Design Verification via Simulation and Automatic Test Pattern Generation[1]

Hussain Al-Asaad and John P. Hayes

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122

Email: {halasaad, jhayes}@eecs.umich.edu

Abstract

We present a simulation-based method for combinational design verification that aims at complete coverage of specified design errors using conventional ATPG tools. The error models used in prior research are examined and reduced to four types: gate substitution errors (GSEs), gate count errors (GCEs), input count errors (ICEs), and wrong input errors (WIEs). Conditions are derived for a gate to be completely testable for GSEs; These conditions lead to small test sets for GSEs. Near-minimal test sets are also derived for GCEs. We analyze redundancy in design errors and relate this to single stuck-line (SSL) redundancy. We show how to map all the foregoing error types into SSL faults, and describe an extensive set of experiments to evaluate the proposed method. Our experiments demonstrate that high coverage of the modeled design errors can be achieved with small test sets.

1 Introduction

Design verification is the process of ensuring that a new design exhibits specified behavior. Many approaches to design error detection have been proposed based on formal verification [1]. However, formal verification is impractical for large logic circuits. In practice, such circuits are verified by simulation using representative input patterns (tests) [2]. A basic question that we address here is: Which tests should be applied and what is their efficiency?

Abadir et al. [3] have defined a set of likely design errors for combinational logic and have shown that complete test sets for single stuck-line (SSL) faults detect many, but not all, such errors. Recent research has considered the use of implementation-independent "universal" test sets [4,5], as well as random tests [6] for design error detection. In each case, the number of tests needed for good coverage of design errors can be excessive, and 100 percent coverage is not guaranteed. For example, universal tests exploit any unateness properties of the functions being implemented, but the tests become exhaustive when, as is often the case, there are no unate variables.

1. This research was supported by the National Science Foundation under Grant No. MIP-9200526 and by General Motors R&D Center.

In Section 2, we reduce the design errors considered in the literature to four classes. Then, we study the detection properties of these error classes. Section 3 describes the mapping of design errors into SSL faults, as well as the process of generating test sets for them using standard ATPG tools for SSL faults. Section 4 presents the results of applying our method to representative benchmark circuits.

2 Tests for Design Errors

Many types of design errors have been classified in the literature [3,5-7]. These error types are not necessarily complete, but they are believed to be common in the design process. We condense the errors identified by Abadir et al. [3] into four categories. (A similar classification is given independently in [5]).

- *Gate substitution error (GSE):* This refers to mistakenly replacing a gate by another gate with the same number of inputs. The extra and missing inverter errors of [3,5-7] are considered as substitution of an inverter for a buffer and a buffer for an inverter, respectively.
- *Gate count error (GCE):* This corresponds to incorrectly adding or removing a gate, and includes the extra and missing gate errors of [3]. This category is combined with gate substitution in [5], where, unlike here, XOR and XNOR gates are not considered. A class of "local" errors is defined in [6] which includes only some of the errors in this category.
- *Input count error (ICE):* This corresponds to using a gate with more or fewer inputs than required.
- *Wrong input error (WIE):* This error corresponds to connecting a gate input to a wrong signal. The "signal-like-source" error [6], is a special case of WIE. Although a WIE may be viewed as a multiple ICE, a multiple ICE cannot model a WIE in an inverter.

The errors in each category are studied next, and test patterns to detect them are determined. The following assumptions are made concerning the design to be verified:

- As in [3,5], we have a gate-level implementation that is purely combinational.
- The gate types used are AND, OR, XOR, NAND, NOR, XNOR, BUF (buffer) and NOT.
- Similar to [3,5-6], we have a functional specification of

the design which is completely simulatable, that is, any input pattern can be applied and produces a completely specified output.

- At most a single design error is assumed to occur. This assumption is analogous to the single stuck-line (SSL) fault assumption, which is the standard model used in testing for physical faults.

2.1 Notation

Let E be the set of all 2^n input vectors of an n-input gate G. We divide E into the disjoint subsets $V_0, V_1, \ldots, V_n$, where V_k contains all input vectors with exactly k 1s in their binary representation, $0 \le k \le n$. The disjoint sets V_{null}, V_{all}, V_{odd}, and V_{even} are defined as follows:

$$V_{null} = V_0 \; ; \; V_{all} = V_n \; ;$$
$$V_{odd} = \bigcup_{i=odd \wedge i \neq n} V_i \; ; \; V_{even} = \bigcup_{i=even \wedge i \neq 0 \wedge i \neq n} V_i$$

For example, in the case of 3-input NAND gate, $V_{null} = \{000\}$, $V_{all} = \{111\}$, $V_{odd} = \{001,010,100\}$, and $V_{even} = \{011, 101, 110\}$. The sets V_{null}, V_{all}, V_{odd}, and V_{even} are called the *characterizing sets* or *C-sets* of G.

Table 1 shows the output for each gate type in response to its various C-sets. The sets V_{null} and V_{all} are nonempty and always have cardinality one. For the single-input gates, V_{even} and V_{odd} are empty. For multiple-input gates, the set V_{odd} contains at least two elements, while the set V_{even} is empty only when $n = 2$. The cardinality of V_{even} (V_{odd}) is $2^{n-1}-1$ ($2^{n-1}-1$) when n is odd, and $2^{n-1}-2$ (2^{n-1}) when n is even. Finally, v_k denotes an arbitrary vector of the set V_k.

The above notation enables us to express sets of vectors in a simple way. For example, the complete test set for SSL faults on an n-input NAND gate is $V_n \cup V_{n-1}$. When $n = 3$, we can also write these tests as $V_{all} \cup V_{even} = \{111, 011, 101, 110\}$. In general, to verify the identity of a gate G, that is, to determine the tests required for verification, we use the above notation in conjunction with Table 1.

2.2 Gate Substitution Errors (GSEs)

According to experiments reported in [7], the most frequent design error made by humans is gate substitution, accounting for around 67% of all errors. Gate substitution refers to mistakenly replacing a gate G with another gate G' that has the same number of inputs. We represent this error by G/G'. For gates with multiple inputs, a *multiple-input GSE* (*MIGSE*) can have one of six possible forms: G/AND, G/NAND, G/OR, G/NOR, G/XOR, and G/XNOR. Each multiple-input gate can have five MIGSEs. For example, all MIGSEs can occur on an AND gate except G/AND which is not considered an error. For gates with a single input, i.e., buffers and inverters, a *single-input GSE* (*SIGSE*) can have one of two possible forms: G/NOT and G/BUF. Each single-input gate can have only one SIGSE. To cover extra/missing inverters in GSEs, a buffer can be inserted in each of a gate's fanout branches as well as inputs with fanout.

Table 1 The responses of the various gate types to their C-sets.

C-set	$n = 1$		n even (n odd and $n \ge 3$)					
	NOT	BUF	AND	NAND	OR	NOR	XOR	XNOR
V_{null}	1	0	0 (0)	1 (1)	0 (0)	1 (1)	0 (0)	1 (1)
V_{even}	n/a	n/a	0 (0)	1 (1)	1 (1)	0 (0)	0 (0)	1 (1)
V_{odd}	n/a	n/a	0 (0)	1 (1)	1 (1)	0 (0)	1 (1)	0 (0)
V_{all}	0	1	1 (1)	0 (0)	1 (1)	0 (0)	0 (1)	1 (0)

It has been suggested that most GSEs can be detected by a complete test set for SSL faults [3]. Our simulation study (Section 4) shows that such a test set can cover 80% to 100% of MIGSEs and 100% of SIGSEs. The actual coverage of MIGSEs is a function of the circuit structure, as well as the types of gates used in the circuit. Our goal is to achieve 100% coverage for GSEs.

A single-input gate can be identified by one test vector from either V_{null} or V_{all}. On the other hand, a multiple-input gate can be identified by three test vectors: one from V_{null}, one from V_{odd}, and one from V_{all} (if n is even) or V_{even} (if n is odd). Hence, three test vectors are sufficient to identify an n-input gate. Two test vectors suffice in some cases. For example, an AND gate can be identified by applying one test vector from V_{null} and one from V_{odd}.

The number of tests needed to test an n-input gate for SSL faults is $n+1$ for the gates AND, NAND, OR, and NOR, while it is two or three for XOR and XNOR depending on the parity of n. So, the number of tests needed to test for SSL faults is greater or equal to the number of tests needed to test for MIGSEs in most cases.

We now introduce some further notation to specify the effects of C-sets on a gate G within a circuit.

Definition 1 If the inputs of a gate G in a circuit C can be forced to the pattern v by assigning the primary inputs of C, then G is *controllable* by v, otherwise, it is *uncontrollable* by v. If the output of G with respect to the pattern v is sensitizable to a primary output then the response of v is said to be *observable* at G, otherwise, it is *unobservable*.

Definition 2 A gate G in a circuit C is *V-controllable* if G is controllable by at least one vector v in the input vector set V. If v is also observable at G, then G is *excitable* by V (*V-excitable*). A gate G is *fully excitable* if G is excitable by every nonempty C-set of G, otherwise, it is *partially excitable*.

To illustrate the above definitions, consider the circuit shown in Figure 1. G_1 and G_2 are both controllable by the pattern 00, while G_3 is uncontrollable by 00. The response to the pattern 00 is observable at G_2 but it is not observable at G_1. The gate G_3 is {00,11}-excitable because G_3 is controllable by 11, and the response of 11 is observable at G_3. On the other hand, G_3 is not {00}-excitable because G_3 is uncontrollable by all the elements of the set {00}. The gates G_1, G_2, and G_3 are partially excitable.

The following theorem gives a solution to the verification problem for GSEs:

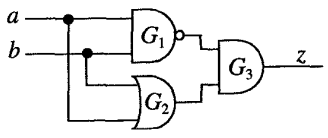

Figure 1 A circuit realizing the XOR function.

Table 2 The test vectors required to verify an n-input fully excitable gate.

Gate	Fanin n	Test set T_1	Test set T_2	Test set T_3
NOT (BUF)	$n = 1$	$\{v_{all}\}$ or $\{v_{null}\}$	$\{v_{all}, v_{null}\}$	$\{v_{all}, v_{null}\}$
AND (NAND)	$n = 2$	$\{v_{null}, v_{odd}\}$	$\{v_{all}, v_{odd}, v_{null}\}$	$\{v_{all}, v_{null}, v_{odd}\}$
	n odd	$\{v_{all}, v_{odd}\}$ or $\{v_{null}, v_{odd}\}$	$\{v_{all}, v_{odd}, v_{null}\}$	$\{v_{null}, v_{odd}, v_{all}, v_{even}\}$
	n even & $n \neq 2$	$\{v_{null}, v_{odd}\}$ or $\{v_{all}, v_{even}\}$	$\{v_{null}, v_{odd}, v_{all}, v_{even}\}$	$\{v_{null}, v_{odd}, v_{all}, v_{even}\}$
OR (NOR)	$n = 2$	$\{v_{all}, v_{odd}\}$	$\{v_{all}, v_{odd}, v_{null}\}$	$\{v_{all}, v_{null}, v_{odd}\}$
	n odd	$\{v_{null}, v_{even}\}$ or $\{v_{all}, v_{even}\}$	$\{v_{null}, v_{even}, v_{all}\}$	$\{v_{null}, v_{odd}, v_{all}, v_{even}\}$
	n even & $n \neq 2$	$\{v_{null}, v_{even}\}$ or $\{v_{all}, v_{odd}\}$	$\{v_{null}, v_{even}, v_{all}, v_{odd}\}$	$\{v_{null}, v_{odd}, v_{all}, v_{even}\}$
XOR (XNOR)	$n = 2$	$\{v_{null}, v_{all}\}$	$\{v_{all}, v_{odd}, v_{null}\}$	$\{v_{all}, v_{null}, v_{odd}\}$
	n odd	$\{v_{odd}, v_{even}\}$	$\{v_{odd}, v_{even}, v_{all}\}$ or $\{v_{odd}, v_{even}, v_{null}\}$	$\{v_{null}, v_{odd}, v_{all}, v_{even}\}$
	n even & $n \neq 2$	$\{v_{null}, v_{all}\}$ or $\{v_{even}, v_{odd}\}$	$\{v_{null}, v_{all}, v_{even}, v_{odd}\}$	$\{v_{null}, v_{odd}, v_{all}, v_{even}\}$

Theorem 1 *A necessary and sufficient condition for a test set S to verify a fully excitable gate is that S produce the test vectors T_1 shown in Table 2 at the inputs of the gate and sensitize the gate output to a primary output.*

All gates in a fanout-free circuit are fully excitable. In a circuit with fanout, it is possible that some input combinations cannot be forced at the inputs of some gates. For example, no element of V_{null} can be forced at the inputs of the AND gate G_3 in Figure 1. From Table 1, we see that V_{null} is necessary to distinguish a 2-input AND gate from an XNOR gate, so, the replacement of the AND by an XNOR gate cannot be detected. This replacement does not change the function of the circuit, hence it is considered to be a *redundant* or *undetectable* MIGSE. Likewise, some input combinations can be forced at the inputs of some gates but their responses cannot be observed. For example, the pattern 00 can be forced at the inputs of G_1 in Figure 1, but the response of G_1 cannot be propagated to the primary output. The above examples show that it is natural to have gates which are not fully excitable and therefore have undetectable design errors. It also suggests a modification of the test vectors T_1 in Table 2 to verify a partially excitable gate.

Definition 3 If a partially excitable gate G is excitable by all but one of its nonempty C-sets, then G is *strong partially excitable*, otherwise, it is *weak partially excitable*.

Consider again the circuit in Figure 1. The gates G_1, G_2, and G_3 are strong partially excitable because they are excitable by two out of the three nonempty C-sets of the respective gates. An example of a weak partially excitable gate is a 3-input XOR with all inputs connected to a single source. In this case, the gate is excitable by only two (V_{null} and V_{all}) of its four C-sets.

Since a strong partially excitable gate G is not excitable by one of the nonempty C-sets, one of its MIGSEs is undetectable. The remaining four MIGSEs on G can be detected with at least two vectors; Table 1 implies that an arbitrary vector detects only three of G's five MIGSEs. Therefore, we have to apply at least three test vectors to G, so that if G is not controllable by one of the vectors or one of the vectors' responses is not observable, then the other two will detect the detectable MIGSEs. This leads to the following result.

Theorem 2 *If all gates of a circuit are either fully excitable or strong partially excitable, then the test set T_2 shown in Table 2 will detect all detectable GSEs in the circuit.*

A further analysis of T_2 shows that to verify a weak partially excitable gate, we have to apply the patterns T_3 shown in Table 2. Since we cannot always assert that the gates in the design under test are fully excitable or strong partially excitable, we may have to apply the patterns T_3 to detect all GSEs. Note that a test set generated for GSEs assuming that the gates are weak partially excitable, will detect all GSEs in the circuit. On the other hand, a test set generated for GSEs by assuming the gates are fully excitable or strong partially excitable may not detect all GSEs.

A complete test set for SSL faults guarantees the detection of all SIGSEs [3]. Tests for MIGSEs also cover many SIGSEs. A circuit is *SSL-irredundant* if it contains no undetectable SSL faults.

Theorem 3 *A complete test set T for MIGSEs in an SSL-irredundant circuit is also a complete test set for SIGSEs on all circuit lines except inputs with fanout if T produces v_{all} at the input of every AND and NAND gate, v_{null} at the input of every OR and NOR gate, and their responses are observable.*

From this theorem, we conclude that detection of most SIGSEs is ensured by test sets T_2 and T_3 but not by T_1. Our experiments show that the test set T_3 detects all SIGSES in all but one of the considered benchmark circuits.

2.3 Gate Count Errors (GCEs)

We distinguish two types of gate count errors: extra-gate errors and missing-gate errors. An *extra-gate design error* (EGE) is defined as inserting a gate G' that has its m inputs taken from the n inputs of a gate G and feeding the output of G' to G. As a consequence, the number of inputs of gate G becomes $n - m + 1$. We represent an EGE by EG(G',G). It is easy to see that EG(AND,AND), EG(AND,NAND), EG(OR,OR), EG(OR,NOR), EG(XOR,XOR), and EG(XOR,XNOR) are undetectable or redundant. Explicit test generation for EGEs is not needed due to the following theorem:

Theorem 4 *A complete test set for GSEs is also a complete test set for EGEs.*

Most EGEs can also be detected by a complete test set for SSL faults, but this is not guaranteed. A complete test set for SSL faults in the circuit of Figure 2 is {000, 100, 001, 010}. This test set does not detect if the XOR gate is an extra gate. To do this, we need the vector 011.

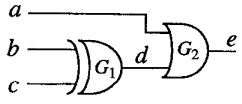

Figure 2 An example which shows an EGE not detected by a complete test set for SSL faults.

Figure 3 The missing-gate design error (MGE).

A *missing-gate design error* (MGE) is defined as removing a gate G' that has m inputs and feeds an n-input gate G, and then changing the inputs of G' into inputs of G; see Figure 3. As a consequence, the number of inputs of G becomes $N = n+m-1$. We represent the MGE by MG(G',G). As in the extra-gate case, the errors MG(AND, AND), MG(AND, NAND), MG(OR, OR), MG(OR, NOR), MG(XOR,XOR), and MG(XOR,XNOR) are undetectable.

Consider the problem of finding a minimal set of vectors that detect all MGEs in an N-input gate G. For each MG(G',G), we insert a gate G'' as shown in Figure 4, where G'' is chosen so that the function of the circuit is not changed. For example, if G is an AND or NAND, then G'' is an AND gate. We have to detect the GSE G''/G' in order to detect MG(G',G).

Theorem 5 *The test sets $V_N \cup V_{N-1} \cup V_{N-2}$, $V_0 \cup V_1 \cup V_2$, and $V_0 \cup V_2 \cup V_N$ are each sufficient and near-minimal for detecting MGEs on an N-input fully excitable AND (or NAND), OR (or NOR), and XOR (or XNOR) respectively.*

The test set given by the above theorem has one test more than the minimum. For example, the 11-member test set generated for MGEs in a 4-input NAND gate G is S = {1111, 1110, 1101, 1011, 0111, 1100, 1010, 1001, 0110, 0101, 0011}. If one of the tests {1110, 1101, 1011, 0111} is dropped, S still detects all MGEs. However, all MGEs in G cannot be detected with fewer than 10 vectors. In general, Theorem 5 gives near-minimal test sets for an N-input fully excitable gate. It is easy to prove that these test sets detect all the MGEs of an N-input partially excitable gate with high probability.

2.4 Input Count Errors (ICEs) and Wrong Input Errors (WIEs)

Input count errors (ICEs) are classified into extra input and missing input errors. An *extra input design error* (EIE) is defined as the replacement of an n-input gate ($n \geq 2$) by an $(n+1)$-input gate with the additional input connected to an arbitrary signal in the circuit. A *missing input design error* (MIE) is defined as the replacement of an n-input gate ($n \geq 3$) by an $(n-1)$-input gate with its $n-1$ inputs connected to an arbitrary subset of the original n inputs. We represent an EIE of a gate G by EI(e,G) where e is the extra input. We represent an MIE of a gate G by MI(m,G) where m is the source of the missing input.

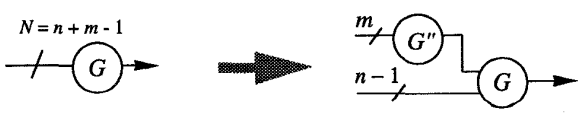

Figure 4 Reducing the problem of detecting MGEs to detecting GSEs.

To test for an EIE at a given input of an AND or NAND gate, the input must be set to 0 to activate the error, the other inputs must be forced to 1, and the gate's output signal must be propagated to a primary output. This is exactly the requirement of a test for a stuck-at-1 fault at the input of the gate in question. Similarly, testing for EIEs at the input x of an OR or NOR gate is the same as testing for a stuck-at-0 fault at x. To test for an MIE on an AND gate G, the inputs of G are set to 1, the signal considered to be missing is set to 0, and G's output signal is propagated to a primary output. This is more restrictive than a test for stuck-at-0 at the output of G. Similarly, testing for an MIE on a NAND, OR, and NOR is more restrictive than testing the gate output for stuck-at-1, stuck-at-1, and stuck-at-0, respectively.

The foregoing tests are complete for AND, NAND, OR, and NOR gates. Hence, a complete test set for ICEs in a given circuit detects all SSL faults at AND, NAND, OR, and NOR gates. A complete test set for ICEs also detects some SSL faults affecting XOR and XNOR gates. For example, testing for EIEs at the input of an XOR or XNOR gate is equivalent to testing for stuck-at-0 fault at the same input.

A *wrong input error* (WIE) is defined as a connection of a gate input to a wrong signal source. We represent a WIE on a gate G by WI(u,w,G), where u is the wrong input of the gate and w is the correct input. If a vector v detects WI(u,w,G), then it must set u and w to opposite values and propagate the signal at u to a primary output. WIE appears to be the second most common design error—around 17% of the errors reported in [7]. The relationship between MIEs and WIEs is stated in the following theorem:

Theorem 6 *A complete test set for MIEs on gates of type AND, NAND, OR, or NOR is a complete test set for WIEs on the same gates.*

In practice, we can rarely find a complete test set for MIEs. The fact that a given MI(x,G) is redundant does not imply that the WI(u,x,G) is redundant for every u. Also, a complete test set for MIEs does not guarantee the detection of WIEs in XOR, XNOR, NOT, and BUF gates. Hence, we cannot conclude that a test set for MIEs covers all WIEs.

The numbers of ICEs and WIEs in a circuit are very large—approximately $O(k^2)$, where k is the number of distinct signals in the circuit. Hence, we use simulation to extract the errors detected by the test set $S_T = S_{SSL} \cup S_{GSE} \cup S_{MGE}$, where S_{SSL}, S_{GSE}, and S_{MGE} are complete test sets for SSL faults, GSEs, and MGEs, respectively. In fact, all EIEs are detected by the test set for SSL faults alone [3], hence, we only have to generate tests for the undetected MIEs and WIEs. Our experimental results show that most MIEs and WIEs are detected by the set S_T.

A basic question concerning MIEs (WIEs) is the source of the missing (wrong) input. It must not depend on the erroneous gate's output, otherwise, the circuit can become sequential. Errors that make a combinational circuit sequential can be detected by a levelization procedure [2].

The coverage relationships for the various design errors are summarized as follows; A complete test set for MIGSEs

detects all EGEs. On the other hand, a complete test set for SSL faults detects all EIEs and SIGSEs. Complete test sets for MIEs, MGEs, and WIEs do not guarantee the detection of other error types. For example, a test for MIEs detects many, but not necessarily all, SSL faults.

2.5 Design Error Redundancy

We noted earlier that some design errors are undetectable. This leads to a type of redundancy that is quite different from that previously studied [8].

Definition 4 A gate G in a circuit C has *redundant inputs* if the function implemented by C is not changed when a proper subset of the inputs of G are removed. A circuit C is called *GI-irredundant* if no gate in C has redundant inputs.

GI-redundancy does not imply SSL-redundancy. For example, a 5-input XOR with all inputs connected to the same source is GI-redundant but SSL-irredundant. Similarly, SSL-redundancy does not imply GI-redundancy. For example, a buffer whose input is connected to ground is SSL-redundant but GI-irredundant.

A *redundant* design error is one for which no test vector exists. For example, the substitution of an XNOR gate for G_3 in Figure 1 cannot be detected by any input vector. Hence, the MIGSE G_3/XNOR is redundant. The following theorem characterizes redundant GSEs:

Theorem 7 *In a GI-irredundant and SSL-irredundant circuit C, the following holds: (1) C has no redundant SIGSEs; (2) If G/G' is a redundant MIGSE then every other MIGSE on G is irredundant, and if* $G \in \{XOR, XNOR\}$ *then* $G' \in \{AND, NAND, OR, NOR\}$ *and vice versa.*

Corollary 1 *If the gates in a GI-irredundant and SSL-irredundant circuit C are restricted to AND, NAND, OR, NOR, NOT, and BUF, then C has no redundant GSEs.*

The number of gates that can have redundant MIGSEs in a circuit C varies with the circuit structure and the types of gates in C. For example, fanout-free circuits have no redundant GSEs. On the other hand, a 2-input exclusive-or implementation using four 2-input NAND gates has four possible redundant MIGSEs: each NAND gate can be replaced with an XOR without affecting the overall exclusive-or function.

3 Test Generation

In this section we describe our test generation method for design errors. In order to use standard ATPG tools, we map the design errors into SSL faults. The mapping process consists of modifying the circuit's netlist and injecting a predefined set of SSL faults. A test set is then generated for the SSL faults in the modified netlist which detects all design errors in the original netlist.

To map MIGSEs and MGEs into SSL faults, each gate in the original netlist is replaced by a functionally equivalent circuit called a *gate replacement module*. A few carefully selected SSL faults are injected in the gate replacement module, so that the test for each injected fault forces the input of the gate to be a vector from one of the sets required to verify the gate. To cover all possible MIGSEs in a circuit,

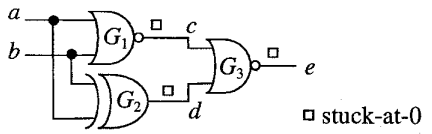

Figure 5 The replacement module for detecting GSEs in a 2-input AND gate.

we must assume that the gates are weak partially excitable. Consider, for example, the AND replacement module shown in Figure 5. The faults c stuck-at-0, d stuck-at-0, and e stuck-at-0 will force the inputs of the AND replacement module to v_{null}, v_{odd}, and v_{all}, respectively. These input patterns determine if the AND gate in the circuit is correct or not, i.e., the presence of any MIGSE on the gate is detected. The gate replacement modules for MIGSEs and MGEs on all gate types can be designed systematically in a similar way.

In general, the requirements to be met by a gate replacement module $M(G)$ of a gate G are the following:

- The function of $M(G)$ must be the same as that of G.
- A test for an injected SSL fault in $M(G)$ must force the input of G to a certain vector that is needed to verify G.
- The injected SSL faults must be sensitizable to the output of $M(G)$.
- If an injected SSL fault in M(G) is detected by a vector $v \in V_i$, then it must be detected by any vector of V_i. This requirement simplifies the detection of the injected SSL faults by the test generator, and leads to smaller test sets.

The mapping of MIGSEs and MGEs into SSL faults is many-to-one. Detecting a given set of injected SSL faults detects a larger set of MIGSEs and MGEs. For example, detection of the three SSL faults in Figure 5 detects five MIGSEs. There is a one-to-one correspondence between net errors (EIEs, MIEs, and WIEs) and SSL faults. The mapping of an EIE into an SSL fault is very simple; to detect whether an AND or NAND gate's input x is extra, we need to set x to 0, set every other input to 1, and propagate the gate's output signal to a primary output. This is the same as testing for x stuck-at-1. Also, to test for an extra input in an OR, NOR, XOR, or XNOR gate, a test for the input stuck-at-1 is required.

The detection of MIEs and WIEs is performed by inserting a mapping circuit called a *net attachment module*, as shown in Figure 6. Let C and C' be the circuits obtained before and after adding the net attachment module. The following requirements must be met:

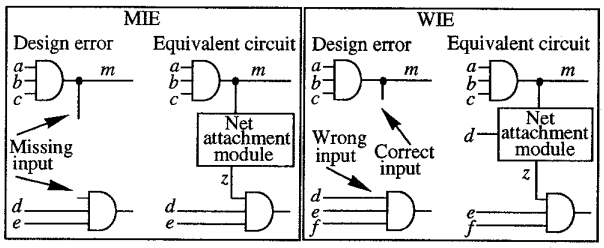

Figure 6 Mapping MIEs and WIEs into SSL faults.

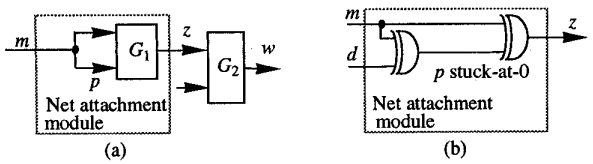

Figure 7 A net attachment module for (a) MIEs and (b) WIEs.

- The function of circuit C must be the same as that of C'.
- A test for the injected SSL fault in the net attachment module must detect the MIE or WIE.
- The injected SSL fault must be sensitizable in the net attachment module.

A typical design of the net attachment module for MIEs is shown in Figure 7a. If G_2 is an AND or NAND, then G_1 must be an XNOR and the fault p stuck-at-1 is injected. On the other hand, if G_2 is any of the gates {OR, NOR, XOR, XNOR}, then G_1 must be an XOR and the fault p stuck-at-0 is injected. In both cases, the output of G_2 is independent of z and hence the function of the circuit is not changed. Also, the SSL fault is sensitizable to the output of the net attachment module and the vector testing it detects MI(m,G_2). A typical design of the net attachment module for a WIE is shown in Figure 7b. The output of the net attachment module is $z = d$, hence the circuit function is preserved. The test for p stuck-at-0 forces opposing values on m and d, and hence the corresponding WIE will be detected by the same test.

The overall verification process is divided into two phases. The first phase checks for gate errors (MIGSEs and MGEs) and is shown in Figure 8. The second phase performs the error simulation for net errors (MIEs, WIEs) and then generates tests for the undetected ones. The flowchart of phase 2 is similar to that of phase 1. Complete coverage of net errors may require several iterations through phase 2. If after checking for all modeled errors, the implementation is found to match the functional specifications, we can conclude with high confidence that the circuit is correct as designed.

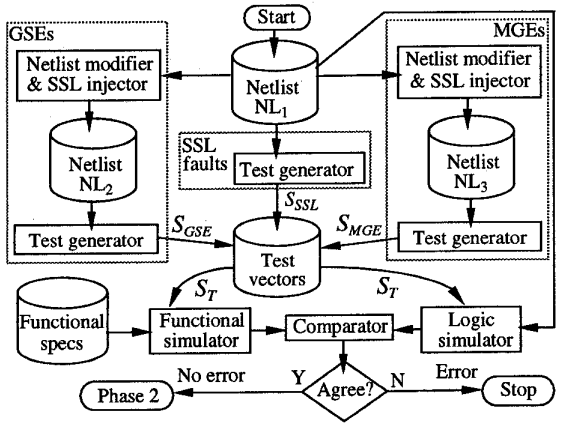

Figure 8 The first phase of the design verification process.

4 Experimental Results

In this section we describe the experiments performed to support the preceding analysis; these experiments used the ATPG tool ATALANTA [9]. To determine the capability of a given test set to detect design errors and SSL faults, we developed an error/fault simulator ESIM. The simulator uses parallel-pattern evaluation and critical path tracing techniques [2]; It simulates the circuit with multiple vectors concurrently and determines the detected errors/faults without explicit simulation of each error/fault.

The circuits used throughout the experiments are ISCAS 85 benchmarks [10] and standard TTL circuits [11]. Note that all the circuits except c432 and c499 are SSL-irredundant. We conducted a preliminary experiment to determine the coverage of design errors using a complete test set for all detectable SSL faults. The results in Table 3 show that a complete test set for SSL faults guarantees the detection of all SIGSEs and EIEs, confirming results in [3]. The detection of the other design errors is not guaranteed but they are likely to be detected because the test set does exercise each net in the circuit.

Our next experiments are concerned with generating almost complete test sets for all design errors. They use the method described in the previous section to generate test vectors targeting the indicated errors. The modified netlist is supplied to ATALANTA and a test set is generated. The

Table 3 The percentage of design errors detected using complete test sets for SSL faults.

Circuit	Test set size	Det'd SSL faults	Detected GSEs		Detected GCEs		Detected ICEs		Det'd WIEs
			SIGSE	MIGSE	EGE	MGE	EIE	MIE	
c17	5	100.0	100.0	80.0	100.0	n/a	100.0	57.5	88.0
c432	46	99.2	100.0	89.3	100.0	95.5	98.6	71.3	96.4
c432nr	44	100.0	100.0	89.1	100.0	95.5	100.0	73.1	96.9
c499	52	98.9	100.0	97.8	46.2	89.6	97.8	88.8	98.6
c499nr	52	100.0	100.0	97.9	46.2	93.8	100.0	88.8	98.9
c880	47	100.0	100.0	90.3	100.0	94.6	100.0	84.9	98.6
7485	25	100.0	100.0	88.4	100.0	89.8	100.0	83.4	92.7
74181	18	100.0	100.0	96.2	88.9	90.6	100.0	81.8	94.0
74283	12	100.0	100.0	91.3	100.0	84.1	100.0	74.5	92.2

Table 4 The percentage of errors detected using the generated tests.

Circuit	Tests targeting GSEs				Tests targeting MGEs		Error simulation for MIEs and WIEs using S_T		
	Test set size	Det'd MIGSEs	Det'd SIGSEs	Det'd EGEs	Test set size	Det'd MGEs	Test set size	Det'd MIEs	Det'd WIEs
c17	5	100.0	100.0	100.0	n/a	n/a	10	82.5	95.7
c432	34	91.9	99.4	100.0	92	99.8	170	86.5	98.8
c432nr	39	92.8	100.0	100.0	92	99.9	174	88.8	99.5
c499	41	99.8	100.0	46.2	45	97.1	136	93.0	99.5
c499nr	39	99.8	100.0	46.2	43	98.4	133	93.2	99.7
c880	49	92.8	100.0	100.0	66	100.0	162	95.0	99.8
7485	14	88.4	100.0	100.0	47	94.4	85	89.3	96.4
74181	15	98.5	100.0	88.9	36	99.5	69	94.9	98.8
74283	10	94.7	100.0	100.0	31	100.0	51	88.7	95.2

Table 5 Improved coverage of MIEs and WIEs after the second phase of test generation.

Circuit	Tests targeting MIEs not detected by S_T		Tests targeting WIEs not detected by S_T	
	Total test set size	Detected MIEs (%)	Total test set size	Detected WIEs (%)
c17	13	95.0	12	100.0
c432	184	87.1	206	99.3
c432nr	190	89.9	195	99.6
c499	228	95.7	147	99.6
c499nr	220	95.8	147	99.8
c880	225	96.5	192	99.9
7485	91	91.2	92	96.4
74181	83	96.6	78	98.9
74283	58	90.0	56	96.4

generated test sets are then evaluated using simulation to find their coverage of GSEs, as shown in Table 4. Since tests for MIGSEs cover EGEs (Theorem 4), the results on detecting EGEs are shown in Table 4. The coverage of MGEs using the generated test set is also shown in Table 4. Testing for MIEs, and WIEs is performed only for those errors that are not detected by error simulation using the set $S_T = S_{SSL} \cup S_{GSE} \cup S_{MGE}$. The coverage of EIEs will be the same as that shown in Table 3 because a complete test set for SSL faults detects all EIEs. The error simulation results for MIEs and WIEs also appear in Table 4. Tests are generated using ATALANTA for the remaining undetected MIEs and WIEs after the error simulation. ATALANTA reported that a large percentage of those errors are redundant. After adding the generated tests to S_T, the coverage of MIEs and WIEs is improved, as shown in Table 5.

The coverage of design errors using the generated test sets is quite high, 80%-100% for most cases. We are confident that most irredundant design errors are detected. To explore this further, we analyzed the circuits 7485, 74181, and 74283. We found that MIGSEs and EGEs not detected by our test sets are redundant, and hence, these test sets cover 100% of the detectable MIGSEs and EGEs.

It is difficult to compare the coverage results obtained in this paper to related work in the literature for the following reasons: (1) different error models are used; (2) test set sizes are missing from the results of [6]; and (3) standard benchmarks are not used in most prior work. Moreover, the CPU times cannot be accurately compared because ATALANTA runs on a SUN workstation while ESIM runs on an IBM PC. The CPU times were found to range from a few seconds to a few minutes in all cases.

5 Conclusions

We have presented a method for verifying logic circuits by using standard simulation and ATPG tools. We showed that all common design errors can readily be mapped into SSL faults and presented a systematic method to perform this mapping. Our experimental results show that complete test sets for SSL faults detect almost all detectable errors. Our test sets for design errors are small in size and provide high coverage—the percentage of detected design errors from all modeled errors, detectable and redundant, is greater than 90% for most benchmark circuits. Our experiments also show that the fraction of redundant design errors is significant in practical circuits even when the circuit is SSL-irredundant. For example, 11.6% of the MIGSEs in 7485 comparator circuit are undetectable.

Our design verification method is directly applicable to gate-level sequential circuits. All we need is a sequential ATPG tool that gives test sets for SSL faults with high coverage. We are now extending our verification method to high-level circuits as well as gate-level design error diagnosis. These extensions cannot be easily performed using BDD-based design verification methods, because of their high memory requirements and their independence of the design structural representation.

We stress that our design verification method can be used with any ATPG tool because it does not require modification of the test generation program. We ensure full detectability of design errors by injecting SSL faults into a modified netlist and apply an ATPG program to it. Therefore future improvements in ATPG tools can be extended directly to design error detection.

Acknowledgments

We wish to thank Krishnendu Chakrabarty and Mark Hansen for their helpful comments.

References

[1] M. Yoeli (ed.), *Formal Verification of Hardware Design*, IEEE Computer Society Press, Los Alamitos, Calif., 1990.

[2] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, New York, 1990.

[3] M. S. Abadir, J. Ferguson, and T. E. Kirkland, "Logic design verification via test generation", *IEEE Trans. on CAD*, Vol. 7, pp. 138-148, Jan. 1988.

[4] D. Brand, "Exhaustive simulation need not require an exponential number of tests", *IEEE Trans. on CAD*, Vol. 12, pp. 1635-1641, Nov. 1993.

[5] B. Chen, C. L. Lee, and J. E. Chen, "Design verification by using universal test sets", *Proc. Third Asian Test Symposium*, 1994, pp. 261-266.

[6] S. Kang and S. A. Szygenda, "The simulation automation system (SAS); concepts, implementation, and results", *IEEE Trans. on VLSI Systems*, Vol. 2, pp. 89-99, March 1994.

[7] E. J. Aas, T. Steen, and K. Klingsheim, "Quantifying design quality through design experiments", *IEEE Design and Test*, Vol. 11, pp. 27-37, Spring 1994.

[8] J. P. Hayes, "On the properties of irredundant logic networks", *IEEE Trans. on Computers*, Vol. C-25, pp. 884-892, Sept. 1976.

[9] H. K. Lee and D. S. Ha, "On the generation of test patterns for combinational circuits", Dept. of Elec. Eng., Virginia Tech., Rep. 12-93, 1993.

[10] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran", *Proc. IEEE International Symposium on Circuits and Systems*, 1985, pp. 695-698.

[11] Texas Instruments, *The TTL Logic Data Book*, Dallas, 1988.