuit, and the resulting output is compared with the output of a known faultless circuit. The set of test patterns cover a given fault type τ if they distinguish all faults of type τ .

1.2. The Fault Model

To make fault detection tractable, the integrated circuit is generally assumed to be acyclic, with no feedback lines. If feedback lines exist, either (1) they are simply cut, resulting in an acyclic network with some additional inputs or preferably, (2) the sequential logic is oblivious

We would like to thank Carl Lieberherr for his useful comments on this paper. This work has been supported in part by Air Force Contract AFOSR-87-0386, DARPA/ISTO Contract N00014-88-K-0458, NASA/CESDIS Contract 550-63 NAS 5-30428 URSA. Approved for public release: distribution is unlimited.

to the inputs and so can be unraveled to form an acyclic network. (If required, the circuit can be made into a tree, by further cutting lines so all gates have fan-out ≤ 1 .) The integrated circuit is thus modeled as an acyclic, Boolean circuit C . A *Stuck-At (S-A) fault* is a gate which is permanently set to a Boolean value b . In particular, a *S-A-0 (S-A-1)* fault is where the gate is set to $b = 0$ ($b = 1$), respectively. The S-A is the most common fault type tested for, and of the most importance for currently used technologies such as nMOS. We will consider only the S-A fault type in this paper.

Test generation is the problem of finding a set of test patterns, and *(S-A) fault simulation* is the problem of detecting those S-A faults which a given test pattern covers. *Fault coverage* is the problem of finding a set of test patterns which cover all possible S-A faults of the circuit. These problems—as our extensive references will indicate—have been very thoroughly and extensively studied since the early 1960s up to the current time. In spite of considerable progress in the use of heuristic algorithms, many fundamental problems still remain, and become more difficult as the circuit size n of current VLSI continues to very quickly grow.

1.3. Previous Work in Test Generation

In theory, finding a test vector which detects a given fault is NP complete [1]. S-A fault test generation is done in practice by a number of known heuristic methods including path sensitization [2–5], and the D-algorithm [6,7].

Goldberg & Leiberherr [8] have an ingenious algorithm for test pattern generation in the special case that the given Boolean circuit is a tree; in this case they reduce test pattern generation to a simple parity computation which can be done in linear time. Hence, their method for test pattern generation requires a total of $O(n^2)$ time to generate test patterns for the $2n$ possible S-A faults.

Another increasingly popular, and practically very efficient method for test generation is to simply generate a random set of test inputs using an efficient pseudorandom number generator. This was first suggested by Schnurmann *et al.* [9]. Various authors [9–14] have developed analytic methods for testing the effectiveness of random testing. In many cases, they show that a small number of random test patterns cover almost all $2n$ possible S-A faults, whereas the previously sighted deterministic methods may require a distinct test vector for each possible S-A fault.

A large number of sufficiently random test patterns can be very easily generated by a feedback shift register circuit [15–17]. The response of the circuit tested can then be compressed by the signature methods of Carter [18,19].

1.4. Previous Work in Fault Simulation

This paper is concerned with developing an efficient method for S-A fault simulation, i.e., to determine exactly all S-A faults which are covered by a given test pattern. All previously known methods [3,4,6,20–30] for S-A fault simulation of a general Boolean circuit of size n require cn^2 time for some constant $c \geq 1$. This quadratic time bound is already very costly for moderate size LSI since if $n \geq 10^4$, then at least $n^2 = 10^{10}$ steps required by these previous methods seems unfeasible even on the fastest known sequential machine. Recently, Abramovici *et al.* [30] give a linear time sequential algorithm for fault simulation of Boolean circuit trees.

1.5. Our Results

In this paper, we restrict our attention primarily to the case where C is a circuit tree, i.e., it has fan-out 1. (However, in the last section, we describe how to extend our techniques to circuits with fan-out > 1 .)

We describe a method for constructing a circuit $FS(C)$ (for “Fault Simulation”) which simultaneously determines all detectable S-A faults for every gate of the circuit tree C . The circuit $FS(C)$ has in the worst case $\leq 7n$ gates, (but in many cases, where C does not have a majority of v gates, $FS(C)$ has only $6n$ gates). The VLSI area of $FS(C)$ is only a small constant factor more than the VLSI area of C . The sequential execution time of our algorithm is just that used to evaluate $FS(C)$, which is at most $\leq 7n$ steps. Even in applications such as VLSI with $n = 10^5$ gates, sequential execution of our algorithm takes only a fraction of a second on conventional machines. Furthermore, our algorithm is inherently parallel, since all that is required is evaluation

of the circuit $FS(C)$. The parallel time for evaluation of \hat{C} is its depth which is $\leq 3(d+1)$, if C has depth d .

The advantage of our method over that of Abramovici *et al.* [30] is that we explicitly construct a Boolean circuit for fault simulation, whereas theirs is a sequential method. Our method is also significantly different.

The approach we use to solve the S-A fault simulation problem is to compute the Boolean differences (the mathematical theory of the Boolean difference is discussed in [31–33]) of the circuit output with respect to values computed by the circuit at each of its gates. This approach is discussed in [26,28,34–36], but was previously considered inefficient because of the apparently large amount of algebraic manipulation required.

Baur and Strassen [37] gave an $O(n)$ time straight line algorithm for all the partial derivatives of a polynomial over an infinite field computed by a straight line program of depth n , and Jerrum [38] gave a more combinatorial proof of this result. We use related techniques for the circuit construction of the Boolean differences over the field $GF(2)$, to solve the S-A fault simulation problem.

1.6. A VLSI Chip for Fault Simulation and Coverage

We show that if the original Boolean circuit tree C has VLSI area A , then the VLSI area required for our on-fault simulation circuits $FS(C)$ is only a small constant factor more than A . (The constant factor depends on the VLSI technology used.)

Note that because of the simplicity and locality properties of our rules for generating C' from C , this construction can easily be done automatically by known VLSI hardware designed layout tools (for example, those of Lieberherr *et al.* [39,40]) without any participation by the original designer of C .

Our suggestion for the practical utilization of our results is as follows: Given an acyclic Boolean VLSI circuit C , we would construct "Fault Coverage" VLSI chip $FC(C)$ which contains both our fault simulator $FS(C)$ as well as a feedback shift register for generating pseudorandom test inputs. We do not need to output the S-A faults detected by $FS(C)$ for each test pattern. Instead we keep two Boolean counters (initially 0) for each gate v of C , which indicates whether a S-A-0 or a S-A-1 fault at v has been covered by a previous test input. Given a new test input vector these counters can easily be updated by a single Boolean operation at the appropriate gate of $FS(C)$. The new test pattern is output only if it covers at least one S-A fault not previously covered. We can then very quickly output from $FC(C)$ a sequence of nonredundant test vectors until these test vectors cover all S-A faults. If it happens that not all S-A faults are covered after sampling a given number of pseudorandom test vectors, then $FC(C)$ would indicate that C must be redesigned appropriately. However, as the above reference on the analysis of random testing indicates, a small number of random test vectors usually suffice to cover almost all S-A faults.

1.7. Fault Simulation of Circuits with Feedback Lines

We generalize our results to a Boolean circuit S with feedback lines (i.e., S may have cycles). Suppose S has n gates and parallel computation time d . We can unravel S into an acyclic Boolean circuit tree $C(S)$ with $n(d+1)$ gates and depth d . Thus, we can apply our previously described S-A fault simulation algorithm to $C(S)$, yielding a fault simulation circuit $FS(C(S))$ of $7n(d+1)$ gates and depth $3(d+1)$.

Suppose furthermore, S has VLSI area A using d layers. Then we can avoid actual construction of $C(S)$ by repeatedly recomputing C when required. This idea yields a VLSI circuit (with feedback lines) for fault simulation with $O(n)$ gates, $O(A)$ area, $O(\ell)$ degrees, and parallel time bound d^2 . S is *reversible* if it can be made to run its computation both forward in time and in reverse (i.e., it can compute in constant time both its immediately previous state as well as its immediate succeeding state). Many common sequential circuits are naturally reversible in this sense, for example the Cooley-Tukey FFT circuit, and other convolution circuits, the Benes switching circuit, Batcher's sorting network, and the DES cryptographic standard. If S is reversible, we show how to construct a fault simulation circuit for detecting all S-A faults (with feedback lines) which has only small constant factors, more size, area, layers and parallel computation time than that of the original circuit S .

1.8. Organization of the Paper

This paper is organized as follows. In Section 2 we formally define a Boolean circuit and the Boolean differences. In Section 3, we give our circuit construction for computing the Boolean differences and solving the S-A fault simulation problem for Boolean circuit trees. In Section 4, we prove the correctness of our Boolean differences circuit. In Section 5, we describe how we may extend these techniques to acyclic circuits with fan-out > 1 .

2. PRELIMINARY DEFINITIONS

2.1. The Boolean Difference

Let $f(\vec{x})$ be a Boolean function with indeterminates $\vec{x} = (x_1, \dots, x_m)$. The *Boolean difference* of $f(\vec{x})$ with respect to x_i is

$$\frac{df(\vec{x})}{dx_i} = f(x_1, \dots, x_{i-1}, 0, x_i, \dots, x_n) \oplus f(x_1, \dots, x_{i-1}, 1, x_i, \dots, x_n).$$

(That is, $\frac{df(\vec{x})}{dx_i}$ is the exclusive of $f(\vec{x})$ with x_i set to 0, and of $f(\vec{x})$ with x_i set to 1.) We say $f(\vec{x})$ is *sensitive to x_i* if the value of f changes, depending on whether x_i is 0 or 1. Thus, $\frac{df(\vec{x})}{dx_i} = 1$ if and only if $f(\vec{x})$ is sensitive to x_i .

2.2. Boolean Circuits

A *Boolean circuit* C is a triple $(V, v_0, E, L, \Sigma, OP)$ where

- (i) V is a set of n gates (or nodes) with distinguished root $v_0 \in V$;
- (ii) $E \subseteq V \times V$ are directed edges which are assumed to contain no cycles;
- (iii) L is a vertex labeling from V to $\Sigma \cup OP \cup \{0, 1\}$;
- (iv) $\Sigma = \{x_1, \dots, x_m\}$ is the set of *input variables*;
- (v) $OP = \{\neg, \wedge, \vee, \oplus\}$ are the usual Boolean operators.

The *fan-in* (*fan-out*) of each $v \in V$ is the number of edges entering (departing from) v . For simplicity in our construction, we assume the maximum fan-in of any $v \in V$ is not more than 2. If $L(v) \in \Sigma \cup \{0, 1\}$, then v is an *input gate*, and we require that v have in degree 0. We assume no two input gates are labelled with the same input variable. C is a *tree* if all its gates have fan-out at most 1.

The depth d of circuit C is the length of the largest path in C . Since C is an acyclic digraph, C has a *topological ordering* $<$ which is a total ordering such that $\forall (u, v) \in E, v < u$. A topological ordering can be computed in linear sequential time on a unit cost RAM using a depth first search [41]. If C is a tree, then any postordering is a topological ordering. (See Figure 1 for an example of a Boolean circuit. Note that its gates are indexed in a topological order.)

For each $v \in V$, let C_v be the Boolean circuit identical to C except with root v . Given any input vector \vec{x} , let $C_v(\vec{x})$ be the Boolean function computed from gate v .

Let $C(\vec{x}) = C_{v_0}(\vec{x})$ be the Boolean function computed by the circuit C from root v_0 . Note that $C(\vec{x})$ can be sequentially evaluated in n steps using a traversal in reverse topological ordering of C . Alternately, $C(\vec{x})$ can be evaluated in parallel in time d using n processors, where d is the depth of C .

It will be frequently useful to modify the circuit C so that it computes a given Boolean function $f(\vec{y})$ at a gate $v \in V$ rather than $C_v(\vec{x})$. This *substitution operation* $C[v/f(\vec{y})]$ is formally defined as follows:

- (i) Let F be a Boolean circuit such that $F(\vec{y}) = f(\vec{y})$;
- (ii) Let $\tilde{C} = C[v/f(\vec{y})]$ be the Boolean circuit derived by (a) taking the union of C and F , (b) deleting all edges of C entering v , and (c) merging the root of F with v , so the label of v is now the label of the root of F .

Observe that $\tilde{C}_v(\vec{x}, \vec{y}) = f(\vec{y})$ as required. As a simple example, the substitution $\tilde{C} = C[v/b]$, where $b \in \{0, 1\}$, is the circuit derived from C by deleting all edges entering gate v and resetting the label of v to be b . If y is a variable, $C[v/y]$ is similarly constructed, except the label of v is set to y . See Figure 2 for a simple example of the substitution operation.

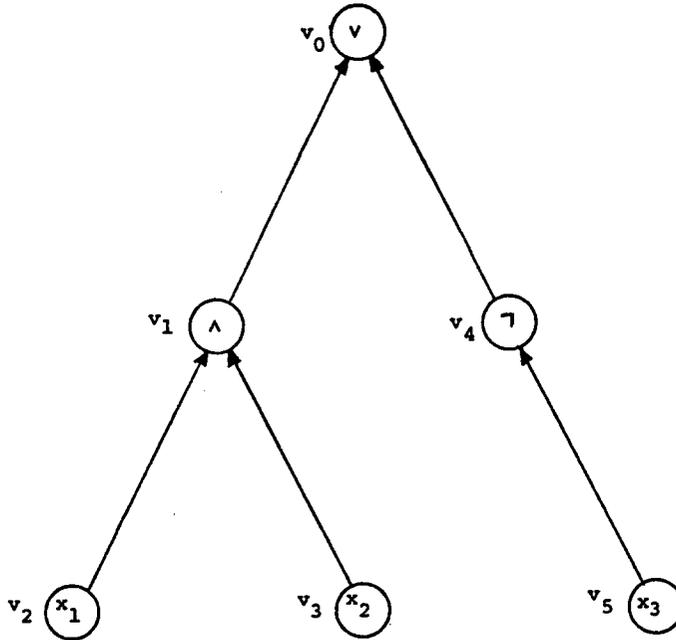


Figure 1. A Boolean circuit tree C computing $C(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee \neg x_3$.

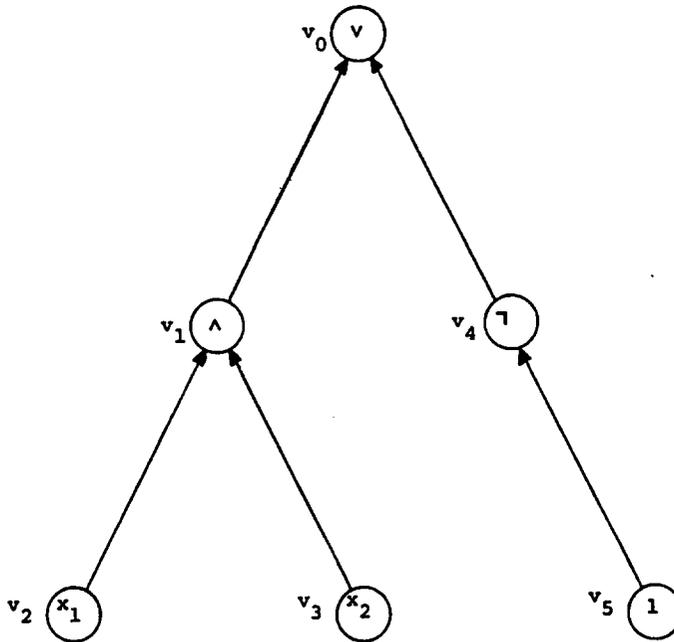


Figure 2. Boolean circuit $\tilde{C} = C[v_5/1]$ derived from the circuit C of Figure 1 by introduction of a S-A-1 fault at gate v_5 , so $\tilde{C}(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee \neg 1) = x_1 \wedge x_2$.

2.3. S-A Faults

Let C be a Boolean circuit. Fix some gate v and Boolean value $b \in \{0, 1\}$. Observe that by definition, $\tilde{C} = C[v/b]$ is a circuit identical to C except it has a S-A- b fault at gate v . This S-A- b fault at gate v is *detectable* on input \vec{x} , if $\tilde{C}(\vec{x}) \neq C(\vec{x})$.

For each $v \in V$, let z_v be a new indeterminate variable distinct from Σ . Observe that $\hat{C} = C[v/(z_v \oplus C_v)]$ is the circuit derived from C by substituting $z_v \oplus C_v$ in place of the subcircuit C_v . The definition of Boolean difference implies

$$\frac{d\hat{C}(\vec{x}, z_v)}{dz_v} = 1$$

if and only if $\hat{C}(\vec{x}, 0) \oplus \hat{C}(\vec{x}, 1) = 1$ if and only if $\hat{C}(\vec{x}, z_v)$ is sensitive to a change in z_v . But clearly $\hat{C}(\vec{x}, z_v)$ is sensitive to a change in z_v if and only if $C(\vec{x})$ is sensitive to a change in $C_v(\vec{x})$. Hence, we can let

$$\frac{dC(\vec{x})}{dC_v(\vec{x})} \text{ denote } \frac{d\hat{C}(\vec{x}, z_v)}{dz_v}.$$

This definition implies $\frac{dC(\vec{x})}{dC_v(\vec{x})} = 1$ if and only if $C(\vec{x})$ is sensitive to a change in $C_v(\vec{x})$. Hence,

$$(b \oplus C_v(\vec{x})) \wedge \frac{dC(\vec{x})}{dC_v(\vec{x})} = 1$$

if and only if $b \neq C_v(\vec{x})$ and $C(\vec{x})$ is sensitive to a change in $C_v(\vec{x})$ if and only if $\tilde{C}(\vec{x}) \neq C(\vec{x})$ where $\tilde{C} = C[v/b]$. Thus, we have

LEMMA 1. $(b \oplus C_v(\vec{x})) \wedge \frac{dC(\vec{x})}{dC_v(\vec{x})} = 1$ if and only if a S-A-b fault at gate v is detectable on input \vec{x} .

3. THE CIRCUIT CONSTRUCTIONS

3.1. A Circuit for all Boolean Differences

Fix a Boolean circuit tree $C = (V, v_0, E, L, \Sigma, OP)$. We construct a circuit $C' = (V', v'_0, E', L', \Sigma, OP)$ from C as follows. C' contains C as a subcircuit. The vertex set V' of C' contains the original vertex set V , a distinguished new vertex s , and a distinguished new vertex v' for each original vertex $v \in V$, plus some additional nondistinguished new vertices. All these new vertices are distinct from those in V . The vertex s has constant label $L(s) = 1$ and an edge $(s, v'_0) \in E'$ from s to v'_0 . The circuit construction now proceeds in topological order of C , using the rules R_1 – R_5 given in Figure 3. The left portion of each rule is a subcircuit of C , and the right portion of each rule is the corresponding new subcircuit of C' . See Figure 4 for an example of the construction using the rules R_1 – R_5 .

Note that the rules R_1 – R_5 if applied to a gate v with fan-in 1, yields a new gate v' with in degree 1 and label \oplus . In this case, the gate v' is to be collapsed into its unique predecessor. This operation is called a \oplus -gate collapse. (See Figure 5 for an example of a \oplus -gate collapse.) After applying \oplus -gate collapse wherever possible, the resulting circuit is denoted C' . (See Figure 6.)

It follows immediately from our construction that:

LEMMA 2. C' has $\leq 4n$ gates and depth $\leq 3d + 1$, where n, d are the number of gates and depth of C , respectively.

PROOF. The most costly rule in our construction is R_5 , where 3 new gates are introduced and not collapsed. The additional depth is 2 for each original gate. This rule can be applied at most d times along each path, and a total of at most n times. ■

NOTE. In the case $OP = \{\oplus, \neg, \wedge\}$, then, at most, 1 new gate is introduced and not collapsed and the additional depth is only 1 for each original gate. In this case, the number of gates of C' is $\leq 2n$, and the depth is $\leq 2n + 1$. (In practice, one can probably expect C' to have approximately $3n$ gates if C has a significant proportion of gates that are not v -gates.)

In Section 4, we prove:

THEOREM 1. For each gate $v \in V$,

$$C'_{v'}(\vec{x}) = \frac{dC(\vec{x})}{dC_v(\vec{x})}.$$

Thus each gate v' computes in C' the Boolean difference of $C(\vec{x})$ with respect to subcircuit $C_v(\vec{x})$. (See Figures 6 and 8.)

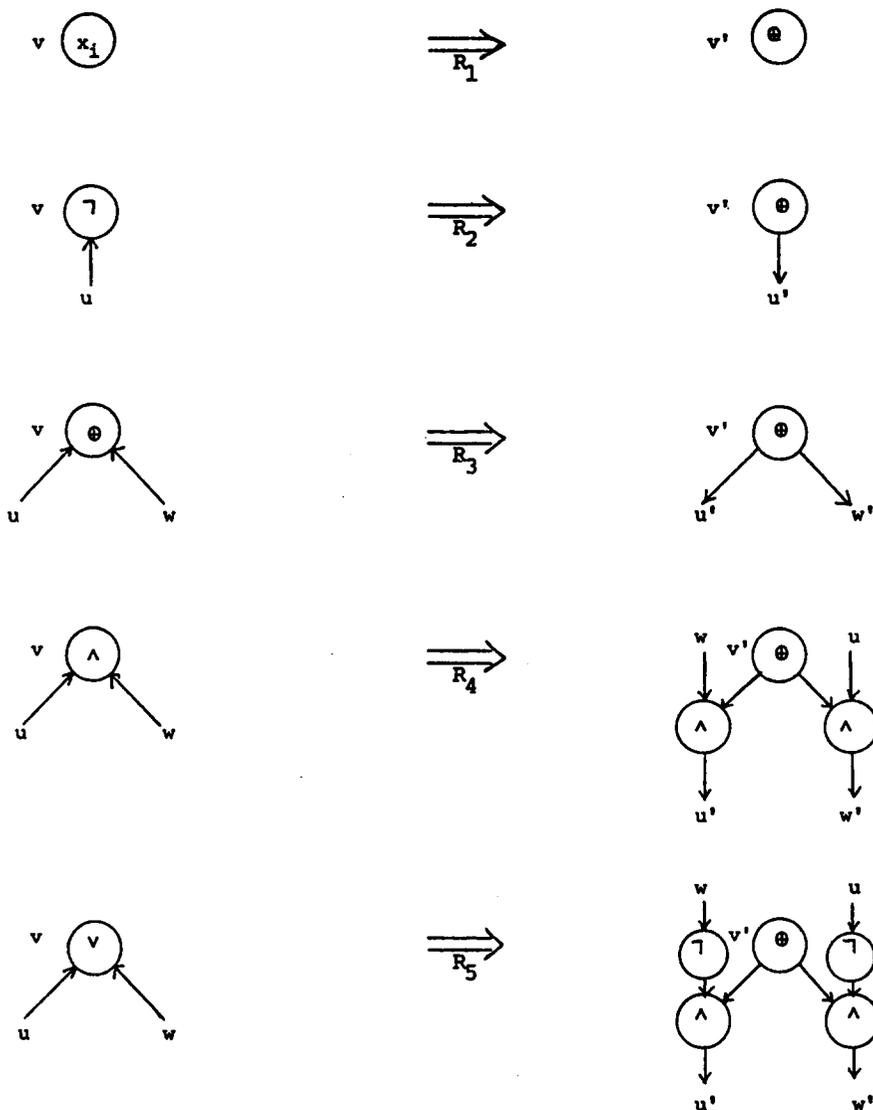


Figure 3. Rules for construction of C' from C .

3.2. The VLSI Area of C'

Let us assume a standard VLSI model such as described in [42]. A VLSI chip consists of a constant number of layers, where on each layer, the wires run along the edges of a square grid with unit length separation between grid points. Each grid point may contain a Boolean gate. Ullman [42] shows that any such VLSI chip with ℓ layers can be redesigned to have only 2 layers, with only a factor of ℓ^2 area increase.

LEMMA 3. C' can be constructed within VLSI area $\leq 9A$ using $\leq 2(\ell + 1)$ layers, if C has a VLSI chip with area A using ℓ layers.

PROOF. We begin with the given VLSI chip for C and then add an additional ℓ layers above the original ones. We also add an additional pair of horizontal and vertical grid lines between every previous pair of consecutive grid lines. In each application of the rules R_1 – R_5 , we place the new subcircuit associated with the distinguished new vertex v' on the ℓ new layers immediately above the subcircuit associated with the original vertex v , using the new grid lines to do the required wiring and for placement of the additional circuit. It is easy to verify this wiring can be done within the required area. ■

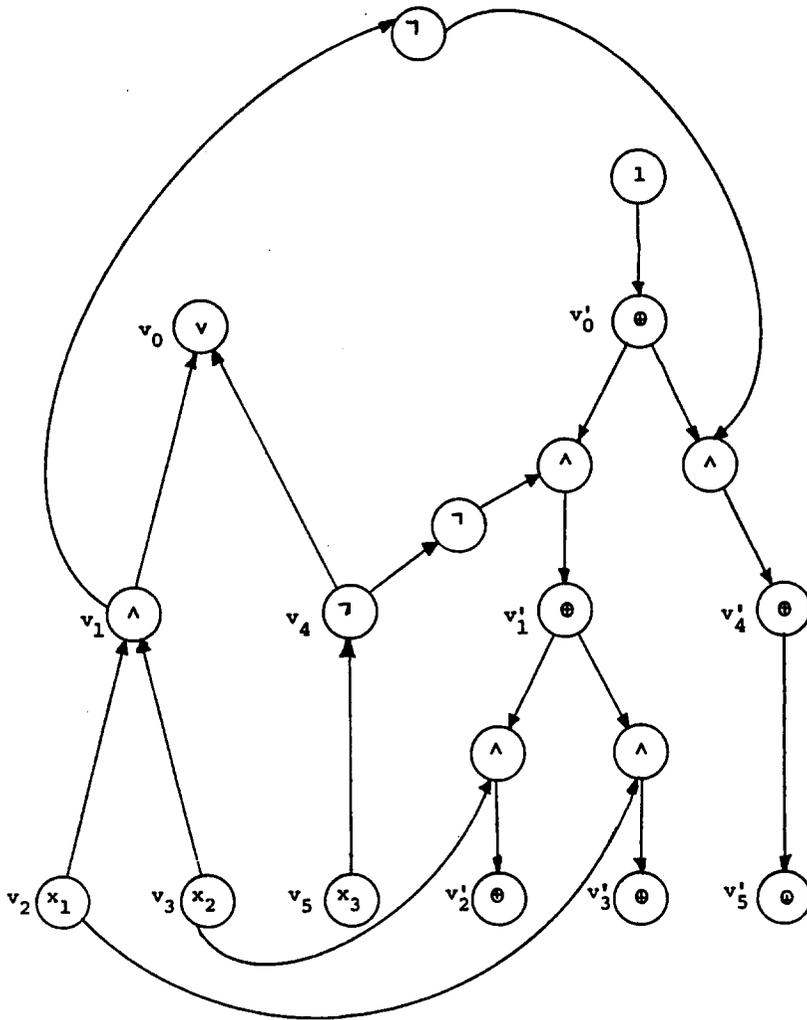


Figure 4. A circuit derived from circuit C of Figure 1 using rules R_1 - R_5 .

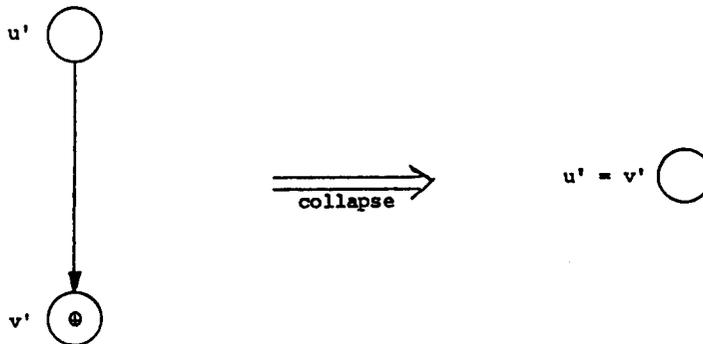


Figure 5. A collapse of gate v' , with fan-in 1 and label \oplus , into its predecessor u' .

NOTE. The actual area increase for C' will depend somewhat on the VLSI technology utilized. However, it is clear that the area increase is a small constant, for the standard technologies such as nMOS and cMOS.

3.3. Construction of Our Fault Simulation Circuit

Figure 7 gives a simple construction for building our fault simulation circuit $FS(C)$ from the Boolean differences circuit C' . Note that this construction is applied to each gate $v \in V$, and results in $3n$ additional gates and an additional depth of 2. By Theorem 1, the resulting circuit

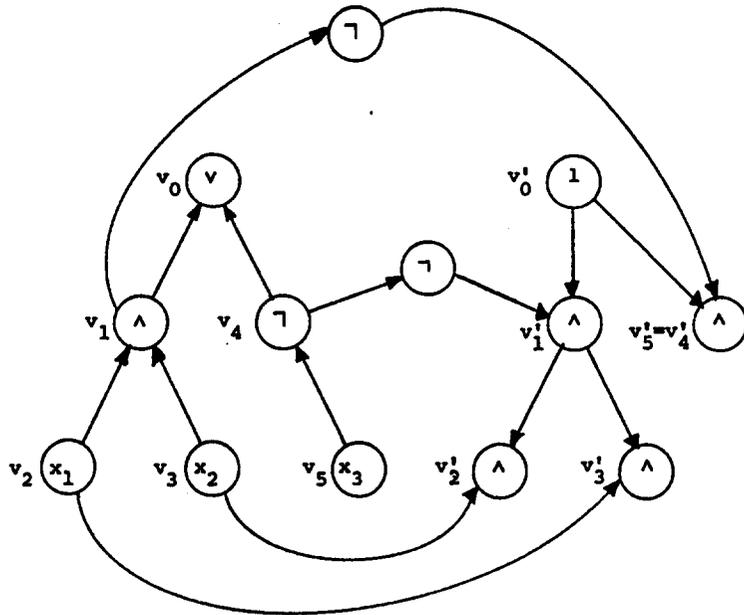


Figure 6. The circuit C' derived from circuit of Figure 4 after applying \oplus -gate collapses.

satisfies

$$FS(C)_{S-A-1(v)}(\vec{x}) = (\neg C'_v(\vec{x})) \oplus C'_{v'}(\vec{x}) = (\neg C_v(\vec{x})) \oplus \frac{dC(\vec{x})}{dC_v(\vec{x})},$$

and furthermore,

$$FS(C)_{S-A-0(v)}(\vec{x}) = C'_v(\vec{x}) \oplus C'_{v'}(\vec{x}) = C_v(\vec{x}) \oplus \frac{dC(\vec{x})}{dC_v(\vec{x})}.$$

The following are then implied by Lemmas 1 and 2.

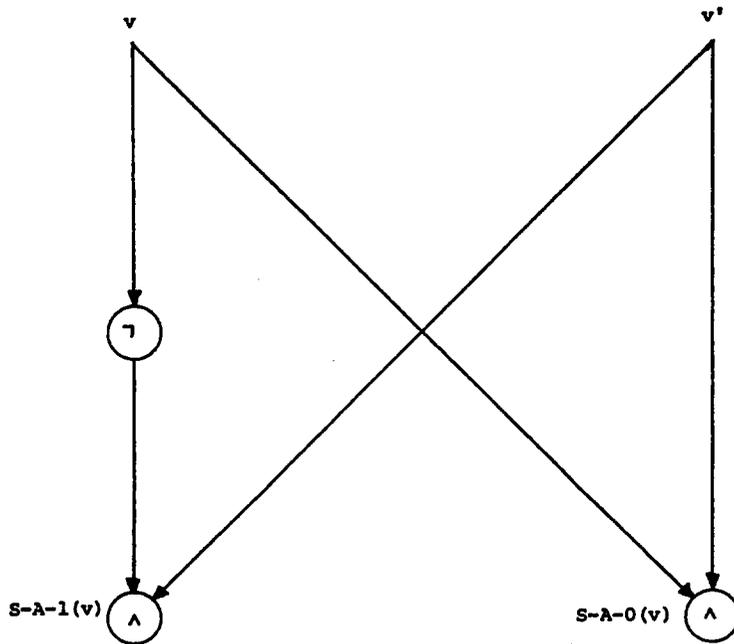


Figure 7. Construction of the fault simulation circuit $FS(C)$ from C' . The gate $S-A-b(v)$ indicates an $S-A-b$ fault at gate v , for $b \in \{0, 1\}$.

vertex v	computed value $C_v(\vec{x})$	Boolean difference $C'_{v'}(\vec{x}) = \frac{dC(\vec{x})}{dC_v(\vec{x})}$	S-A-1 fault S-A-1 (v)	S-A-0 fault S-A-0 (v)
v_0	$(x_1 \wedge x_2) \vee \neg x_3$	1	0	$(x_1 \wedge x_2) \vee \neg x_3$
v_1	$x_1 \wedge x_2$	x_3	$(\neg(x_1 \wedge x_2)) \wedge x_3$	$x_1 \wedge x_2 \wedge x_3$
v_2	x_1	$x_2 \wedge x_3$	$(\neg x_1) \wedge x_2 \wedge x_3$	$x_1 \wedge x_2 \wedge x_3$
v_3	x_2	$x_1 \wedge x_3$	$x_1 \wedge (\neg x_3) \wedge x_3$	$x_1 \wedge x_2 \wedge x_3$
v_4	$\neg x_3$	$\neg(x_1 \wedge x_3)$	$(\neg(x_1 \wedge x_2)) \wedge x_3$	$(\neg(x_1 \wedge x_2)) \wedge \neg x_3$
v_5	x_3	$\neg(x_1 \wedge x_3)$	$(\neg(x_1 \wedge x_2)) \wedge \neg x_3$	$(\neg(x_1 \wedge x_2)) \wedge x_3$

Figure 8. Boolean differences and S-A- b faults of circuit C given in Figure 1.

THEOREM 2. Suppose we are given a Boolean circuit tree C with n gates, a depth of d , and VLSI area A using ℓ layers. Then using the rules in Figures 3, 4 and 7, we can construct from C in linear time a Boolean circuit $FS(C)$ consisting of $\leq 7n$ gates and depth $\leq 3(d+1)$, using VLSI area $\leq 9A$ and $2(\ell+1)$ layers. For each $b \in \{0, 1\}$, gate $v \in V$ and input \vec{x} , $FS(C)_{S-A-b(v)}(\vec{x}) = 1$ (i.e., gate S-A- $b(v)$ is evaluated to 1 by $FS(C)$ on input \vec{x}) if and only if a S-A- b fault in C at gate v is detectable on input \vec{x} .

3.4. Fault Simulation of VLSI Circuits with Feedback

We briefly describe here how our techniques for fault simulation can be extended to apply to some VLSI circuits S with feedback lines (i.e., S may have cycles). We note that S is input *oblivious* in the sense that the sequence of logical operations (but not necessarily the values computed by these operations) of its gates are independent of its given input \vec{x} . Let V be the set of gates of S . We assume the computation sequence of length d of S on input \vec{x} can be modeled by a Boolean circuit tree $C(S)$ of depth d such that the gates of $C(S)$ are partitioned into disjoint levels V_0, V_1, \dots, V_d . Intuitively, each level V_t corresponds to the logical operations executed at time t . More precisely, for each $t = 0, 1, \dots, d$, we assume a 1-1 mapping r_t from V to level V_t such that $\forall v \in V$, $C_{r_t(v)}(\vec{x})$ is the Boolean value computed at gate v by S at time t given an input vector \vec{x} . Thus, $r_t(v)$ evaluates to just the value computed by S at gate v of time t . We require that each edge entering a gate of level V_t depart only from a gate of level V_{t-1} , and so the value of a gate at level V_t depends only on the values computed by gates at level V_{t-1} . We shall model faults in S simply by S-A- b faults in $C(S)$. Suppose S has n gates. Then $C(S)$ has $n(d+1)$ gates. A direct consequence of Theorem 2 is:

COROLLARY 1. All S-A faults for S on input \vec{x} can be determined by an acyclic Boolean circuit $FS(C(S))$ of $\leq 7n(d+1)$ gates and depth $\leq 3(d+1)$, and hence in $\leq 7n(d+1)$ sequential time.

Note that the fault simulation circuit $FS(C(S))$ utilized just the values of gates at level V_t in reverse order $t = d, d-1, \dots, 0$. We can avoid construction of $FS(C(S))$ and instead construct a VLSI circuit with feedback lines which simulatively determines S-A faults of level V_t . To get the required values of the gates of level V_t we can simply reexecute S on the same input \vec{x} . Thus, (if we wish to avoid actually constructing $FS(C(S))$), we require by this method a total of d executions of S , each requiring parallel time d .

Assuming S has VLSI area A using ℓ layers, we get by this construction:

COROLLARY 2. All S-A faults for S can be determined by a VLSI circuit (with feedback lines) of $O(n)$ gates, $O(A)$ area and $O(\ell)$ layers, taking $O(d^2)$ parallel time.

We consider S to be *reversible* if after any execution of d parallel steps, a special *reverse* switch can be set so that on consecutive steps $t = d+1, d+2, \dots, 2d$, the gates of S have the same value at time t as they previously did at time $|t-2d-1|$. As mentioned in the introduction, many switching, sorting and convolution circuits are both oblivious and reversible in this sense. Observe that if S is reversible, then S can compute the values of levels V_d, V_{d-1}, \dots, V_0 in reverse order using d parallel steps.

COROLLARY 3. If S is reversible, then all S-A faults can be determined by a VLSI circuit (with feedback lines) of $O(n)$ gates, $O(A)$ area, $O(\ell)$ layers, and $O(d)$ time.

4. PROOF OF THE CORRECTNESS OF OUR BOOLEAN DIFFERENCES CIRCUIT

In this section we prove Theorem 1.

4.1. Chain Rules for Boolean Differences

We will use the following fundamental rules for manipulating Boolean differences. It can be easily proved from the definition of Boolean difference.

PROPOSITION 1. (*The Chain Rule*) If $f(\vec{x}, \vec{y}) = f_1(f_2(\vec{x}), \vec{y})$, then

$$\frac{df(\vec{x}, \vec{y})}{dx_i} = \frac{df_1(f_2(\vec{x}), \vec{y})}{df_2} \wedge \frac{df_2(\vec{x})}{dx_i}.$$

The following propositions follow from the Chain Rule by use of elementary Boolean algebra identities and the definition of the Boolean difference.

PROPOSITION 2.

$$\frac{d(\neg f(\vec{x}))}{dx_i} = \frac{df(\vec{x})}{dx_i}.$$

PROPOSITION 3.

$$\frac{d(f_1(\vec{x}) \oplus f_2(\vec{x}))}{dx_i} = \frac{df_1(\vec{x})}{dx_i} \oplus \frac{df_2(\vec{x})}{dx_i}.$$

PROPOSITION 4.

$$\frac{d(f_1(\vec{x}) \wedge f_2(\vec{x}))}{dx_i} = f_1(\vec{x}) \wedge \frac{df_2(\vec{x})}{dx_i} \oplus f_2(\vec{x}) \wedge \frac{df_1(\vec{x})}{dx_i} \oplus \frac{df_1(\vec{x})}{dx_i} \wedge \frac{df_2(\vec{x})}{dx_i}.$$

PROPOSITION 5.

$$\frac{d(f_1(\vec{x}) \vee f_2(\vec{x}))}{dx_i} = (\neg f_1(\vec{x})) \wedge \frac{df_2(\vec{x})}{dx_i} \oplus (\neg f_2(\vec{x})) \wedge \frac{df_1(\vec{x})}{dx_i} \oplus \frac{df_1(\vec{x})}{dx_i} \wedge \frac{df_2(\vec{x})}{dx_i}.$$

4.2. An Inductive Proof of Theorem 1

Fix a topological ordering $<$ of the gates of circuit tree C . The proof of Theorem 1 will proceed by induction in this topological order $<$, beginning from the root gate v_0 .

For the basis of the induction, we observe that $C_{v_0}(\vec{x}) = C(\vec{x})$, so $\frac{dC(\vec{x})}{dC_{v_0}(\vec{x})} = 1$. Furthermore, the construction gives $C'_{v_0}(\vec{x}) = 1$, so we have

$$C'_{v_0}(\vec{x}) = 1 = \frac{dC(\vec{x})}{dC_{v_0}(\vec{x})},$$

as required.

Now fix some gate $u \in V - \{v_0\}$. Let us make the inductive assumption that

$$C'_{v'}(\vec{x}) = \frac{dC(\vec{x})}{dC_{v'}(\vec{x})},$$

for all gates $v \in V$ strictly preceding u in the given topological order of C . Let $(u, v) \in E$ be the unique edge of C departing from gate u . By the induction hypothesis,

$$C'_{v'}(\vec{x}) = \frac{dC(\vec{x})}{dC_v(\vec{x})}.$$

Define

$$\lambda_v(\vec{x}) = \begin{cases} 1, & \text{if } L(v) \in \{\neg, \oplus\}, \\ C_w(\vec{x}), & \text{if } L(v) = \vee, \\ \neg C_w(\vec{x}), & \text{if } L(v) = \wedge, \end{cases}$$

where $(u, v), (w, v) \in E$ are the edges entering gate v in C .

By examining the rules R_1 - R_5 , we observe that we have defined

$$C'_{u'}(\vec{x}) = C'_{v'}(\vec{x}) \wedge \lambda_v(\vec{x}).$$

We must now prove that

$$C'_{u'}(\vec{x}) = \frac{dC(\vec{x})}{dC_u(\vec{x})}.$$

Let us define

$$H = C_v[u/(z_u \oplus C_u)].$$

Recall by definition

$$\frac{dC_v(\vec{x})}{dC_u(\vec{x})} = \frac{dH(\vec{x}, z_u)}{dz_u} = H(\vec{x}, 0) \oplus H(\vec{x}, 1)$$

gives the conditions where $C_v(\vec{x})$ (which is the subcircuit of C rooted at gate v) is sensitive to the value transmitted from gate u .

LEMMA 4.

$$\lambda_v(\vec{x}) = \frac{dC_v(\vec{x})}{dC_u(\vec{x})}.$$

PROOF. This Lemma can be proved either by application of the Propositions 1-5, or directly by definition of the Boolean difference. We give here a complete direct proof.

If $L(v) = \neg$, then $C_v(\vec{x}) = \neg C_u(\vec{x})$ and $C_v(\vec{x}, z_u) = \neg(z_u \oplus C_u(\vec{x}))$ so

$$H(\vec{x}, 0) \oplus H(\vec{x}, 1) = (\neg C_u(\vec{x})) \oplus C_u(\vec{x}) = 1 = \lambda_v(\vec{x}),$$

in this case.

If $L(v) = \oplus$, then $C_v(\vec{x}) = C_u(\vec{x}) \oplus C_w(\vec{x})$ and $H(\vec{x}, z_u) = (z_u \oplus C_u(\vec{x})) \oplus C_w(\vec{x})$ so

$$\begin{aligned} H(\vec{x}, 0) \oplus H(\vec{x}, 1) &= (C_u(\vec{x}) \oplus C_w(\vec{x})) \oplus ((\neg C_u(\vec{x})) \oplus C_u(\vec{x})), \\ &= 1 = \lambda_v(\vec{x}), \end{aligned}$$

in this case.

If $L(v) = \wedge$, then $C_v(\vec{x}) = C_u(\vec{x}) \wedge C_w(\vec{x})$ and $H(\vec{x}, z_u) = (z_u \oplus C_u(\vec{x})) \wedge C_w(\vec{x})$ so

$$\begin{aligned} H(\vec{x}, 0) \oplus H(\vec{x}, 1) &= (C_u(\vec{x}) \wedge C_w(\vec{x})) \oplus ((\neg C_u(\vec{x})) \wedge C_w(\vec{x})), \\ &= C_w(\vec{x}) = \lambda_v(\vec{x}), \end{aligned}$$

in this case.

If $L(v) = \vee$, then $C_v(\vec{x}) = C_u(\vec{x}) \vee C_w(\vec{x})$ and $H(\vec{x}, z_u) = (z_u \oplus C_u(\vec{x})) \vee C_w(\vec{x})$ so

$$\begin{aligned} H(\vec{x}, 0) \oplus H(\vec{x}, 1) &= (C_u(\vec{x}) \vee C_w(\vec{x})) \oplus ((\neg C_u(\vec{x})) \vee C_w(\vec{x})), \\ &= \neg C_w(\vec{x}) = \lambda_v(\vec{x}), \end{aligned}$$

in this case. Thus, we have shown in each case

$$\frac{dC_v(\vec{x})}{dC_u(\vec{x})} = H(\vec{x}, 0) \oplus H(\vec{x}, 1) = \lambda_v(\vec{x}).$$

■

LEMMA 5.

$$\frac{dC(\vec{x})}{dC_u(\vec{x})} = C'_{v'}(\vec{x}) \wedge \lambda_v(\vec{x}).$$

PROOF. By the Chain Rule,

$$\frac{dC(\vec{x})}{dC_u(\vec{x})} = \frac{dC(\vec{x})}{dC_v(\vec{x})} \wedge \frac{dC_v(\vec{x})}{dC_u(\vec{x})} = C'_{v'}(\vec{x}) \wedge \lambda_v(\vec{x}),$$

since by the induction hypothesis for $v < u$,

$$C'_{v'}(\vec{x}) = \frac{dC(\vec{x})}{dC_v(\vec{x})},$$

and by Lemma 4,

$$\frac{dC_v(\vec{x})}{dC_u(\vec{x})} = \lambda_v(\vec{x}).$$

From Lemma 5, we immediately get

$$\frac{dC(\vec{x})}{dC_u(\vec{x})} = C'_{v'}(\vec{x}) \wedge \lambda_v(\vec{x}) = C'_{u'}(\vec{x}),$$

proving the induction step required for Theorem 1.

5. BOOLEAN DIFFERENCES OF CIRCUITS WITH FAN-OUT > 1

In this section, we describe how to extend our techniques to computing all the Boolean differences for a circuit C with fan-out > 1. Without loss of generality, we can assume C has fan-out ≤ 2 (since a transformation to fan-out ≤ 2 requires only a size increase linear in the number of edges of C). We will fix $C = (V, v_0, E, L, \Sigma, OP)$ to be such a Boolean circuit.

5.1. Edge Sensitivity

Let \mathcal{E} be the circuit derived from C by replacing each edge $(u, v) \in E$ of C by a chain of two new edges (u, uv) , (uv, v) where uv is a new \oplus gate with fan-in 1 and fan-out 1 (see Figures 9 and 10). Note that \mathcal{E} is logically equivalent to C since $\mathcal{E}_{uv}(\vec{x}) = \mathcal{E}_u(\vec{x}) = C_u(\vec{x})$. Hence, by definition,

$$\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv}(\vec{x})} = \frac{d\mathcal{E}[uv/(z_{uv} \oplus \mathcal{E}_{uv})](\vec{x}, z_{uv})}{dz_{uv}}$$

gives the conditions where $C(\vec{x})$ is sensitive to the value computed across edge (u, v) . (Observe that $\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv}(\vec{x})}$ may differ from $\frac{dC(\vec{x})}{dC_u(\vec{x})}$ if u has fan-out > 1 in C .)

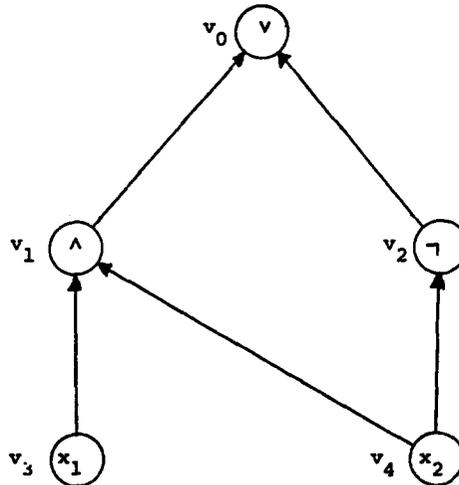


Figure 9. A Boolean circuit C computing $C(\vec{x}) = (x_1 \wedge v_2) \vee \neg x_2$. Note that gate v_4 has fan-out 2.

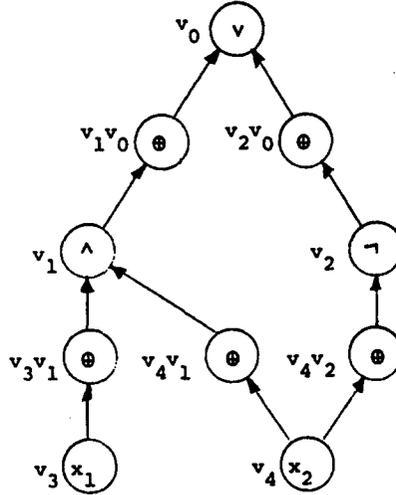


Figure 10. The circuit \mathcal{E} derived from C of Figure 9.

5.2. A Boolean Difference Circuit Construction for Fan-Out 2

We will construct our Boolean difference circuit C' as follows. C' will initially contain the circuit \mathcal{E} . We then apply in topological order of \mathcal{E} the rules R_1 – R_5 of Figure 3 previously described in Section 3. We also apply in this order a new rule R_6 , as given in Figure 11, to each gate u with fan-out 2. Let $(u, v_1), (u, v_2)$ be the edges departing v . Before application of rule R_6 to gate u , since $v_1 < u$, we have already constructed a subcircuit of C' which computes

$$C'_{(uv_1)'}(\vec{x}) = \frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_1}(\vec{x})}$$

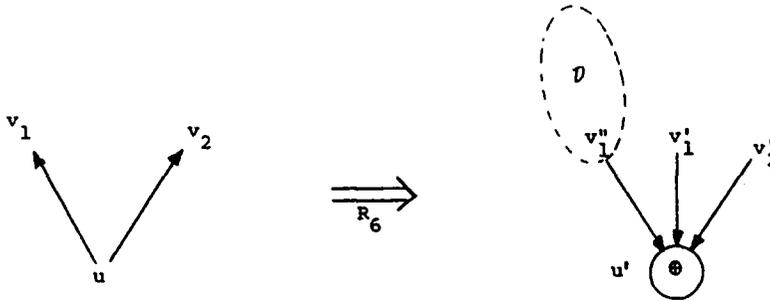


Figure 11. A rule R_6 for a gate u with departing edges $(u, v_1), (u, v_2) \in E$. Here v_1'' is the root of a new subcircuit \mathcal{D} computing $\mathcal{D}(\vec{x}) = \frac{d}{d\mathcal{E}_{uv_2}} \left[\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_1}(\vec{x})} \right]$.

The application of rule R_6 to gate u requires that we recursively construct a subcircuit \mathcal{D} that computes

$$\mathcal{D}(\vec{x}) = \frac{dC'_{(uv_1)'}(\vec{x})}{d\mathcal{E}_{uv_2}} = \frac{d}{d\mathcal{E}_{uv_2}} \left(\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_1}(\vec{x})} \right)$$

To complete the construction of C' , we apply the \oplus -gate collapsing rule given in Figure 5 (see Figure 12).

In the case there are many gates of fan-out 2, this construction is obviously finite, but may be computationally inefficient (at least in comparison with our linear time construction in the case of fan-out 1), but it nevertheless is certainly much more efficient than the direct computation of Boolean differences by application of chain rules, which was the only previously known method.

By examining the rules R_1 – R_6 , we observe that

$$C'_{u'}(\vec{x}) = \mathcal{D}(\vec{x}) \oplus (C'_{v_1}(\vec{x}) \wedge \lambda_{v_1}(\vec{x})) \oplus (C'_{v_2}(\vec{x}) \wedge \lambda_{v_2}(\vec{x})),$$

where

$$\mathcal{D}(\vec{x}) = \frac{d}{d\mathcal{E}_{uv_2}(\vec{x})} \left(\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_1}(\vec{x})} \right).$$

We must now prove that

$$C'_{u'}(\vec{x}) = \frac{dC(\vec{x})}{dC_u(\vec{x})}.$$

For $i = 1, 2$ let us define

$$\mathcal{E}^{(i)} = \mathcal{E}_{v_i}[uv_i/(z_{uv_i} \oplus \mathcal{E}_{uv_i})].$$

Recall by definition

$$\frac{d\mathcal{E}_{v_i}(\vec{x})}{d\mathcal{E}_{uv_i}(\vec{x})} = \frac{d\mathcal{E}^{(i)}(\vec{x}, z_{uv_i})}{dz_{uv_i}} = \mathcal{E}^{(i)}(\vec{x}, 0) \oplus \mathcal{E}^{(i)}(\vec{x}, 1)$$

gives the conditions where $C_{v_i}(\vec{x})$ (which is the subcircuit of C rooted at gate v_i) is sensitive to the value transmitted across the i^{th} edge (u, v_i) entering gate v_i .

LEMMA 6. For each $i = 1, 2$,

$$\lambda_{v_i} = \frac{d\mathcal{E}_{v_i}(\vec{x})}{d\mathcal{E}_{uv_i}(\vec{x})}.$$

PROOF. If $L(v_i) = \neg$, then $C_{v_i}(\vec{x}) = \neg C_u(\vec{x})$ and $\mathcal{E}^{(i)}(\vec{x}, z_{uv_i}) = \neg(z_{uv_i} \oplus C_u(\vec{x}))$ so

$$\mathcal{E}^{(i)}(\vec{x}, 0) \oplus \mathcal{E}^{(i)}(\vec{x}, 1) = (\neg C_u(\vec{x})) \oplus C_u(\vec{x}) = 1 = \lambda_{v_i}(\vec{x}),$$

in this case.

If $L(v_i) = \oplus$, then $C_{v_i}(\vec{x}) = C_u(\vec{x}) \oplus C_w(\vec{x})$ and $\mathcal{E}^{(i)}(\vec{x}, z_{uv_i}) = (z_{uv_i} \oplus C_u(\vec{x})) \oplus C_w(\vec{x})$ so

$$\begin{aligned} \mathcal{E}^{(i)}(\vec{x}, 0) \oplus \mathcal{E}^{(i)}(\vec{x}, 1) &= (C_u(\vec{x}) \oplus C_w(\vec{x})) \oplus ((\neg C_u(\vec{x})) \oplus C_u(\vec{x})), \\ &= 1 = \lambda_{v_i}(\vec{x}), \end{aligned}$$

in this case.

If $L(v_i) = \wedge$, $C_{v_i}(\vec{x}) = C_u(\vec{x}) \wedge C_w(\vec{x})$ and $\mathcal{E}^{(i)}(\vec{x}, z_{uv_i}) = (z_{uv_i} \oplus C_u(\vec{x})) \wedge C_w(\vec{x})$ so

$$\begin{aligned} \mathcal{E}^{(i)}(\vec{x}, 0) \oplus \mathcal{E}^{(i)}(\vec{x}, 1) &= (C_u(\vec{x}) \wedge C_w(\vec{x})) \oplus ((\neg C_u(\vec{x})) \wedge C_w(\vec{x})), \\ &= C_w(\vec{x}) = \lambda_{v_i}(\vec{x}), \end{aligned}$$

in this case.

If $L(v_i) = \vee$, then $C_{v_i}(\vec{x}) = C_u(\vec{x}) \vee C_w(\vec{x})$ and $\mathcal{E}^{(i)}(\vec{x}, z_{uv_i}) = (z_{uv_i} \oplus C_u(\vec{x})) \vee C_w(\vec{x})$ so

$$\begin{aligned} \mathcal{E}^{(i)}(\vec{x}, 0) \oplus \mathcal{E}^{(i)}(\vec{x}, 1) &= (C_u(\vec{x}) \vee C_w(\vec{x})) \oplus ((\neg C_u(\vec{x})) \vee C_w(\vec{x})), \\ &= \neg C_w(\vec{x}) = \lambda_{v_i}(\vec{x}), \end{aligned}$$

in this case. Thus, we have shown in each case

$$\frac{d\mathcal{E}_{v_i}(\vec{x})}{d\mathcal{E}_{uv_i}(\vec{x})} = \mathcal{E}^{(i)}(\vec{x}, 0) \oplus \mathcal{E}^{(i)}(\vec{x}, 1) = \lambda_{v_i}(\vec{x}). \quad \blacksquare$$

We will use a generalized chain rule for Boolean differences:

PROPOSITION 6. *If $f(\vec{x}) = f_0(f_1(\vec{x}), f_2(\vec{x}))$, then*

$$\frac{df(\vec{x})}{dx_i} = \left[\frac{df_0(f_1(\vec{x}), f_2(\vec{x}))}{df_1} \wedge \frac{df_1(\vec{x})}{dx_i} \right] \oplus \left[\frac{df_0(f_1(\vec{x}), f_2(\vec{x}))}{df_2} \wedge \frac{df_2(\vec{x})}{dx_i} \right] \\ \oplus \left[\left(\frac{d}{df_2} \left(\frac{df_0(f_1(\vec{x}), f_2(\vec{x}))}{df_1} \right) \right) \wedge \frac{df_1}{dx_i} \wedge \frac{df_2}{dx_i} \right].$$

Proposition 6 implies

$$\frac{dC(\vec{x})}{dC_u(\vec{x})} = \frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_1}(\vec{x})} \oplus \frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_2}(\vec{x})} \oplus \frac{d}{d\mathcal{E}_{uv_2}} \left(\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_1}(\vec{x})} \right).$$

LEMMA 7.

$$\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_i}(\vec{x})} = C'_{v'_i}(\vec{x}) \wedge \lambda_{v_i}(\vec{x}), \quad \text{for } i = 1, 2.$$

PROOF. By the Chain Rule and Lemma 6,

$$\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_i}(\vec{x})} = \frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{v_i}(\vec{x})} \wedge \frac{d\mathcal{E}_{v_i}(\vec{x})}{d\mathcal{E}_{uv_i}(\vec{x})} = C'_{v'_i}(\vec{x}) \wedge \lambda_{v_i}(\vec{x}),$$

since by the induction hypothesis for $v_i < u$,

$$C'_{v'_i}(\vec{x}) = \frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{v_i}(\vec{x})}.$$

Applying Lemma 7, we immediately get

$$\frac{dC(\vec{x})}{dC_u(\vec{x})} = (C'_{v'_1}(\vec{x}) \wedge \lambda_{v_1}(\vec{x})) \oplus (C'_{v'_2}(\vec{x}) \wedge \lambda_{v_2}(\vec{x})) \oplus \frac{d}{d\mathcal{E}_{uv_2}} \left(\frac{d\mathcal{E}(\vec{x})}{d\mathcal{E}_{uv_1}(\vec{x})} \right), \\ = C'_{u'}(\vec{x}),$$

proving the induction step of Theorem 3. ■

REFERENCES

1. H. Fujiwara and S. Toida, The complexity of fault detection problems for combinational logic circuits, *IEEE Trans. Comput.* C-31 (6), 555-560 (1982).
2. R.D. Eldred, Test routines based on symbolic logic statements, *J. Ass. Comput. Mach.* 6 (1), 33-36 (1959).
3. D.B. Armstrong, On finding a nearly minimal set of fault detection tests for combinatorial nets, *IEEE Trans. Electron. Comput.* EC-15 (2), 63-73 (1966).
4. H.Y. Chang, E.G. Manning and G. Metzger, *Fault Diagnosis of Digital Systems*, Wiley Interscience, New York, (1970).
5. E.B. Eichelberger, Method of level sensitive testing a functional logic system, U.S. Patent 3761695, Sept. 25, 1973.
6. J.P. Roth, W.G. Bouricius and P.R. Schneider, Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits, *IEEE Trans. Electron. Comput.* EC-16 (10), 567-580 (1967).
7. H. Fujiwara and T.S. Shimono, On the acceleration of test generation algorithms, *IEEE Trans. Comput.* C-32 (12), 1137-1143 (1983).
8. A.V. Goldberg and K.J. Lieberherr, Deterministic versus random pattern testability, Design Automation Comp. Sci. Lab., G.T.E. Labs., Waltham, MA (1985).
9. H.D. Schnurmann, E. Lindbloom and R.G. Carpenter, The weighted random test pattern generator, *IEEE Trans. Comput.* C-24, 695-700 (1975).
10. J. Savin and P.H. Bondell, On random pattern test length, *1983 International Test Conference*, pp. 95-106, (1983).
11. J. Savin, G.S. Ditlow and P.H. Bondell, Random pattern testability, *IEEE Trans. Comput.* C-33 (1), 79-90 (1984).

12. K.J. Lieberherr, Parameterized random testing, *Design Automation Comp. Sci. Lab.*, G.T.E. Labs., Waltham, MA, (1983).
13. G. Markowsky, Bounding signal probabilities in combinational circuits, Technical Report, Dept. Comp. Sci., Univ. Maine, at Orono, 1984.
14. V.J. Losq, Efficiency of random compact testing, *IEEE Trans. Comput. C-27* (6), 516-525 (1978).
15. S.W. Golomb, *Shift Register Sequences*, Holden-Day, San Francisco, CA, (1967).
16. Z. Barzilai, D. Coppersmith and A.L. Rosenberg, Exhaustion generation of bit patterns with applications to VLSI self-testing, *IEEE Trans. Comput. C-32* (2), 190-194 (1983).
17. D.T. Tang, Logic test pattern generation using linear cycles, *IEEE Trans. Comput. C-33* (9), 845-850 (1984).
18. J.L. Carter, The theory of signature testing for VLSI, IBM Watson Research Center, Yorktown Heights, NY, (1982).
19. W.C. Carter, Signature testing with guaranteed bounds for fault coverage, *IEEE Test Comp.*, 75-82 (1982).
20. J.F. Poage, Derivation of optimum tests to detect faults in combinational circuits, In *Mathematical Theory of Automation*, Polytechnic Press, New York, (1963).
21. D.B. Armstrong, A deductive method for simulating faults in logic circuits, *IEEE Trans. Comput. C-21* (5), 464-471 (1972).
22. J.P. Roth, Diagnosis of automata failures: A calculus and a method, *IBM J. Res. Develop.* 10, 278-281 (1966).
23. E. Manning and H.Y. Chang, Functional technique for efficient digital fault simulation, *IEEE Int. Conv. Digest* 195 (1968).
24. H.Y.P. Chang, Comparison of parallel and deductive fault simulation, *IEEE Trans. Comput. C-23*, 1132-1138 (1974).
25. M.A. Breuer, Functional partitioning and simulation of digital circuits, *IEEE Trans. Comput. C-19*, 1038-1046 (1970).
26. A.D. Friedman and P.R. Menon, *Fault Detection in Digital Circuits*, Prentice-Hall, Englewood Cliffs, NJ, (1971).
27. E.G. Ulrich and T. Baker, Concurrent simulation of nearly identical digital networks, *Computer* 7 (4), 39-44 (1974).
28. E.I. Muehldorf and A.D. Savkar, LSI logic testing—An overview, *IEEETC C-30*, 1-17 (1981).
29. S.K. Jain and V.D. Agrawal, Stafan: An alternative to fault simulation, *IEEE 21st Design Automation Conf.*, pp. 18-22, (1984).
30. M. Abramovici, P.R. Menon and D.T. Miller, Critical path tracing: An alternative to fault simulation, *IEEE Design and Test*, 83-93, (February 1984).
31. A. Brown and H.W. Young, Algebraic logic network analysis, IBM Corp., Poughkeepsie, NY (NASA Contract NAS 12-689), IBM TR 0018 74, (1969).
32. A. Thayse and M. Davis, Boolean differential calculus and its application to switching theory, *IEEE Trans. Comput. C-22*, 409-420 (1973).
33. A. Thayse, *Boolean Calculus of Differences*, Monograph, Springer, Berlin, (1981).
34. E.I. Muehldorf, Theory of a switching calculus for the diagnosis of binary logic, *Elektron. Rechenanlagen* 15, 215-222 (1973).
35. E.F. Sellers, M.Y. Hsiao and L.W. Bearnson, Analyzing errors with the Boolean difference, *IEEE Trans. Comput. C-17*, 676-683 (1968).
36. C.T. Ku and G.M. Masson, The Boolean difference and multiple fault analysis, *IEEE Trans. Comput. C-24*, 691-695 (1975).
37. W. Baur and V. Strassen, The complexity of partial derivations, *Theoretical Computer Science* 22, 317-330 (1983).
38. V.M. Jerrum, personal communication, (1983).
39. K.J. Lieberherr, Standard hardware description language, *Design Automation Comp. Sci. Lab.*, G.T.E. Labs., Waltham, MA, (1983).
40. S.M. German and K.J. Lieberherr, Zues: A language for expressing algorithms in hardware, *Design Automation Comp. Sci. Lab.*, G.T.E. Labs., Waltham, MA, (1984).
41. A.V. Aho, J.E. Hopcroft and J.D. Ullman, *Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, (1974).
42. J.D. Ullman, *Computational Aspects of VLSI*, Chapter 1, Computer Science Press, Rockville, MD, (1984).