Identifying Redundant Gate Replacements in Verification by Error Modeling

Katarzyna Radecka and Zeljko Zilic
McGill University, Montreal, Canada
{kasiar,zeljko}@macs.ece.mcgill.ca

ABSTRACT

This paper considers verification of combinational circuits by test vectors under assumption of gate and wire replacement faults. Identifying redundant faults is critical to the quality and speed of such verification schemes. We propose the first known exact redundancy identification of gate replacement faults, together with its efficient approximations. While both solutions use the SAT formulation of redundancy identification, we propose the means to effectively use any single stuck-at-value redundancy identification in the approximate schemes, with varying detection accuracy. Critical to the latter are the novel uses of don't care approximations that detect many redundant faults and quickly identify those that can be detected by methods for stuck-at value faults. Test generation scheme that uses the error-correcting properties of Arithmetic Transform is incorporated into the overall verification procedure, and is shown to provide high fault coverage for these fault models.

1. INTRODUCTION

The goal of design verification is to ensure that a circuit behaves as specified. Among methods for detecting design errors, techniques based on test vector simulations are easiest to adopt by engineers. They are shown to augment and overcome the limitations of formal verification [5], [16]. In this paper, we address the issues important for simulation-based verification, i.e., modeling of design failures in combinational logic at the gate level, and the identification of redundant faults belonging to a model.

An integral part of the verification of digital circuits via simulations is an *error model*, needed to obtain a test set for failure detection and providing confidence in fault coverage. Such error models have been proposed in [2], [3] and [6]. A large class of these failures consists of erroneous replacements of a gate or wire in a network with another gate or wire, respectively. It was shown in [6] that most actual errors encountered fall in this category. By applying modern design flows, 98.9% of all design errors fall within this error model for the case of the DLX processor, and 94.2% for PUMA floating point units. Also, authors in [1] reported that 97.8% of design errors that occur during the manual interventions belong to the category of an error model from [2].

The early approach to representing the design errors at the gate level considered failures, which were proven to be detectable by testing for single or multiple stuck-at-value (s-a-v) faults [2]. It was shown that the substitutions with *AND*, *OR*, *NAND* and *NOR* gates could be tested by s-a-v methods. Test sets were obtained through the automatic test pattern generation (ATPG) for that model [2], [3].

The major obstacle in applying the simulation-based verification is that the coverage and the running time of testing procedures is seriously impaired by redundant faults. For example, the obtained fault coverage of 13x7 array divider is only 75.5% if the redundant faults are not detected, and 100% otherwise. Unless applied exhaustively, simulations alone cannot deal with that problem. Redundancy identification has been mainly considered in the context of stuck-at faults. The algorithms use either structure-based search [15] (or its quick approximations [11]), or rely on algebraic approaches, such as the satisfiability formulation and its extensions [12], [7]. In [3], the issue of redundancy identification in the context of implementation verifications by test vectors was brought up. The stuck-at ATPG-based scheme was applied to determine redundancies for a subset of gate replacements by multiple runs of s-a-v redundancy identification for each gate replacement fault.

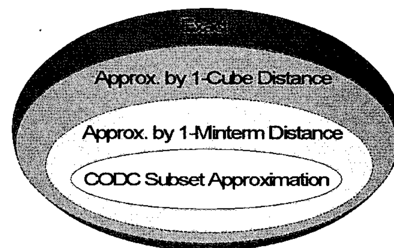

Figure 1: Relation Among Proposed Methods

In this paper, we propose novel procedures for identifying whether the gate replacement faults are redundant. In Section 3, safe approximations to local don't cares are used to eliminate many redundant faults and also identify those faults that are either likely to be redundant or detected by standard s-a-v methods. Undetected faults are then identified by extending a s-a-v redundancy identification. In Section 4 we construct an exact identification for replacement faults, using an all-

0-7803-7169-0/01 $10.00 © 2001 IEEE

SAT formulation. Figure 1 relates the considered methods.

Our redundancy identification schemes are applied to evaluating the test generation scheme based on Arithmetic Transform (AT) decoding algorithms [17] that can be parameterized by the size of the error.

2. REPLACEMENT FAULTS

In this paper, we say that a *replacement fault* is an erroneous substitution of either a gate or a wire. In addition to verification by error modeling [6], [19], these faults are shown to be present in deep sub-micron manufacturing fault models [8], [9]. Replacement faults are affecting either single or multiple nodes. Gate replacements influence single nodes, while wire replacements affect either single or multiple nodes.

Definition 1: *A single gate replacement error (SGRE) erroneously substitutes a single gate in the netlist with another gate of the same number of inputs and outputs.*

Please note that this definition also includes the missing/added inverter, as well as the extra gate added at the inputs of the original gate, as shown in Figure 2. These faults are simply subsumed to either fan-out of fan-in of the immediate neighboring node.

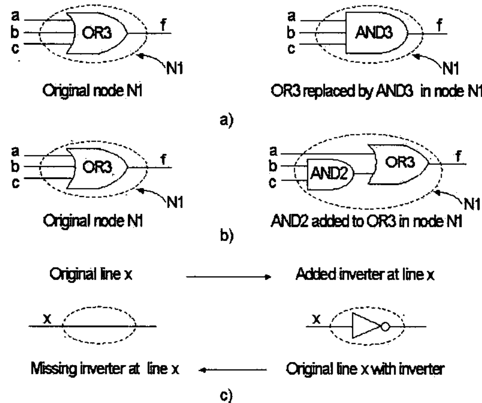

Figure 2: Errors in a Netlist

The gate replacements, unlike s-a-v faults, affect only the components, while the interconnect stays fault-free. Therefore, the deterministic tools for test pattern generation and redundancy identification, which work well with the s-a-v fault models, cannot be straightforwardly applied to the detection and redundancy identification of the gate replacement errors. To solve this problem of incompatibility of tools and error models, we can construct the transformation of a gate replacement error into s-a-v fault domain, and then use standard testing tools for the s-a-v detection and redundant error identification. Alternatively, we can treat gate replacement errors as an independent problem, and look for some alternative ways of their detection. The former approach is summarized in Section 2.2, the latter is the scheme proposed in this work, and is described through the remainder of this paper.

2.1 Redundant Replacement Faults

A *redundant replacement* at a given node is a substitution that does not change the original function of the circuit. Redundant errors of any kind significantly deteriorate the performance of a testing scheme. By identifying them, we avoid costly simulations of these faults.

There are $2^{2^n}-1$ possible SGREs at the n-input node. To the set of all SGREs, as special cases, belongs up to $2(n+1)$ single s-a-v faults associated with the node. Unlike a s-a-v fault, which permanently ties a signal to either 0 or 1, the polarity of an error caused by a replacement fault depends on stimuli. To deal with such faults, we are forced to seek new redundant fault identification methods.

2.2 Previous Work

Contributions to the detection of the gate replacement errors were presented in [2] and [3]. In [2] authors observed that the complete test set for single stuck-at faults detects all single gate replacement errors, when errors are restricted to gates: *AND*, *OR*, *NAND* and *NOR*.

In [3], a method was proposed for SGRE detection and identification of redundancies. This approach first generates a test set that uniquely distinguishes each gate from all the other gates in the library. Then, the replacement model of each gate is constructed. The model, consisting of few gates, is functionally equivalent to the original gate. Its role is to provide the "infrastructure" for the set of single s-a-v faults which, when injected into the model can be detected only by the set of vectors which uniquely distinguishes a given gate from all the other gates, Figure 3. Finally, the ATPG is called (possibly) several times per given SGRE in order to detect at least one of the s-a-v faults superimposed on the gate model. The detection of at least one such s-a-v fault guarantees that the SGRE, represented by the model will be detected. Consequently, the gate replacement error is declared redundant if none of the s-a-v faults is detected by ATPG.

Figure 3 illustrates the model for erroneous replacement with gate AND2, together with all the single s-a-v faults injected to this model. V_{null} is the test vector consisting of only "0"s, V_{odd} is the set of vectors with odd number of "1"s and V_{all} is the vector of all "1"s.

The complete test sets for detecting AND2 and all the imposed s-a-v faults in Figure 3 are presented in Table 1.

Entry "e_{ff}" is the fault free behavior of the model, while the entries to the right of "e_{ff}" describe the behavior of the model subjected to the injected single s-a-v faults.

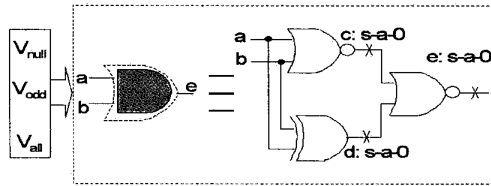

Figure 3: The Replacement Module for Detecting a Replacement Error with AND2

a	b	Vec.	e_{ff}	c:s-a-0	d:s-a-0	e:s-a-0
0	0	V_{null}	0	1	1	0
0	1	V_{odd}	0	1	1	0
1	0	V_{odd}	0	1	1	0
1	1	V_{all}	1	1	1	0

Table 1: Vectors Detecting Errors in Figure 3

Note that the single s-a-v faults in the replacement model tackled by APTG bear no relation to the original single s-a-v faults of the circuit, which are targeted during testing process. Hence the results of verification for SGREs cannot be reused in testing for single s-a-v faults, and vice versa.

Secondly, in order to detect each irredundant SGRE we should expect, on average, multiple passed of ATPG. In the case of redundant SGRE, however, multiple runs of ATPG are guaranteed. Therefore, although it is feasible to used ATPG in detecting single gate replacement errors, this may be not the most efficient way to do that.

2.3 Overview of our approach

In this paper we present the methods that can handle all the possible SGREs without restrictions to particular gates. This was not the case in the method from [3], although we understand that their extensions could possibly handle all the cases.

We will identify the conditions that prohibit either the excitation of a SGRE with the appropriate inputs, or the propagation of the erroneous result to the primary outputs. The redundant fault identification has been mostly considered in context of s-a-v faults. To benefit from the wealth of s-a-v methods, we propose a scheme that for many SGREs results in a s-a-v fault representation. In contrast to the multiple s-a-v representations, [3], we consider at most one s-a-v fault per SGRE. Then, any s-a-v technique can detect all such SGREs. Additionally, our method identifies in the process many other SGREs, which cannot be represented as single s-a-v faults.

We will demonstrate this procedure in the way analogous to the well-known s-a-v SAT formulation by Larrabee [13]. Structural s-a-v ATPG approaches can be applied as well. Further, an exact SAT formulation is presented in Section 4; together with the preprocessing steps for the algorithm speedup.

3. REDUNDANCY DETECTION BY DON'T CARES

Redundancies are caused by DC conditions at nodes affected by faults. These are either *observability* DC (ODC) or *controllability* DC (CDC) conditions inhibiting the error detection.

Our first redundant fault identification consists of two steps, both of which can be performed in varying degree of approximation. First, we use the *don't care* information in the network, to screen out most of the redundant faults. We always use the approximate don't care (DC) construction, for performance reasons and to deal with multiple node faults (wire replacements). The use of don't care subsets guarantees that no irredundant fault will be declared redundant. For selected remaining faults, we apply the modification of single stuck-at-fault redundancy identifications. One such method, based on the satisfiability (SAT) formulation of the problem is employed. Information on which replacements are to be probed by the SAT approach is provided by the don't care sets obtained in the first step.

The controllability and observability analyses, such as SCOAP [10] and its successors, have been employed in guiding the ATPG branching heuristics. While such methods are quick and suitable for their intended application, if applied to directly detect redundancies, they can result in two-sided errors – the redundant faults might be estimated as redundant or vice versa. Unlike that, our proposal avoids that problem completely - irredundant faults will never be discarded as being redundant. This is why we say that our approximations are *safe*.

3.1 Using Local Don't Cares

We will explicitly deal with *local don't care* sets at a given node, and their complements – *local care sets.*

Definition 2: *A local don't care set associated with a given circuit node constitute of CDCs and ODCs associated with this node. A local care set ($Care_{local}$) of a given node is the complement of the local don't care set.*

Note that the DCs observed at the primary I/Os of the circuit are the *global DCs* associated with this circuit.

Each replacement gate h that coincides with the original gate g on a local care set, $Care_{local}$, at a given node, creates a redundant fault.

Lemma 1: *Let a gate g be replaced with a gate h. By considering their respective ON-sets, g^{ON} and h^{ON} and local care set $Care_{local}$, the replacement is redundant if:*

$$g^{ON} \cap Care_{local} = h^{ON} \cap Care_{local}.$$

Proof: Examine the local care set at a node. This set alone can be both excited and observed. By replacing gate g with a gate that coincides with g on the care set, the original function has not changed. Hence, the substitution is redundant. ♦

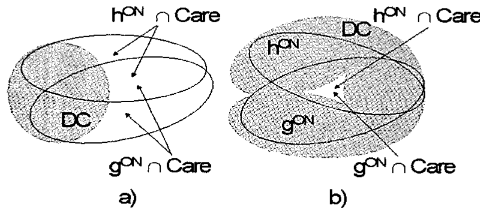

Figure 4: Don't Care Set Influence on Replacement Errors

Figure 4 shows how a gate replacement error with $g^{ON} \neq h^{ON}$ can become hard to detect or even redundant when a DC set overlaps the difference between the two. The DC set that partially intersects with the Boolean difference between g^{ON} and h^{ON} creates the irredundant error, as in Figure 4.a. Due to DCs, fewer test patterns might detect the error gate h. The DC set that completely covers the difference between the two creates the redundant error, Figure 4.b.

Note that for all possible replacement faults, DC sets have to be obtained only once. They are often available for free as a by-product of the synthesis. If we have the full don't care sets, Lemma 1 provides us with the exact redundancy identification. Practical schemes use only a subset of DCs.

3.1.1 Don't Care Approximations

Calculating don't care sets in a network consists of two parts. CDCs that occur by impossible input combinations are usually generated by determining an image of the care set at the input of the network to a node of interest. ODCs are obtained by traversing the network backwards. At each node, the ODCs are calculated as a Boolean difference among the downstream DCs, to which local satisfiability DC (SDC) conditions are added. SDCs simply express impossible assignments of values at a node, for given gate; if the gate changes, the SDCs change as well. Of all DC conditions, ODCs are the most time consuming. Because of their size, they are commonly represented by BDDs associated with each node. The limitations on the practicality of this approach come from the ability to handle large BDDs. The exact local don't care determination often results in BDDs that are too large, and therefore we approximate this set by *Compatible Observability* DCs (CODCs) [18]. They are generated by traversing the network backwards. At multiple fan-out nodes, CODCs of the fan-out nodes are simply intersected. The advantage of ODC subsets obtained this way is that the multiple node replacements will not change their value. Hence, CODCs can be used for multiple location faults, such as wire replacement faults.

3.1.2 Using S-A-V Redundancy Identification - Single Minterm Approximation

Since we employ subsets of don't cares, not all redundant replacements will be detected by this approach alone.

Definition 3: *An approximated don't care set associated with a given node is the set of local don't cares where ODCs are approximated with CODCs, while other DCs are exact. Correspondingly, an approximated care set ($Care_{approx}$) of a given node is the complement of the approximated local don't care set of this node.*

The approximated local don't care set (DC_{approx}) is a subset of the exact local don't care set (DC_{local}) at a given node

$$DC_{approx} \subseteq DC_{local},$$

while the $Care_{approx}$ is the superset of the exact local care set of this node.

Consider the following use of the Hamming distance $d: \{0,1\}^n \times \{0,1\}^n \mapsto N$ between two Boolean functions. The distance is equal to the number of minterms (w) of the Boolean difference between the two intersections of functions with the local care set:

$$d_{Care}(g,h) = w\left(\left(g^{ON} \cap Care\right) \oplus \left(h^{ON} \cap Care\right)\right) > 0. \quad (1)$$

Symbol "⊕" denotes the Boolean difference between two sets, which can be calculated by XOR-ing their characteristic functions.

Distance d_{Care} in Equation 1 is positive for redundant faults not detected by the use of $Care_{approx}$, as well as for irredundant replacements. Our next task will be to distinguish these two cases. The distances obtained by the exact and approximate local DC sets can be related, based on the size of these sets, by the following Lemma.

Lemma 2: *The following relation is true for the Hamming distances between gate g and its faulty replacement h*

calculated with the use of local (d_{local}) and approximated (d_{approx}) DC sets:

$$d_{local}(g,h) \le d_{approx}(g,h).$$

Proof: We rewrite the distance function in Equation 1 as:

$$w((g^{ON} \cap Care) \oplus (h^{ON} \cap Care)) = w((g^{ON} \oplus h^{ON}) \cap Care).$$

Since the approximated set is a superset of the exact local care set, i.e., $Care_{approx} \supseteq Care_{local}$, the distance from Equation 1 can only be larger for the approximated set. ♦

The distance information provides the means to formulate a series of further refinements to filter out the irredundant faults from those identified in Equation 1. If the distance is small, it is possible that the fault was redundant, but not detected due to the use of DC subsets.

Example 1: Consider a replacement error at node *b* of a circuit in Figure 5.

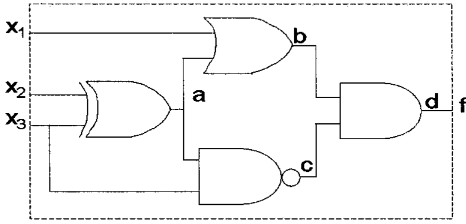

Figure 5: Circuit with replacement Error at Node ***b***

Assuming the full controllability, the local DC set at node *b* consists only of ODCs. Equations describing original function *f*, ODC and $Care_{local}$ sets are shown in Table 2.

Original Function	$f = x_1 * x_2 * x_3 + x_1 * x_3' + x_2 * x_3'$
Local DC Set	$ODC_b = x_2 * x_3 + x_3'$
$Care_{local}$	$Care_{local} = ODC_b' = x_2' * x_3$

Table 2: ODC and $Care_{local}$ Sets for Node *b* in Fig. 3

Let *L* = {*AND*, *NOR*, *NAND*, *XOR*, *XNOR*} be the possible gate replacements, and let *W* denote the exchanging of input wires at node *b*. Table 3 describes the effects of the replacement faults at node *b*. The first row includes the Hamming distances between the correct and faulty gates with no DC set taken into account.

	W	XOR	AND	NAND	XNOR	NOR
d	0	1	2	2	3	4
d ∩ Care	0	1	1	1	1	4
# vec	0	1	3	3	5	6

Table 3: Hamming Distances and Number of Vectors Detecting Error Gates for Circuit in Fig. 3

The Hamming distance, calculated according to Equation 1, is shown in the second row of Table 3. The number of test vectors detecting the fault (the third row) increases with the Hamming distance.♣

Further redundancy identification can be parameterized by the nonzero distance

$$d((g^{ON} \cap Care_{approx}), (h^{ON} \cap Care_{approx})) < \varepsilon \quad (2)$$

to guarantee that the s-a-v redundancy detections can be directly applied. For that, we define the class of *single-value faults*.

Definition 4: *We say that the replacement fault that results in only one polarity (0 or 1) of faulted value at a gate output is a single-value (0 or 1) fault. For non-failing stimuli, it behaves as a correct circuit, and can carry signals of both polarities.*

An example of a single-value replacement fault is a substitution of a two-input *AND* with *XNOR* gate. For input stimulus (0,0), it behaves as an output s-a-1 fault. Otherwise, it behaves as a correct circuit.

Lemma 3: *The gate replacement h, that is within a Hamming distance of one:*

$$d(g^{ON} \cap Care_{approx}, h^{ON} \cap Care_{approx}) = 1$$

results in a single-value fault at the output.

Proof: Since the functions differ in only one minterm, only one polarity of the fault can be present. ♦

3.1.3 Redundant Single Cube Replacements

A closer approximation can be considered to detect more redundant faults while employing the modified s-a-v identifications. We will checked for redundancies the replacements of distance 1, measured in cubes, instead of minterms.

Lemma 4: *If the gate replacement h is within a single-cube distance, and if*

$$g^{ON} \cap Care_{approx} \ge h^{ON} \cap Care_{approx} \quad (3)$$

then, the replacement is a single-value 0 fault. For a single-value 1 fault, sign "≤" is used instead in the inequality.

Proof: The functions in this case differ in more than one minterm. By inspecting the intersections with $Care_{approx}$, it follows from Equation 3 that the replacement is of single value 0. The condition opposite to Equation 3 then holds for a *single-value* 1 replacement. ♦

Theorem 1: *The replacements satisfying Lemma 4 are redundant iff a single stuck-at fault at the gate outputs is*

redundant for the single-cube input assignment that distinguishes the two functions.

Proof: (⇒) Redundancy of such replacement implies single stuck-at redundancy restricted to the input cube for which the functions differ.

(⇐) If the s-a-v is redundant, and only the input cube for which the functions differ is exercised, then we have a single-value replacement, which is redundant. ♦

Example 2: Consider an error caused by replacing the 2-input *OR* gate in the circuit in Figure 5 with an *XOR* gate. The replacement forces a single-value 0 fault for a single cube ($x_1 = 1$, $a = 1$) assignment of its inputs. This fault can be tested as a s-a-0 fault with an additional constraint: $x_1 = 1$, $a = 1$. ♣

As a corollary to Theorem 1, single s-a-v redundancy identifications can be employed. To adjust the s-a-v approaches, we rely on SAT-based identifications.

3.2 Use of SAT in Redundancy Identification

Satisfiability formulation for testing for s-a-v faults has been elaborated in [13]. For each fault in a circuit, a conjunctive normal form (CNF) is constructed. Such product of sums (*clauses*) is equal to one for all solutions, and finding one satisfying assignment amounts to obtaining a test vector. If there is no solution, i.e., if the expression is unsatisfiable, then the fault is redundant. The SAT formulation for single-stuck-at fault redundancy identification and/or test generation consists of several types of clauses. *Good circuit clauses* represent the correct operation of the whole circuit. *Faulty circuit clauses* describe the effects of a single stuck-at fault on the downstream network nodes. *Active clauses* are introduced to give the activation conditions of a fault. Finally, the *fault site* and *goal clauses* describe the activation and observation of the fault.

Example 3: The following clauses are generated for a s-a-0 fault at node b of the circuit in Figure 5. The variables are associated with nodes in the network. Those with no subscripts are the good clause variables, x, the faulty circuit variables at the same node have the subscript f, as in x_f, while the activation variables have the subscript a. The nodes and conditions for which the individual clauses are created are placed in square brackets.

Good circuit clauses: [OR]: $(x_1 + a + \bar{b})(\bar{b} + x_1)(\bar{b} + a)$,

[NAND]: $(x_3 + \bar{a} + \bar{c})(a + c)(x_3 + c)$,

[AND]: $(\bar{b} + \bar{c} + d)(b + \bar{d})(c + d)$,

[XOR]: $(x_2 + x_3 + \bar{a})(x_2 + \bar{x}_3 + a)(\bar{x}_2 + x_3 + a)(\bar{x}_2 + \bar{x}_3 + \bar{a})$.

Faulty circuit clauses:

[AND]: $(b_f + \bar{d})(c + \bar{d})(d + b_f + c)$.

Active clauses: [Active ⇒ (Good ≠ Faulty)]:
$(\bar{b}_a + b + b_f)(\bar{b}_a + \bar{b} + \bar{b}_f)$,[Active ⇒ *Output$_a$*: $(\bar{b}_a + d_a)$.

Fault Location: [Node b s-a-0]: $b_a b \bar{b}_f$.

Goal: [Active Output]: d_a

A solution to this SAT problem, e.g. the satisfying input assignment $x_1 = 1$, $x_2 = x_3 = 0$ produces a test vector. ♣

This approach can be time consuming if applied to all faults. Next, we show how our DC-based algorithm can be used to filter out many cases of replacement faults.

3.2.1 Passing Proximity Information to SAT

We can derive a SAT formulation for replacements not filtered by DCs using Lemma 4. The distance between the original and the replacement gate, obtained by an approximated DC set, can be passed to SAT. This proximity information can represent additional criteria for creating further approximations to the problem.

By considering only the single-cube distance replacements, as in Theorem 1, a simple and efficient SAT formulation can be obtained. We first create a s-a-v SAT instance corresponding to the polarity of the single-value faults, according to Lemma 3. It is sufficient to add to that CNF the 1-clauses that restrict the gate inputs to a single failing cube.

Example 4: To obtain a SAT instance for replacing the *OR* gate from Figure 5 with an *XOR* gate, clauses are added to restrict the inputs to their (single-cube) assignments that differentiate the gate functions. The following 1-clauses are added to those for s-a-0 fault at node b (Example 3).

Additional clauses: [OR → XOR Replacement Inputs]: $x_1 a$. ♣

If the original and replaced function differ in one cube, say in k literals, only k 1-clauses will be added. Adding each 1-clause amounts to assigning the values to the variable throughout the CNF, i.e., restricting the search space by factor of 2. This contributes to a total reduction by a factor of 2^k. Hence, it might be significantly quicker to find the solution to this problem, than to the original s-a-v instance.

The approximate algorithm for redundant gate replacement identification is outlined in Algorithm 1. After intersecting the original and replaced gates with the approximate care sets, their Boolean difference (⊕) is obtained. If the difference is empty, the fault is redundant, and is not simulated. Otherwise if the difference is a single cube with properties from Lemma 4, a 1-Cube check is performed by s-a-v identifications, augmented with enforcing the distinguishing single-cube input assignment. If redundancy is not detected, the fault is

assumed irredundant and is simulated by several top lattice layer vectors.

```
1.  Generate CODC Approx. of DC set for the network
2.  for each fault (g → h)
3.  {
4.    Obtain: (h ∩ Care_approx), (g ∩ Care_approx),
           l = (h ∩ Care_approx) ⊕ (g ∩ Care_approx)
5.    if (h ∩ Care_approx = = g ∩ Care_approx)
6.      { /* l = = Ø */
7.        break;
8.      }
9.    if (d(h ∩ Care_approx, g ∩ Care_approx) = 1)
10.     { /* l = 1, 1-Cube approx. */
11.       if (h ∩ Care_approx ≥ g ∩ Care_approx)
12.           if (detect_s-a-0_with_cube(l))
13.             break;
14.       else if (h ∩ Care_approx ≤ g ∩ Care_approx)
15.             if (detect_s-a-1_with_cube(l))
16.               break;
17.     }
18.   simulate_fault_n_lattice_layers (g, network)
19. }
```

Algorithm 1: Approximate Redundancy Identification

4. EXACT REDUNDANT FAULT IDENTIFICATION

The previous solution had the advantage of reusing fast stuck-at fault methodologies, while possibly missing some redundant faults. We next present an all-SAT formulation that is exact.

The SAT formulation uses the good circuit, faulty circuit and active clauses that are the same as in the standard single stuck-at SAT formulations. However, fault site clauses will be modified in the following way.

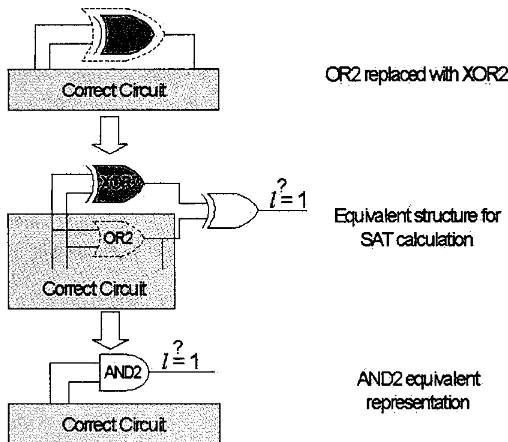

Figure 6: Exact SAT for a SGRE

To describe the conditions for activating a fault, the assignments of the gate inputs must be such that $g \neq h$, regardless on the fault polarity. We create an auxiliary node that is equal to $l = g \oplus h$. Then, the fault location clause will assert its value to $l = 1$, as shown in Figure 6, on an example of replacing an OR with an XOR gate.

Hence, a SAT-based formulation of the redundant replacement fault identification can be created by the addition of a node that differentiates the original and the replacement function, and by generating a clause that forces this gate to 1. We contrast this to the verification approach in [4], where a comparison *XOR* gate is added to two complete circuit implementations, resulting in a substantially larger problem.

Example 5: To formulate a SAT instance for replacing the *OR* gate from Figure 5 with an *XOR* gate, clauses are added which restrict the inputs to the assignments that differentiate the gate functions. In this case, it is *OR* ⊕ *XOR* = *AND*, as shown in Figure 6. At the same time, as polarity of the fault is not known, the stuck-at fault clauses should be removed.

Additional clauses: [OR → XOR Replacement ⇒AND]: $(\bar{x}_1 + \bar{a} + b)(x_1 + b)(a + \bar{b})$

Removed clauses: [s-a-0]: $b\bar{b}_f$. ♣

4.1.1 Preprocessing

Since invoking SAT instances can be costly, the preprocessing steps usually precede SAT procedures. Often, random simulations are applied. Only if they fail to distinguish the correct and faulty circuits, SAT is invoked.

In our case, properties of the Arithmetic Transform (AT) vector set can be exploited to surpass what is usually possible with random vectors. It is known [17] that, by selecting the top $\lceil log_2(t+1) \rceil - 1$ layers of an input space Boolean lattice, any fault with less than t-term AT will be detected. A smaller number of such layers can be used for preprocessing. Upon their failure in detecting a fault, they can be passed to the SAT procedure, as non-satisfying assignments. This restricts the search space for satisfying assignments resulting in speeding up SAT. Unlike random vectors, this set has a very compact description.

5. EXPERIMENTAL RESULTS

The redundancy identification schemes have been implemented and run on 440 MHz SUN Ultra10 workstation with 512 MB of main memory. We used UC Berkeley SIS SAT solver and the BDD package for representing local don't care sets and various subsets. No additional ATPG-related optimizations were applied.

Experiments were conducted on MCNC benchmarks and arithmetic circuits (adders, multipliers, ALU and dividers).

As no comparison could be done with the other work, due to the differences in fault models, the proposed redundancy identification schemes were compared with respect to their running times and the performance. The exact redundancy identification finds all redundant errors for the benchmarks considered. Also, 1-cube distance approximation performed almost as well. We report the fault coverage with AT test vectors generated from the five top lattice layers.

The number of possible gate replacement faults is very large, as many gates can substitute the correct ones. The fault list size is significant impediment to the overall algorithm execution time. To investigate the reduction in the fault list, and to better expose the performance of the proposed methods, we resort to *worst-case* modeling that allows us to discard many easily detected faults.

Definition 5: *We say that the gate h is the minimum distance (M.d) replacement*

$$\min_{h \in L} \; d\left(g^{ON}, h^{ON}\right)$$

among all replacement candidates in set L. *If the DC sets are available, then the gates that have the smallest positive Hamming distance:*

$$\min_{h \in L} \; d\left(g^{ON} \cap Care_{local}, h^{ON} \cap Care_{local}\right) > 0$$

are used as the minimum distance replacements.

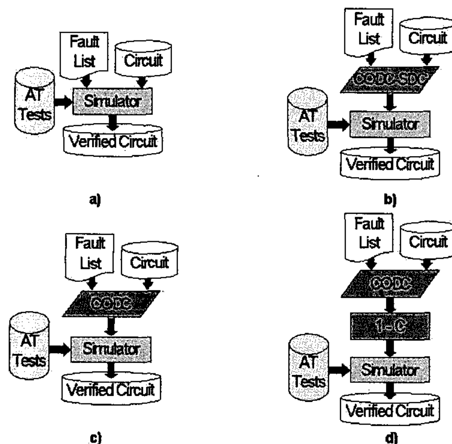

Figure 7: Approximate DC Methods Used in Experiments

Several variations of the approximate redundancy identification were considered, as shown in Figure 7, to assess the efficiencies of the parts of Algorithm 1. They differ in the amount of DC subsets considered, and in the application/omission of the 1-Cube distance s-a-v identification.

In Table 4 we compare the fault coverage of typical gate replacement design errors obtained through AT test vectors and four methods from Figure 7. The columns labeled as "Red" refer to fault coverage with no redundancy removal (as in Figure 7.a), while "CODC-SDC" columns show the coverage after removing redundancies using DC approximations that do not include the local SDC sets, as explained in Section 3.1.1, (Figure 7.b). The next two columns contain the coverage with the SDCs included (Figure 7.c), followed by the 1-Cube distance approximation (Figure 7.d). We present the cumulative results (*All*) of simulations where each gate in the network was replaced with all possible elements from the gate library. The effectiveness of redundancy identifications is compared better by the results obtained using the worst-case modeling (*M.d*). Note that Table 4 demonstrates that the DC approximate methods are in relation, as in Figure 1. Further, for all the cases, the 1-Cube (1-C) approximation identifies all the redundant faults, i.e., performs as well as the exact identification. The fault coverage is less than 100% in some cases only due to the AT test vectors.

Circuit	Size	Red		CODC-SDC		CODC		1-C
		All	M.d	all	M.d	all	M.d	
Look-ahead Adder	12	93.0	84.7	95.0	85.2	96.1	88.5	96.7
	16	93.8	85.2	95.3	86.1	97.4	88.6	96.8
	24	94.1	86.1	95.8	87.0	98.3	89.0	96.8
ALU	10	99.1	89.6	99.2	92.7	99.2	94.5	95.1
	12	99.2	98.2	99.3	92.7	99.3	94.2	94.7
CLA Divid.	9x5	87.0	76.9	97.6	96.0	100	100	100
	11x6	88.2	95.0	97.4	96.1	100	100	100
	13x7	85.8	90.2	98.6	96.3	100	100	100
Array Divid.	9x5	78.7	62.8	96.3	92.6	100	100	100
	11x6	75.9	63.7	98.2	95.0	100	100	100
	13x7	75.5	64.3	97.6	94.9	100	100	100

Table 4: Fault Coverage for Arithmetic Circuits: Impact of Redundant Gate Replacement Removal

In Table 5 we compare times spent on the redundancy identifications using DC approximations with single-cube distance SAT and exact SAT with and without 3 top lattice layer pre-simulations. The exact identifications are illustrated in Figure 8. Also, included are the time and space requirements for DC BDD constructions (second and third column of the table), if the DCs are not readily available in the circuit. We report two time numbers for each benchmark. They allow us to determine bounds on the performance of any possible preprocessing before invoking SATs. Columns "Redund" and "Total" report times spent on redundant and all SAT cases, respectively.

They present the lower and upper bounds on SATs with respect to preprocessing. We don't report separately the time required in different parts of SAT, as overall the routines are fast. The impact of passing the information between the BDD-based DC calculation and the SAT procedures is obtained by comparing the SAT times in these two cases.

From times reported, we see that the approximate identification of DC conditions achieves almost complete coverage in roughly half the time of the exact SAT. Additionally, preprocessing substantially reduces exact SAT running time, as shown in the last two columns of Table 5. Depending on the circumstances, either the exact and approximate redundancy identifications are appealing.

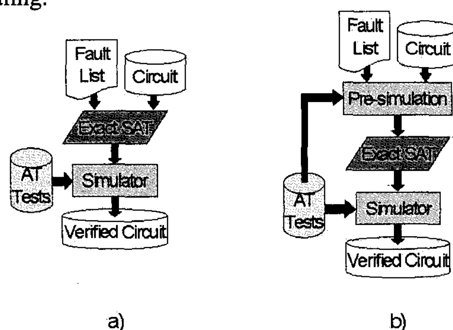

Figure 8: Exact SAT Methods Used in Experiments

Further, as the test set vectors have a very simple structure (top layers of lattice), information on failing to detect the fault was passed to the exact SAT procedure, to speed it up (Figure 8.b). These times are reported last. While BDDs perform worst with respect to their space complexity, preprocessing can reduce the time requirements of SAT. All the improvements in modern SAT solvers, such as those from [14] can be equally applied to both approaches. Any advances in structure-based s-a-v ATPG can be used towards speeding up the approximate methods.

5.1 SAT vs. ATPG

Our approach to redundant faults was based on incorporating relevant DC circuit conditions to speed up the redundancy identification process. The soundness of such a solution is based on the fact that all redundancies in a given circuit are caused entirely by DCs associated with the design and synthesis process.

As "the execution engine" of proposed solutions we selected SAT formulation. However, the proposed approximation algorithms can be equally easily incorporated into a structure-based ATPG scheme. Similarly to the SAT solution, all improvements to the ATPG can be applied to speed up the procedure.

Further optimizations could be applied to SAT implementation of the redundancy identification as well. For example, the true circuit clauses could be constructed only ones, for all the faults injected in a given circuit. Currently, they are re-created for every fault.

6. CONCLUSIONS AND FUTURE WORK

In this paper we considered verification by test vectors. Under a fault model that includes gate and wire

Circuit	DC BDD		DCs and s-a-v SAT			Exact SAT			Pre-sim Exact SAT	
	size	time	Redund [s]	Total [s]	AT Cov. [%]	Redund [s]	Total [s]	AT Cov. [%]	Redund [s]	Total [s]
i1	180	0.113	0.0	0.005	94.4	0.0	0.0	94.4	0.0	0.0
alu2	2187	0.990	0.310	3.590	95.7	0.350	8.120	96.2	0.300	0.830
alu4	2148	2.330	2.47	48.79	95.2	11.190	88.650	95.9	9.690	37.820
9symm	1252	2.364	0.083	0.223	97.5	0.110	0.400	97.5	0.080	0.260
cordic	401	0.520	0.015	0.057	92.3	0.050	0.090	92.8	0.080	0.090
C499	64547	14.381	0.0	1.836	100	0.0	1.460	100	0.0	0.540
C432	173829	15.061	0.0	0.509	100	0.380	0.780	100	0.380	0.380
C17	17	0.015	0.0	0.002	100	0.0	0.002	100	0.0	0.010
C1355	176390	381.205	0.0	0.653	100	4.950	5.600	100	4.820	4.980
C1908	443558	541.553	0.0	0.544	91.2	4.260	5.220	91.2	4.310	4.920
C2670	4401323	540.163	0.101	3.587	98.3	4.360	10.310	98.5	-	-
C6288	∞	∞	-	-	-	88.070	138.660	100	114.100	114.130
C880	30501	13.176	0.069	0.857	97.6	0.320	1.170	97.6	0.290	0.900

Table 5: AT Coverage and Execution Time Comparison among Approximate, Exact and Pre-processing Identifications

replacements, there are many redundant faults. These redundancies seriously impact the verification scheme. Simulations overhead and the confidence in fault coverage could suffer and create an obstacle in the practicality of these verification methods.

We proposed the methods for redundant fault identification. The methods differ in the level of approximation to the problem. While the exact SAT-based solution is practical, we showed that the consideration of replacements that are within a single-cube distance from the replaced gate provides almost complete redundancy identification by the use of standard s-a-v methods. In the latter, we use the CODC subsets of local don't cares to reduce the number of cases considered, and to provide distance information. Further preprocessing to SAT procedures that exploits the properties of the test set is demonstrated. Figure 1 relates the proposed methods. Both approaches can benefit by improvements in underlying SAT and structure-based s-a-v methods. The approximate construction can completely avoid the use of SAT solvers.

We have shown that the use of test sets obtained by Arithmetic Transform decoding results in high coverage vectors for the considered gate and wire replacements.

As the size of the fault list is significant, to simulate larger circuits we would have to rely on reduced fault list by methods such as the worst case modeling (Definition 5), instead of considering of all the cases. More work is planned on establishing the criteria to reducing the fault list, while not sacrificing the accuracy.

The results presented in this work can be extended in future to broader classes of wire replacement models and their redundancy identification. Also, we believe that the sequential and high-level design verifications can benefit from methods presented here.

References

[1] E. J. Aas, K. Klingsheim and T. Steen, "Quantifying Design Quality: A Model and Design Experiments", In *Proc. of EURO-ASIC*, pp.172-177, 1992.

[2] M. S. Abadir, J. Ferguson and T. Kirkland, "Logic Verification via Test Generation", *IEEE Trans. CAD of Integrated Circuits Systems*, 7(1), pp.138-148, Jan.1988.

[3] H. Al-Assad and J. P. Hayes, "Design Verification via Simulation and Automatic Test Pattern Generation", In *Proceedings of International Conference on Computer Aided Design*, pp. 174-180, 1995.

[4] D. Brand, "Verification of Large Synthesized Designs", In *Proc. of International Conference on CAD*, pp. 534-537, 1993.

[5] J. R. Burch and V. Singhal, "Tight Integration of Combinational Verification Methods", In Proc. *of International Conference on Computer Aided Design*, pp. 570-576, 1998.

[6] D. van Campenhout, H. Al-Asaad, J. P. Hayes, T. Mudge and R. B. Brown, "High-Level Design Verification of Microprocessors via Error Modeling", *ACM Transactions on Design Automation of Electronic Systems*, 3(4), pp. 581-599, Oct. 1998.

[7] S.T. Chakradhar. V.D. Agrawal and S.G. Rothweiler, "A Transitive Closure Algorithm for Test Generation", *IEEE Transactions on CAD of Integrated Circuits and Systems*, 12(7), pp. 1015-1028, Jul. 1993.

[8] K.T. Cheng, S. Dey, M. Rodgers and K. Roy, "Test Challenges for Deep Sub-Micron Technologies", In *Proc. of Design Automation Conference*, pp. 142-149, 2000.

[9] K. T. Cheng and A. Krstic, "Current Directions in Automatic Test Pattern Generation", *IEEE Computer*, 32(11), pp.58-64", Nov.1999.

[10] L.H. Goldstein and E.L. Thigen, "SCOAP: Sandia Controllability/Observability Analysis Program", In *Proc. 17th Design Automation Conference*, pp.190-196, 1980.

[11] M.A. Iyer and M. Abramovici, "FIRE: A Fault-Independent Combinational Redundancy Identification Algorithm", *IEEE Trans. VLSI Systems*, 4(2), pp.295-301, Jun.1996.

[12] W.Kunz, and D.Pradhan, "Recursive Learning: A New Implication Technique for Efficient Solutions to CAD Problems - Test, Verification and Optimizations", *IEEE Transactions on CAD of Integrated Circuits and Systems*, 13(9), pp. 1143-1158, Sep.1994.

[13] T. Larrabee, "Test Pattern Generation using Boolean Satisfiability", *IEEE Transactions on CAD of Integrated Circuits and Systems*, 11(1), pp. 4-15, Jan. 1992.

[14] J.P. Marques-Silva and K.A. Sakallah, "GRASP: a search algorithm for prepositional satisfiability", *IEEE Trans. on Computers*, 48(5), pp. 506 -521, May 1999.

[15] P.R. Menon, H. Ahuja, and M. Harihara, "Redundancy Identification and Removal in Combinational Circuits", *IEEE Transactions on CAD of Integrated Circuits and Systems*, 13(5), pp. 646-651, May 1994.

[16] R. Mukherjee, J. Jain, K. Takayama, M. Fujita, J. A. Abraham and D. S. Fussell, "An Efficient Filter-Based Approach for Combinational Verification", *IEEE Transactions on CAD of Integrated Circuits and Systems*, 18(11), pp.1542-1557, Nov. 1999.

[17] K. Radecka and Z. Zilic, "Using Arithmetic Transform for Verification of Datapath Circuits via Error Modeling", *Proc. of VLSI Test Symposium*, pp. 271-277, May 2000.

[18] H. Savoj and R. Brayton, "The Use of Observability and External Don't Cares for the Simplification of Multi-level Logic Networks", In *Proc. of International Conference on Computer Aided Design*, pp. 297-301, 1990.

[19] S. W. Tung and J. Y.Jou, "Verification Pattern Generation for Core-Based Design Using Port Order Fault Model", In *Proc. of Asian Test Symposium*, pp. 402-407, 1998.