Identifying Redundant Wire Replacements for Synthesis and Verification

Katarzyna Radecka and Zeljko Zilic
McGill University, Dept. of ECE
{kasiar,zeljko}@macs.ece.mcgill.ca

ABSTRACT

We propose the redundancy identification of wire replacement faults. The solutions rely on the satisfiability (SAT) formulation of redundancy identification, augmented with the means to effectively use any single stuck-at-value redundancy identification in the approximate schemes. In the latter, we employ the novel use of don't care approximations that detect many redundant faults and quickly identify those that can be detected by stuck-at value identifications. A test generation scheme that uses the error-correcting properties of Arithmetic Transform is incorporated into the overall verification procedure. The test set provides high fault coverage.

1. INTRODUCTION

Recently, there has been a lot of interest in using the automated test pattern generation (ATPG) based schemes in logic optimization, which explore the redundant wires [10]. Further, implementation verification techniques based on simulations, place special emphasis on wire replacement errors in a netlist. The wire error modeling and detection, as well as the redundant error identification, are used in such verifications and logic optimizations. Design errors are often the results of changes introduced manually or even during the automated synthesis [1]. The most common errors of that kind are gate or wire replacements which belong to the design error models proposed in [1] and [2]. A fault model is a necessity when verification is performed by simulations. It has been reported in [2] that by applying modern design flows, 98.9% of all design errors of a common processor, and 94.2% of errors in floating point units are caused by gate or wire replacement errors.

A detecting capability and running time of the simulation-based verification is seriously impaired by redundant faults. Simulations alone cannot deal with that problem, unless applied exhaustively.

In this paper we propose novel procedures for identifying redundant wire replacement faults. Safe approximations to local don't cares are used to identify faults that are either likely to be redundant or detected by standard s-a-v methods. In Section 3 we construct an exact identification of replacement faults using an all-SAT formulation. In addition to identifying redundant wire replacements for the verification purposes, such information can be used to improve the process of circuit optimization. Our approach to redundant wire identification can be applied to such logic optimizations.

Majority of rewiring techniques are based on s-a-v ATPG. However, as the rewiring faults differ from s-a-v faults, and the redundant wire detection requires that both s-a-0 and s-a-1 faults at that wire are redundant, each identification of such a redundant wire requires two runs of ATPG. Additionally, ATPG algorithms are optimized for quick deterministic vector generation, rather than for identifying the faults that are redundant.

Similarly, verification by test vectors relies on full testability of all possible design faults considered. This application also critically depends on the ability to detect redundant faults from a fault model.

2. WIRE REPLACEMENT FAULTS

Wire replacements are used both in synthesis [3], [4], [9], [10] and verification, [1], [2] applications. Many recent advances in synthesis are due to the rewiring approach, where selected wires are chosen for the replacement, in the attempt to find a better implementation for logic at nodes considered. It is recognized [9] that the rewiring techniques critically depend on quick identification of redundant faults.

Rewiring or insertion of design–for-testability (DFT) elements like scan, test points or Built-in Self Test (BIST) are post-synthesis steps, as they are performed on the originally synthesized netlist, and generally affect small areas of the design. They often require human interaction, and can potentially result in errors, such as wrong wire connections, missing/added gate or wire, etc. Hence, there is an obvious need to verify whether such errors were introduced to the final netlist. In the remainder of this paper we discuss the detection of *wire replacement errors* in combinational networks. We are interested only in the errors that do not create cycles, which are easy to detect.

We can classify wire replacements into two categories. The first one deals with errors affecting I/O port connections. I/O ports in this context can be either primary I/O pins, or can represent ports of internal complex blocks (cores) such as adders in the datapath architecture. Detection of these errors is discussed in detail in Section 4. The second category describes errors causing one or more internal wires in the circuit to be wrongly connected to nodes. We distinguish two types of such replacements.

Proceedings of the 15th International Conference on VLSI Design (VLSID'02)
0-7695-1441-3/02 $17.00 © 2002 IEEE

Definition 1*: Single port wire errors (SPWEs) are the replacements affecting a single node in a netlist, such as added/missing wire and permuted wire at the input to a node, Figure 1.a. Internal wire interchange errors (IWIEs) are the replacements affecting multiple nodes in the netlist. IWIE errors include exchange of two nets, where the two single stem nets are swapped to drive the sink node originally driven by the other net, as in Figure 1.b.* ◊

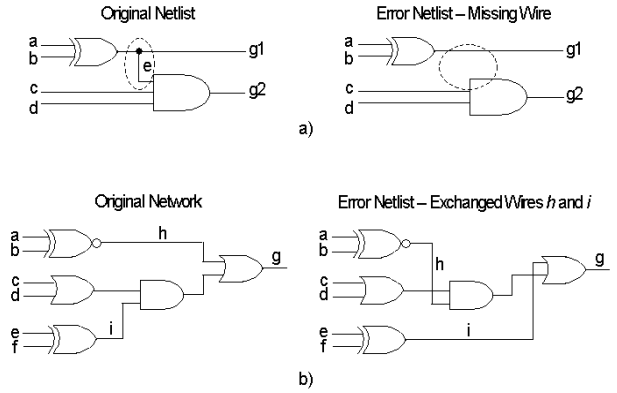

Figure 1: Wire Replacements: SPWE and IWIE

Note that wire replacement errors, unlike gate replacements or s-a-v faults can change the design at different levels of abstraction. Wire replacement errors can affect multiple wires and change functionality of the nodes.

A *redundant replacement* is a substitution that does not change the original function of the circuit. 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 efficiently with such faults, we seek new redundant fault identification methods.

2.1 Redundancy Detection by Don't Cares

Although single port wire errors (SPWEs) and Internal wire interchange errors (IWIEs) affect the wires, the fault effects are observed at nodes to which the erroneous wires fan in. Therefore, the redundancies are caused by don't care (DC) conditions at such nodes. These are either *observability don't care* (ODC) or *controllability don't cares* (CDCs) inhibiting the error detection.

Our first redundant fault identification consists of two steps, both of which can be performed in various degrees of approximation. First, we use the don't care (DC) information in the network to screen out most of the redundant faults. The use of DC subsets guarantees that no irredundant fault will be declared as redundant. For selected remaining faults, in the second step, we apply the modification of single stuck-at-fault redundancy identifications. We employ a method based on the satisfiability (SAT) formulation of the problem. Information on replacements that are to be probed by SAT is provided by DC care sets obtained in the first step.

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

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

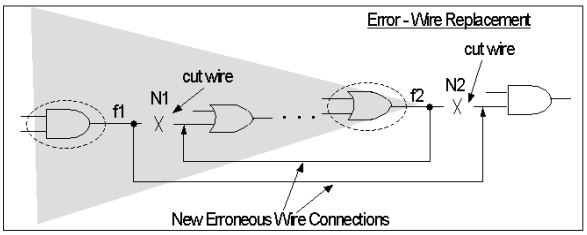

Figure 2: Internal Wire Interchange Error (IWIE)

Lemma 1: *Consider an interchange of two wires, Figure 2. Let these wires be driven by nodes N_1 and N_2, performing functions f_1 and f_2, respectively. By taking into account ON-sets of N_1 and N_2, i.e., f_1^{ON} and f_2^{ON}, and local care sets of N_1 and N_2, $Care_{N1}$ and $Care_{N2}$ (before the exchange) and $Care'_{N1}$ and $Care'_{N2}$ (after the exchange) the replacement is redundant if:*

$$f_1^{ON} \cap Care_{N1} = f_2^{ON} \cap Care'_{N1}$$

and

$$f_1^{ON} \cap Care_{N2} = f_2^{ON} \cap Care'_{N2}.$$

Proof: If the first relation holds, then the net cut at N_1 is redundant, as f_1 at N_1 is being replaced with the equivalent function f_2 at net N_1. Consequently, if both equations are true, then cuts at nets N_1 and N_2, Figure 2, are both redundant, and the wire interchange is redundant. □

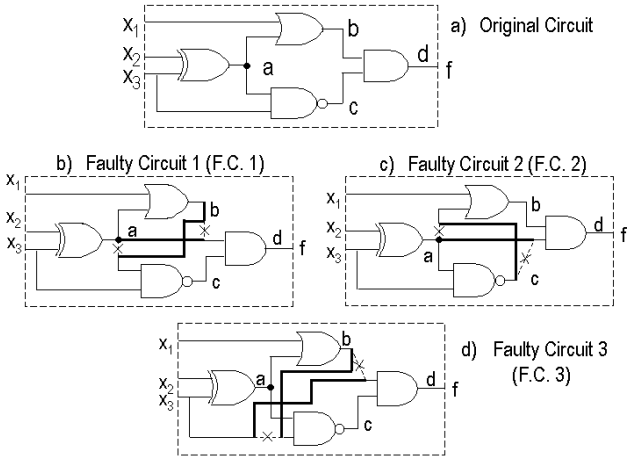

Figure 3: Circuit with IWIEs

Example 1: Consider a circuit and IWIE errors in Figure 3, where nets b and c are replaced with their respective predecessor nets. The local DC sets at nodes b and c consist of observability DCs (ODCs). The original function f and $Care_{local}$ sets are shown in Table 1. When a stem has multiple branches, the additional index is added to the

branch, as with a_1, a_2 and x_{3_2}. The functions at the nets a, b and c are given as well. Upon a replacement, these functions are interchanged, and we aim to detect the difference between the original and replaced functions, according to **Lemma 1**. ♦

Function	$f = x_1 * x_2 * x_3 + x_1 * x_3' + x_2 * x_3'$
$Care_b$	$Care_b = ODC_b' = x_2 + x_3'$
$Care_c$	$Care_c = ODC_c' = x_1 + x_2 x_3' + x_2' x_3$
$Care_{a\_1}$	$Care_{a_1} = x_1'(x_2 + x_3')$
$Care_{a\_2}$	$Care_{a_2} = x_3(x_1 + x_2')$
$Care_{x3\_2}$	$Care_{x_{32}} = x_2 x_3' + x_2' x_3$
f_b	$f_b = x_1 + x_2 * x_3' + x_2' * x_3$
f_c	$f_c = x_2 + x_3'$
f_a	$f_a = x_2 x_3' + x_2 x_3'$

Table 1: $Care_{local}$ Sets for Figure 3

2.1.1 Don't Care Approximations

If we have the exact don't care sets, Lemma 1 provides us with the exact redundancy identification. Practical schemes use only the subset of DCs. ODC space requirements are the biggest, and we approximate them by *compatible observability don't cares* (CODCs) [8]. The advantage of ODC subsets is that the replacements affecting multiple nodes do not change the CODC value. CODCs can hence be used for multiple wire replacements.

Definition 3: *An approximated don't care set (DC_{approx}) associated with a given node is the set of local don't cares where observability don't cares (ODCs) are approximated with compatible observability don't cares (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.* ◊

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

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

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

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:

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

Symbol "⊕" denotes the Boolean difference between two sets, calculated by XOR-ing their characteristic functions. *Care* set represents either $Care_{local}$ or $Care_{approx}$ for determining d_{local} or d_{approx}, respectively.

Example 2: Consider the three faulty circuits (F.C.1 through F.C.3) from Figure 3. Additionally, let W denote the permutation of input wires to OR gate (node b). Table 2 describes the effects of these wire replacement faults. The first row includes the Hamming distances between the correct and faulty gates with no DC set taken into account.

	W	**F.C.1**	**F.C.2**	**F.C.3**
d	0	2	6	4
d_{Care}	0	1, 2	2, 4	2, 3
# vec	0	2	3	4

Table 2: Hamming Distances (d) and Number of Vectors Detecting Error Gates for Circuit in Figure 3

The Hamming distances of the intersection with local care sets, calculated at both affected nodes, are shown in the second row of Table 2. The number of test vectors (the last row) increases with the Hamming distance. Replacement W is redundant, and no vector detects it. ♦

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 is to distinguish these two cases. The distances obtained by the exact and approximate local DC sets are related as follows [7].

Lemma 2: *The following relation is true for the Hamming distances d_{local} and d_{approx}, for a single node replacement between function g and its fault h, calculated with the use of local and approximated DC sets, respectively:*

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

The distance information is used for assessing the likelihood that a fault will be redundant without applying a complete redundancy identification scheme. Further identifications can be parameterized by nonzero distance:

$$d_{Care}(g,h) = d((g^{ON} \cap Care),(h^{ON} \cap Care)) < \varepsilon. \quad (2)$$

The small distance replacements possess the additional properties, useful in detecting redundant replacements. The distance-1 replacements (i.e., $\varepsilon = 2$), can be detected by s-a-v identifications, as the fault will be of one polarity only.

2.2 Using SAT for Redundancy Identification

A satisfiability (SAT) solver is used for our redundancy identifications [5]. To test for a s-a-v fault in a circuit, a *Conjunctive Normal Form* (CNF) is constructed. This product of sums (*clauses*) is equal to one for all solutions; therefore the satisfying assignment is a test vector. If there is no solution, i.e., the fault is redundant, then the CNF expression is unsatisfiable. The SAT formulation for a single s-a-v fault redundancy consists of several types of clauses. *Good circuit clauses* represent the correct operation of a circuit. *Faulty circuit clauses* describe effects of a single s-a-v 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 observation and detection of the fault.

Example 3: The following clauses are generated for a s-a-0 fault at node *b* of the correct circuit in Figure 3.a. The SAT variables are associated with nodes in the network. Those with no subscripts are the good clause variables, such as *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 conditions for which the individual clauses are created are placed in square brackets, Table 3. An assignment $x_1 = 1, x_2 = x_3 = 0$ is a test vector sought. ♦

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	**[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

Table 3: SAT Clauses for Node *b* s-a-0 (Figure 3.a)

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.

2.2.1 Approximate Redundancy Identification

We can derive a SAT formulation for replacements not filtered by don't cares (DCs) using Lemma 2. The distance between the original and the replacement gate, obtained by approximate 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 Equation 2, an efficient SAT formulation can be obtained. We first create an s-a-v SAT instance corresponding to the polarity of the single-value faults, according to Lemma 2. It is sufficient to add to CNF the 1-clauses (clauses with one literal) to restrict gate inputs to a single failing cube.

Example 4: To obtain a SAT instance for replacing the *OR* gate from Figure 3 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: [Distinguishing OR → XOR]: $x_1 a$.♦

The approximate identification algorithm is shown in Algorithm 1. After intersecting the original and replaced gates with the approximate care sets, their Boolean differences (⊕) are obtained. If the differences are empty, then the fault is redundant and is not simulated (line 5). Otherwise, if the difference is a single cube with faults of one polarity, a 1-Cube distance check is performed by s-a-v identifications, augmented with the distinguishing single-cube input assignment (line 12 or 15). If redundancy is not detected, the fault is assumed irredundant and is simulated.

```
1.    Generate CODC Approx. of 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 = = ∅ */
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)}
```

Algorithm 1: Approximate Redundancy Identification

3. EXACT REDUNDANCY IDENTIFICATION

The approximate solution has the advantage of reusing fast stuck-at fault methodologies, while possibly missing some redundant faults. Next, we 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. However, fault site clauses will be augmented by an *activating condition*. The condition for activating a fault where function *g* is replaced by a function *h* is $g \neq h$, regardless of the fault. We hence create an auxiliary node equal to $l = g \oplus h$. Then, the fault location clause will assert $l = 1$. Figure 4 depicts the SAT formulation that is applicable for any faults affecting two nodes in a netlist.

In the case of internal wire interchange errors (IWIEs), when multiple nodes are affected, the same inequality is introduced for each node affected. The conditions at each node are OR-ed, i.e., the additional OR gate is added to assert that the difference in the faulty and fault-free circuits

has to be present in at least one of the nodes affected, as shown in Figure 4.b. Further optimizations are applied for replacements that do not create combinational cycles.

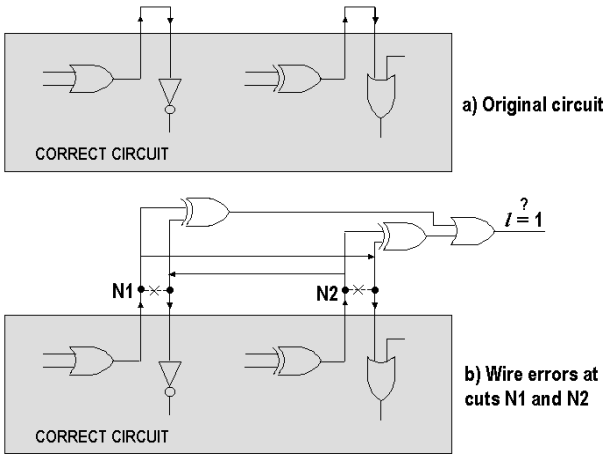

Figure 4: Exact SAT Formulation

Lemma 3: *Consider an internal wire interchange error (IWIE) at nets g and h, applied to acyclic combinational netlist that results in a faulty netlist that is still acyclic and combinational. The activating condition for a fault is to XOR the logic functions g and h of the two wires that are replaced. Then, instead of OR-ing the two XOR differences, considering only one difference is sufficient.*

Proof: If there is no path between g and h, the two differences are the same, and the lemma is proven. Assume without loss of generality that the node g is downstream from node h in acyclic combinational netlist. Then, the replacement at net g, will be activated by condition

$$l_1 = g \oplus h,$$

since the function g is replaced by h. The second replacement, i.e., at net h, will in general affect all the downstream nodes, including g, changing it to a function g_1. Then, the condition for activating this wire replacement would be

$$l_2 = g_1 \oplus h.$$

However, notice that if g has changed, a combinational loop was created, and the circuit is cyclic, contradicting our assumption. Hence, the node g, will not change i.e., $g = g_1$, and conditions at both nets affected will be $g \oplus h$. Then, the overall activation condition is:

$$l_1 + l_2 = (g \oplus h) + (g \oplus h) = (g \oplus h).$$

Therefore, one XOR-ing of the nodes affected is sufficient to describe the activating conditions. □

Figure 5 depicts the application of Lemma 3 to the condition for IWIE faults that result in an acyclic netlist.

Example 5: Consider again faults in Figure 3. For exact SAT formulations of these three faults, the activation condition are created. According to Lemma 3, the original functions at the affected node pairs (the second column in Table 4) are XOR-ed with each other. The resulting conditions are shown in the third column of Table 4.

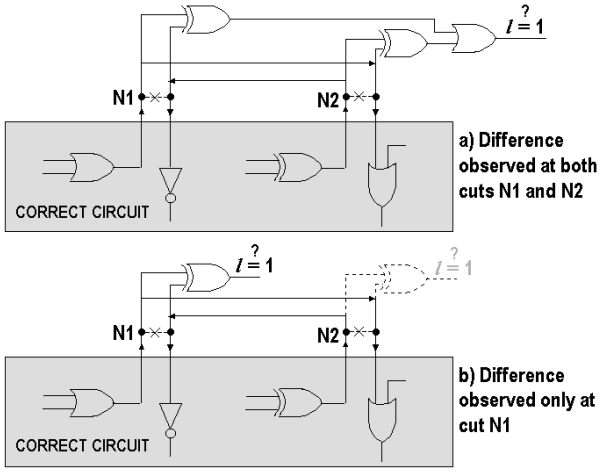

Figure 5: Simplification for Acyclic IWIE Faults

The overall SAT formulation consists of two parts. In addition to the clauses created for the stuck-at-value fault at one of the nodes affected, new clauses are added to describe the activation condition.

Faulty Circuit	Nodes compared	Activation Condition
F. C. 1	a, b	$x_1 \oplus x_2 \oplus x_3$
F. C. 2	a, c	$x'_2 + x_3$
F. C. 3	x_3, b	$x_1 x'_3 + x'_1 x_2 x_3$

Table 4: Activation Conditions for Faults in Figure 3

For example, the SAT formulation for F.C.2 is created by clauses for a (unspecified polarity) fault at node c (similar to those in Table 3), and by additional clauses that assert the activation condition (third column in Table 4):

Additional clauses: $[l = x'_2 + x_3]$:

$$(\bar{l} + \bar{x}_2 + x_3)(x_2 + l)(\bar{x}_3 + l)l.$$

For leaving polarities unspecified, the clauses are removed:

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

4. I/O PORT REPLACEMENT DETECTION

We now investigate conditions for detecting the replacements external to a block. We model such faults by Arithmetic Transform leading to their efficient detection.

4.1 Arithmetic Transform

Arithmetic Transform is a canonical polynomial representation of multi-output Boolean functions $f : B^n \rightarrow B^m$. To describe multi-output functions with a single polynomial, function outputs are "grouped together" into word-level (W) quantities, e.g. integers, resulting in a pseudo-Boolean function $f : B^n \rightarrow W$.

Definition 4: *Arithmetic Transform (AT) of a pseudo-Boolean function* $f : B^n \rightarrow W$ *is a polynomial with*

arithmetic "+" operation, word-level coefficients $c_{i_1 i_2 \cdots i_n}$ *and binary inputs* $x_1, x_2, \ldots, x_n$*:*

$$AT(f) = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \cdots i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \qquad (3)$$

that uniquely and exactly interpolates f. ◊

The transform coefficient vector $C = \{c_{i_1 i_2 \cdots i_n}\}$ is obtained by multiplying the vector f of word-level function outputs with the transform matrix T_n, $C = T_n * f$, where T_n is defined recursively by *arithmetic Davio expansion*:

$$T_n = \begin{bmatrix} T_{n-1} & 0 \\ -T_{n-1} & T_{n-1} \end{bmatrix}, \quad T_0 = 1. \qquad (4)$$

The following theorem, proven in [6], uses the properties of the input space, considered as a Boolean lattice B_n, to derive a test set among the top layers of the lattice. In B_n, vectors belong to one of n+1 lattice layers, depending on the number of ones in a vector - the top lattice layer has n ones, etc.

Theorem 1: *All errors resulting in up to t spectral coefficients can be detected by testing vectors in* $\lceil \log_2(t+1) \rceil - 1$ *top layers of the Boolean lattice.* ♦

This vector generation scheme is sufficient for testing *all* design errors with at most t AT polynomial coefficients. We observed that for common benchmark and arithmetic circuits, high fault detection is obtained using test vectors among the constant number (up to 5) top lattice layers.

4.2 Detection of I/O Port Wire Switching Errors

I/O port wire replacement errors happen, when at least two ports of the design are wrongly connected, see Figure 6. There errors are often caused by either manual intervention or the wrong connection declaration. A typical example is when ports of one block are declared as Little Endian, while ports of the other block are represented as Big Endian. In the case of an I/O port wire replacement error, it is easy to derive its error polynomial.

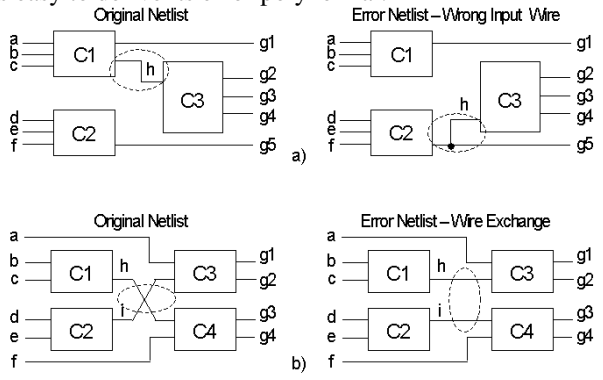

Figure 6: I/O Port Wire Replacement Errors

Definition 5: A single input port wire replacement error (SIPWE) results in the erroneous connection of a single circuit input port to the wrong wire, Figure 6. ◊

Example 6: Let us consider the error caused by interchanging two input wires, k and l, to an N-bit adder, f. AT of the original, fault-free adder is:

$$AT(f) = AT(a+b) = \sum_{i=0}^{N-1} a_i 2^i + \sum_{i=0}^{N-1} b_i 2^i = \sum_{i=0}^{N-1} (a_i + b_i) 2^i.$$

When the error $\tilde{f}$ of interchanging two input port wires k and l $(k < l)$ is introduced, the error polynomial is:

$AT(e) = AT(\tilde{f}) - AT(f) = (a_k + b_l)2^k + (a_l + b_k)2^l$. ♦

The error polynomials are not needed explicitly to obtain the number of lattice layers required for their unique identification. Rather, the lower and upper bounds on the sizes of the possible error polynomials are used. The lower bound (switching only two inputs) is presented in Example 6. The upper bound, i.e. the case of wrongly connecting all the inputs, still results in a small AT.

The same test vector set is applied to the faults within the netlist. It detects not only the faults resulting in t spectral coefficients, but also many faults resulting in larger error spectra. Since the error spectra cannot be simply bounded, unlike I/O wire faults, there is a number of faults that are either hard to detect (and AT decoding vector set will consequently not detect them) or redundant. These faults are handled by our identification schemes.

5. EXPERIMENTAL RESULTS

The redundancy identification schemes have been implemented 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 subsets. The proposed schemes were compared with respect to their running times and the performance. The exact redundancy identification finds all redundant errors for the benchmarks considered. The approximation performed almost as well, as shown in Table 5.

Circuit	DC BDD		DCs	Exact SAT	
	Size	Time[s]	Cov[%]	Time [s]	Cov [%]
alu2	2239	0.97	100	0.34	100
alu4	2849	2.19	100	0.56	100
9symm	2884	0.54	100	0.88	100
cordic	926	0.34	100	0.5	100
C499	73120	11.36	95.6	6.16	96.2
C432	271463	15.52	100	5.4	100
C17	20	0.01	100	0.0	100
C1355	190077	307.22	97.2	18.25	97.6
C1908	120999	459.8	90.1	25.25	90.4
C6288	∞	∞	-	49.27	94.9
C880	36406	10.96	91.0	1.39	97.0
My_adder	4253	0.18	100	0.03	100

Table 5: AT Coverage and Exacution Time Comparison between Approximate and Exact SAT

The number of possible wire replacement faults is very large. To better expose the performance of the proposed methods, we resort to modeling that allows us to discard many easily detected faults and concentrate on those that are more likely to be redundant.

Definition 6: *We say that the wire interchange of two wires w and e, driven by nodes N_1 and N_2, performing functions f_1 and f_2, respectively, is the true fan in acyclic (W.tfi) replacement if*

1. $w_1 \neq e$ *and* $f_1 \neq f_2$
2. N_1 *is in the true fanin of* N_2.
3. *network is acyclic after the interchange.* ◊

This definition regards the replacements as in Figure 2, where the wire is substituted with a wire from its fanin input cone, as shown by the shaded area.

While DCs perform worst with respect to their space complexity, preprocessing can reduce the time requirements of SAT. Table 6 and Table 7 show the fault coverage of SIPWE faults for MCNC benchmarks and arithmetic circuits. In the latter table, vectors among 4 layers were used.

Circuit	# Inputs	Fault Coverage [%]		
		3 layers	4 layers	5 layers
C1355	41	71.8	100	100
C17	5	100	100	100
C1908	33	80.1	85.2	99.1
C3540	50	96.7	99.2	99.2
C432	36	100	100	100
C499	41	71.8	100	100
C6288	32	100	100	100
C880	60	85.0	85.0	97.1
alu2	10	100	100	100
alu4	14	100	100	100
apex7	49	89.4	95.2	99.8
count	35	79.8	99.9	99.9
My_adder	33	96.6	96.6	96.6

Table 6: SIPWE Coverage Experimental Results

6. CONCLUSIONS

In this paper, we proposed the methods for redundant wire replacement identifications. The methods differ in the level of approximation. While the exact SAT-based solution is practical, we have shown 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. Both approaches can benefit by improvements in underlying SAT and structure-based s-a-v methods. The approximate method 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 wire replacements.

Circuit	Size	DC	SAT
Ripple Adder	12	72.8	72.8
	16	72.8	73.4
	24	73.9	77.9
Look-ahead Adder	12	72.0	75.7
	16	68.5	72.1
	24	76.1	79.7
ALU	10	99.7	100
	12	99.7	100
CLA Divider	9x5	100	100
	11x6	94.8	94.9
	13x7	91.4	88.1
Array Divider	9x5	100	100
	11x6	100	100
	13x7	100	100

Table 7: Wire Replacements in Arithmetic Circuits

References

[1] M. S. Abadir, J. Ferguson and T. E. Kirkland, "Logic Verification via Test Generation", *IEEE Transactions on CAD*, 7(1), pp.138-148, Jan.1988.

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

[3] S-C. Chang, L. P.P.P. van Ginneken and M. Marek-Sadowska, "Circuit Optimization by Rewiring", *IEEE Transactions on Computers*, 49(9), pp. 962-970, 1999.

[4] L. Entrena and K-T. Cheng, "Combinational and Sequential Logic Optimization by Redundancy Addition and Removal", *IEEE Transactions on CAD*, 14(7), pp. 909-916, Jul. 1995.

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

[6] K. Radecka and Z. Zilic, "Using Arithmetic Transform in Verification by Error Modeling", In *Proc. IEEE VLSI Test Symposium*, pp. 271-277, 2000.

[7] K. Radecka and Z. Zilic, "Identifying Redundant Gate Replacements in Verification by Error Modeling", to appear in *Proc. International Test Conference*, 2001.

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

[9] C-N. Sze and Y-L. Wu, "Improved Alternative Wiring Scheme Applying Dominator Relationship", *Proc. of ASP-DAC*, pp. 473-478, 2001.

[10] A. Veneris, M. Abadir and I. Ting, "Design Rewiring Based on Diagnosis Techniques", *Proc. ASP-DAC*, pp. 479-484, 2001.