Acceleration Techniques for Dynamic Vector Compaction

Anand Raghunathan
Department of Electrical Engineering
Princeton University, Princeton, NJ 08544

Srimat T. Chakradhar
C & C Research Laboratories
NEC USA, Princeton, NJ 08540

ABSTRACT: We present several techniques for accelerating dynamic vector compaction for combinational and sequential circuits. A key feature of all our techniques is that they significantly improve the computation times without adversely affecting the quality of test sets that can be derived using state-of-the-art compaction methods. Our techniques are based on three key ideas: (1) identification of support sets, (2) target fault switching, and (3) use of dynamic equivalent and untestable fault analysis. All these techniques are useful in significantly reducing the number of faults that have to be considered by a test generator or a fault simulator in a dynamic vector compaction system. For fault simulation, support sets quickly identify a large subset of faults that are guaranteed to be undetectable by a given input sequence. For test generation, support sets identify a large subset of faults that are guaranteed to be undetectable by any extension of a partially specified test sequence. Experimental results on ISCAS 89 benchmark circuits and large production VLSI circuits are included. For full scan designs, our acceleration techniques reduce the overall computation times by a factor of 2 to 3 without adversely affecting the quality (size) of the computed test sets or their fault coverages. The improvement factors obtained are higher for larger circuits. The acceleration techniques enabled the computation of compact test sets for large production circuits that the base test generation system was unable to process in more than 2 CPU days on a Silicon Graphics MIPS 4400 workstation. Results for sequential circuits also show that our acceleration techniques significantly improve the computation times for dynamic vector compaction.

1. INTRODUCTION

Reduction in test application time and test set size is highly desirable to reduce the overall costs incurred in fabricating and testing a large number of chips that implement a specific design. Several combinational test generators aimed at generating small test sets for the stuck-at fault model have been developed. These methods can be classified as static or dynamic. Static methods attempt to reduce the number of vectors in a given test set [1, 2]. Dynamic methods consider vector compaction during the generation of the test set [3, 4, 5]. Vector compaction and test cycle reduction for sequential circuits is significantly more difficult than for combinational circuits. For sequential circuits, static methods have been proposed to reduce the size of the test set or test application cycles [6, 7]. Dynamic methods for reducing test application cycles in partial scan design circuits have been suggested [8, 9]. However, very few dynamic compaction methods for sequential circuits have been reported [5, 10].

1.1 Motivation

Test generators that use dynamic or static compaction techniques can require significantly more computing resources than those that do not attempt vector compaction. Recently, several new techniques have been proposed that produce highly compact test vector sets [2, 4, 5]. The number of test vectors obtained by these methods is typically close to the lower bound on test set size for most of the full-scan versions of the ISCAS 89 benchmark circuits. However, for the larger circuits in the benchmark set, these techniques can require an order of magnitude higher computation times as compared to the underlying test generator. A similar trend is observed for dynamic compaction in sequential circuits [5].

The increase in computation times is mainly due to the large number of faults that have to be processed by a test generator that also performs vector compaction. Table 1 shows the number of faults processed by a test generator and fault simulator with and without dynamic vector compaction, for the three largest ISCAS '89 benchmark circuits. We profiled the TRAN [11] test generation system to obtain data for the case when no vector compaction is attempted. The BECCS [5] vector compaction system uses the TRAN system as the underlying test generator and fault simulator. The number of faults processed by the test generator and fault simulator are shown in columns *ATG* and *FS*, respectively. For example, consider the circuit $s38584$. When no compaction is attempted, the underlying test generator and fault simulator process 5561 and 744267 faults, respectively. For dynamic compaction, the same test generator (fault simulator) now considers about 40 (75) times more faults. Results in the table show that about 40 to 175 times more faults may have to be processed by the test generator and fault simulator during dynamic compaction.

Table 1: Test generation and fault simulation profiles.

Circuit	TRAN		BECCS		BECCS/TRAN	
	ATG	FS	ATG	FS	ATG	FS
s35932	78	108387	12534	19046042	160	175
s38417	1500	1498700	28177	38231609	18	25
s38584	5561	744267	224678	55971504	40	75

High computation times preclude the use of these methods for large production VLSI circuits. It is possible to reduce the computation times by settling for larger test sets (*i.e.*, trade off test generation time for test set size). The focus of this paper, in contrast, is to explore alternative approaches that *significantly reduce computation times without compromising the quality* of the test sets obtained.

1.2 Overview

Our techniques are devised to exploit three important characteristics of dynamic vector compaction: (1) a large number of test vectors that are fault simulated during dynamic compaction are partially specified, (2) many of the faults considered for extending a current partially specified test vector are usually not detectable by any extension of the test vector, and (3) extension of a test vector is an incremental process. Our acceleration techniques are based on the following ideas: (1) identification of *support sets*, (2) target fault switching, and (3) use of dynamic equivalent and untestable fault analysis. All these techniques are useful in significantly reducing the number of faults that have to be considered by a test generator in a dynamic vector compaction system. Support sets are also useful in reducing the number of faults that have to be considered for fault simulation. Unlike critical path tracing [12] that identifies a subset of faults that are guaranteed to be detectable by a given input sequence, support sets identify a large subset of faults that are guaranteed to be *undetectable* by the input sequence.

A technique to identify faults that are guaranteed to be undetectable by a given input vector has been proposed in [13] for speeding up fault simulation in combinational circuits. They consider fully specified input vectors. Their technique cannot be directly used for fault simulation in a compaction environment since a majority of the vectors that are fault simulated during dynamic vector compaction are only partially specified. We compute support sets for partially specified vectors as well, allowing us to significantly reduce the computation effort required for fault simulation during dynamic vector compaction. We also use support sets to accelerate test generation during dynamic compaction. A theoretical basis is presented for computing support sets for sequential circuits, with applications to fault simulation and test generation during dynamic compaction for sequential circuits.

2. TERMINOLOGY

A *test vector* (or vector) is a set of logic values (0, 1, or X) that are simultaneously applied to the primary inputs of the circuit. A test vector is *fully specified* if all inputs are specified to 0 or 1 (*i.e.*, no input assumes a value of X). Test vectors that contain X's are said to be *partially specified*. A *test sequence* (or sequence) is an *ordered* set of test vectors that detects a target fault. A *test set* is an *unordered* set of test sequences. The size of a test sequence is equal to the number of test vectors in the sequence. The size of a test set is calculated as the number of vectors in all its test sequences. Given a test set and the set of faults it detects, a fault is *essential* if only one test sequence can detect the fault. Essential faults can be identified during fault simulation by dropping a fault from the target fault list only after it has been detected *twice* [4].

3. SUPPORT SETS

Support sets can be used to significantly reduce the number of faults processed by a test generator or fault simulator during dynamic or static compaction. These sets can be computed for both combinational and sequential circuits.

3.1 Concept

Consider a vector v that produces a response of 0 or 1 at a primary output Z of a combinational circuit. The input vector may or may not be fully specified. A *support set* for the primary output Z is any set of signals (including primary inputs) that satisfy all the following conditions:

SS1 All signals in the set assume a logic value 0 or 1.

SS2 The primary output Z is a member of the set.

SS3 The logic value on any signal (except a primary input) in the support set is uniquely determined by values of other signals in the support set.

A support set is *minimal* if no signal in the set can be deleted without violating conditions **SS2** or **SS3**. A *minimum* support set has the least cardinality among all possible support sets. As an example, consider the combinational logic of circuit *s27* that is

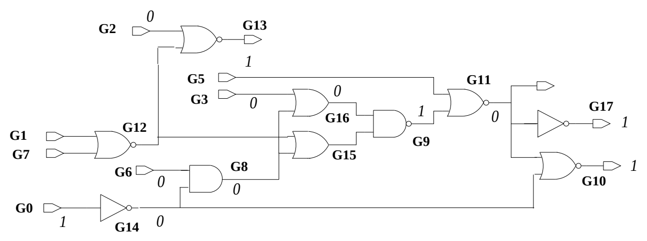

Figure 1: Combinational logic of circuit *s27*.

Table 2: Test vector for fault $G14 \rightarrow G10$ stuck-at 1 in Figure 1.

$G0$	$G1$	$G2$	$G3$	$G5$	$G6$	$G7$
1	x	0	0	1	0	x

shown in Figure 1. This circuit is part of the ISCAS 89 benchmark set. The circuit response to the input vector of Table 2 is also shown in Figure 1. Only signals that assume a logic value of 0 or 1 are marked with their respective values in the figure (the remaining signals assume unknown values). One possible support set for the primary output $G17$ consists of three signals $G5$, $G11$ and $G17$. Since these signals assume known logic values and the set includes $G17$, this set satisfies conditions **SS1** and **SS2**. Also, the value of signal $G11$ is uniquely determined by the value of signal $G5$ and the value of primary output $G17$ is uniquely determined by the value of signal $G11$. Therefore, this set also satisfies condition **SS3**. The support set for primary output $G17$ is also minimal. This is because if we delete signal $G5$ ($G11$) from the set, then the value of signal $G11$ ($G17$) is not uniquely determined by value of other signals in the support set and we violate condition **SS3**. If we delete $G17$, we violate condition **SS2**. Therefore, no signal can be deleted from the support set. Note that a primary output can have several support sets. For example, two possible support sets for primary output $G11$ are as follows: $\{G5, G11\}$ or $\{G0, G14, G8, G3, G16, G9, G11\}$. In the case of multiple primary outputs, condition **SS2** is modified to require that each of the primary outputs be included in the support set. As an example, consider again the circuit of Figure 1. The support sets for primary outputs $G11$, $G17$ and $G10$ are $\{G5, G11\}$, $\{G5, G11, G17\}$, and $\{G0, G5, G11, G14, G10\}$, respectively. Therefore, the support set for the circuit is $\{G0, G5, G10, G11, G14, G17\}$. Again, this set is minimal. This is because, if we delete any signal then it can be shown that we violate one or more of conditions **SS2** and **SS3**.

It is desirable to compute a support set with small cardinality as this leads to greater savings in test generation and fault simulation time (see Section 3.2). However, attempting to compute the minimum support set for each input vector is computationally expensive. This overhead can more than offset the savings that can be obtained by using the support set. Our method to efficiently compute a minimal support set is shown in Figure 2, as procedure COMPUTE_SUPPORT_SET(). This procedure takes a list of primary outputs L with known logic values. We assume that the input vector has been simulated to determine the response of the circuit. We also assume that every gate in the circuit has been assigned an integer value (*level*) that is one more than the maximum level of any fanin of the gate. All primary inputs are at level 0. The *support signals* for a gate are the smallest subset of the gate's inputs that are required to uniquely determine the logic value of the gate. For example, consider an AND gate a that has two inputs b and c. Suppose that signals b and c assume the logic value 0 and 1, respectively. The support signal for gate a is signal b since the value on gate a is uniquely determined by signal b.

This procedure uses an array $Larr$ of size equal to the maximum level ($Maxlevel$) assigned to any signal in the circuit. $Larr[i]$ is

```
Procedure COMPUTE_SUPPORT_SET (primary output list $L$)
{
    support set $S \leftarrow \phi$.
    for (every output in list $L$){
        Add primary output to set $S$.
        Insert output with level $i$ into list $Larr[i]$.
    }
    for (every list $l$ from $Larr[Maxlevel]$ to $Larr[1]$){
        for (each gate $g$ in list $l$){
            Determine support signals for gate $g$.
            Add support signals to $S$ and
            to the appropriate list in $Larr$.
        }
    }
    return support set $S$.
}
```

Figure 2: Procedure to compute minimal support sets.

a list that contains a subset of the set of signals that are assigned the level i. Initially, all lists in $Larr$ are empty. We illustrate the execution of the procedure by an example. Consider the computation of support set for primary output $G10$ in Figure 1. For this example, signals $G10$, $G11$, $G5$, $G14$ and $G0$ are assigned levels 6, 5, 0, 1 and 0, respectively. Here, $Maxlevel$ is 6. We begin by including $G10$ in the support set S. Since $G10$ is at level 6 we include it in the list $Larr[6]$. We first process the signal $G10$ in the list $Larr[6]$. The support signals for gate $G10$ are $G11$ and $G14$. This is because both signals are necessary to uniquely determine the value of $G11$. We include $G11$ and $G14$ in the support set S. Signals $G11$ and $G14$ are also included in lists $Larr[5]$ and $Larr[1]$, respectively. We then process signals in list $Larr[5]$. Signal $G11$ can have either $G5$ or $G9$ as its support signal. This is because the value of gate $G11$ can be uniquely determined by either of its inputs. We make a choice of the support signal as follows. If one of the support signals of the gate has already been included in the support set of the circuit, then this signal is selected as the support signal for the gate. Otherwise, we choose a support signal at the lowest level. In this example, we select $G5$ as the support signal since it is at a lower level than $G9$. This heuristic helps in generating support sets with few signals. We move on to the signal $G14$ that is in the list $Larr[1]$. The support signal for $G14$ is $G0$. We include $G0$ in the support set S. Since all lists in the array $Larr$ are now empty, we terminate the procedure. The set S forms a minimal support set for the primary output $G10$.

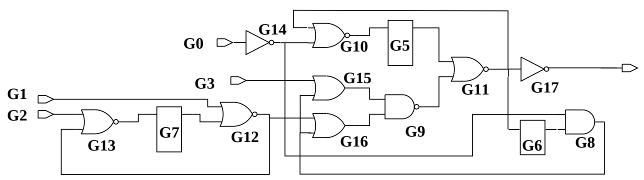

Figure 3: Circuit *s27* of the ISCAS 89 benchmark set.

Table 3: Test sequence for fault $G5$ stuck-at 0 in Figure 3.

Sequence	$G0$	$G1$	$G2$	$G3$
1	1	x	1	0
	x	0	x	1

Support sets can also be computed for sequential circuits. As an example, consider the input sequence shown in Table 3. This sequence has two input vectors and it is applied to the sequential circuit *s27* that is shown in Figure 3. We compute support sets for the combinational logic for each input vector, and use these sets to derive the support set for the sequence. For the first vector, we simulate the circuit with all state variables in an unknown state. The logic values assumed by signals for the first vector are shown in Figure 4. The first vector initializes all state variables. The logic values assumed by signals for the second vector are shown in Figure 5.

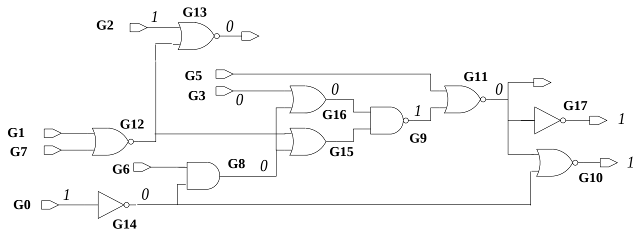

Figure 4: Circuit response for the first vector in Table 3.

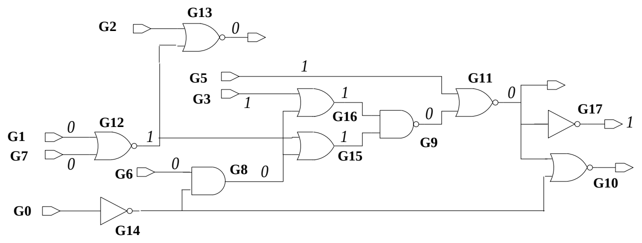

Figure 5: Circuit response for the second vector in Table 3.

For computing support sets, we consider the vectors in reverse order. We first compute a support set for the last vector of the sequence, and then proceed backwards until we reach the first vector of the sequence. As an example, consider the second vector in the sequence shown in Table 3. The circuit response for this vector is shown in Figure 5. The support set S_2 for this vector is computed by using the procedure COMPUTE_SUPPORT_SET (P_2). Here, the list P_2 consists of only the primary output signal $G17$. This is because for the last vector of any sequence, the support set is computed by only considering primary outputs that assume a logic value of 0 or 1. Next state variables with known logic values are not considered in computing the support set for the last vector. The support set S_2 consists of the signals $G17$, $G11$, and $G5$. Note that the cardinality of the support set is significantly smaller than the total number of signals in the circuit.

After computing the support set S_2, we move on to the the first vector. The circuit response for this vector is shown in Figure 4. The support set S_1 for this vector is computed using procedure COMPUTE_SUPPORT_SET (P_1). The list P_1 includes all primary outputs that have a logic value of 0 and 1. In addition, P_1 also includes any next state variable v that satisfies the following two conditions:

C1 Next state variable v is at a logic value of 0 or 1.

C2 The present state variable corresponding to v is included in the support set computed for vector 2.

Since the support set of vector 1 is computed after we process vector 2, it is easy to verify if a next state variable satisfies the condition **C2**. For the current example, the primary output $G17$ is included in P_1 since it assumes a logic value of 1. Also, all next state variables $G10$, $G11$, and $G13$ assume known logic values. Hence, condition **C1** is satisfied for all these signals. As shown in Figure 3, the present state variables corresponding to $G10$, $G11$, and $G13$ are $G5$, $G6$, and $G7$ respectively. Of these, only $G5$ is in the support set S_2. Hence, only signal $G10$ satisfies condition **C2**. We include $G10$ in the list P_1. The support set S_1 as computed by procedure COMPUTE_SUPPORT_SET (P_1) consists of the following signals: $G10$, $G17$, $G11$, $G9$, $G16$, $G8$, $G14$, $G3$, and $G0$. Although not applicable for the present example, if the input sequence consists of many vectors, the last vector is processed similar to the second vector in the present example. All other vectors are processed using a method to similar to the one used for the first vector.

Dynamic vector compaction methods incrementally extend a test sequence. This can be exploited to efficiently compute support sets in an incremental way as explained below. When new primary outputs are set as a result of an extension of a test sequence, the procedure COMPUTE_SUPPORT_SET (P) is called with the list P containing only those primary outputs that have been set to a logic value of 0 or 1 as a result of the last extension. Thus, the added overhead for support set computation is minimal.

3.2 Applications

Fault simulation:

Fault simulation is used in test generators to reduce the number of faults for which test sequences have to be derived by the test generator. For static and dynamic compaction methods, fault simulation is also used extensively to compute essential faults [4]. In dynamic compaction methods, a partially specified test sequence may be extended several times by the test generator and fault simulation is performed for every extension of the test sequence. As shown in Section 1.1, the number of faults processed during dynamic compaction can be very large. We show that support sets can significantly reduce the number of faults that have to be considered for fault simulation. Theorem 1, that is adapted from [13], provides the theoretical basis for identifying a large set of faults that are guaranteed to be undetectable by a given test vector. This theorem applies to combinational circuits.

Theorem 1: *Consider a combinational circuit C and a test vector T. Let S be the support set computed for the primary outputs with* 0 *or* 1 *logic value. If signal g is not in the support set S, then a stuck-at* 0 *or stuck-at* 1 *fault on signal g cannot be detected by vector T.*

From Theorem 1, faults on signals that are not in the support set don't have to be fault simulated for T or any of its extensions. A fault on the fanout branch of signal a to signal b need not be considered if either a or b is not in the support set S. Consider again the circuit shown in Figure 1. The support set is computed for the primary outputs $G10$, $G11$, and $G17$. This is because they assume a logic value of 0 or 1. The support set S consists of the signals, $G10$, $G11$, $G17$, $G5$, $G14$, and $G0$. Only faults on these signals need to be considered for fault simulation. For the circuit shown Figure 1, the collapsed fault list consists of 32 faults. Of these, it turns out that only 13 faults have to be considered for fault simulation.

Support sets can also be used for reducing the number of faults considered in dynamic compaction for sequential circuits. Theorem 2 provides the theoretical basis for sequential circuits. All proofs have been omitted due to lack of space and can be found in [14].

Theorem 2: *Consider a sequential circuit C and a test sequence T that consists of n vectors, $T_1 ... T_n$. Let $S_1 ... S_n$ be the support sets for the circuit corresponding to vectors $T_1 ... T_n$, computed as explained in Section 3.1. Let S be the set formed by taking the set union of $S_1 ... S_n$. If signal g is not in the support set S, then a stuck-at* 0 *or stuck-at* 1 *fault on signal g cannot be detected by the sequence T.*

As an illustration of Theorem 2, consider again the sequential circuit *s27* shown in Figure 3 and the sequence given in Table 3. The circuit response to the first and second vectors of the sequence are shown in Figures 4 and 5 respectively. The support sets S_1 and S_2 for the first and second vectors were previously computed in Section 3.1. The set S formed by taking the union of sets S_1 and S_2 consists of signals $G10$, $G17$, $G11$, $G9$, $G16$, $G8$, $G14$, $G5$, $G3$, and $G0$. Using Theorem 2, it can be shown that we can rule out 9 out of the 32 faults in the collapsed fault list. These 9 faults do not have to be considered in fault simulation. Note that Theorems 1 and 2 do consider the propagation of fault effects along multiple re-convergent paths. This is because, during the computation of the support sets, we add a fanout stem signal to the support set if *any* of its branches have already been included in the support set. In fact, the support sets computed by procedure COMPUTE_SUPPORT_SET() are pessimistic in the sense that faults on signals in the support set may be undetectable.

Test generation:

The main loop in dynamic compaction methods consists of the following two steps: (1) generate a test sequence for a target fault, and (2) extend the test sequence by suitably specifying unspecified primary inputs in the test sequence. By suitably assigning values to unspecified primary inputs, several other faults can be detected by the test sequence. The fault considered in the first step is referred to as the *primary* target fault, while faults that are considered in the second step are called *secondary* target faults (or secondary faults). As shown in Section 1.1, the number of secondary faults can be very large. Support sets are useful in significantly reducing the number of secondary faults. In order to use support sets for secondary fault elimination, we generalize the concept of support sets to cover signals that assume logic values X. This extension is necessary because faults on signals that are at a logic value of X may be detectable by a suitable extension of the test vector. The definition of extended support sets is given below.

SS1 All signals in the set assume a logic value 0, 1 or X.

SS2 The primary output Z is a member of the support set.

SS3 The logic value on any signal (except a primary input) in the support set is uniquely determined by logic values of other signals in the support set.

The procedure COMPUTE_SUPPORT_SET (P) described in Section 3.1 can still be used with one difference - the list P contains all primary outputs. The definition of support signals for a gate is also extended as follows. For a gate g that is at a logic value of X, all inputs of the gate are considered to be support signals. All references to support sets in this subsection are to the extended definition given above.

Our next result shows how to use support sets for reducing the test generation effort involved in dynamic vector compaction. Theorem 3 identifies a large subset of faults in the circuit that do not have be considered as secondary faults while extending a given partial test vector.

Theorem 3: *Consider a combinational circuit C and a test vector T. Let S be the extended support set of the circuit. If signal g is* *not in the support set S, then a stuck-at 0 or stuck-at 1 fault on signal g cannot be detected by any extension of vector T.*

As an example, consider again the circuit of Figure 1. The target (collapsed) fault list consists of 32 faults. Consider the situation when the input vector of Table 2 is extended to detect other faults. The logic values assumed by signals for the input vector of Table 2 are shown in Figure 1. The support set of the circuit, as computed in Section 3.1, consists of the signals $G10$, $G11$, $G17$, $G5$, $G14$ and $G0$. If we also consider primary output signal $G13$ (whose value is X), then signals $G13$, $G12$, $G7$, $G2$ and $G1$ are also added to the support set. Only faults on these signals need be considered as secondary faults. For example, fault $G8$ stuck-at 1 need not be considered as a secondary target fault for test generation. For this circuit, it turns out that 11 of the 32 faults in the target fault list do not have to be considered as secondary faults in trying to extend the input vector of Table 2.

Support sets are also useful for reducing the number of secondary faults in sequential circuits. Theorem 4 provides the basis for secondary fault elimination in sequential circuits. Consider a sequential circuit C and a test sequence T that consists of n vectors, $T_1 ... T_n$. Let $S_1 ... S_n$ be the extended support sets for the circuit corresponding to vectors $T_1 ... T_n$, computed as explained below. The computation of the support sets $S_1 ... S_n$ is similar to the computation illustrated in Section 3.1, with some important differences. As before, we proceed in the reverse order of the vectors, *i.e.*, we generate $S_n ... S_1$ in that order. For generating S_n, the procedure COMPUTE_SUPPORT_SET (P_n) is used where P_n contains all the primary output signals, irrespective of their logic values. S_i ($0 \leq i \leq n-1$) is computed by calling the procedure COMPUTE_SUPPORT_SET (P_i), where P_i consists of both primary outputs and next state variables. All primary outputs are included in P_i irrespective of their logic value. Each next state variable v that satisfies the following condition is included in P_i.

C1 The present state variable corresponding to v is included in the support set S_{i+1} computed for vector T_{i+1}.

Theorem 4: *Consider a sequential circuit C and a test sequence T that consists of n vectors, $T_1 ... T_n$. Let $S_1 ... S_n$ be the extended support sets for the circuit corresponding to vectors $T_1 ... T_n$. Let S be the set formed by taking the set union of $S_1 ... S_n$. If signal g is not in the support set S, then a stuck-at 0 or stuck-at 1 fault on signal g cannot be detected by any extension of the sequence T.*

For example, consider again the circuit responses of circuit *s27*, that are shown in Figures 4 and 5. The support set S_2 corresponding to the second vector (which is the last vector of the sequence) is computed for only the primary output signal $G17$, and consists of signals $G17$, $G11$, and $G5$. We next proceed to compute S_1, the support set corresponding to the first vector of the sequence. Next state variables $G10$, $G11$, and $G13$ correspond to present state variables $G5$, $G6$, and $G7$, respectively. Only $G5$ is included in S_2. Hence S_1 is computed for the primary output signal, $G17$, and the next state variable $G10$. The set S_1 consists of the signals $G17$, $G10$, $G11$, $G9$, $G16$, $G8$, $G14$, $G3$, and $G0$. The set S is the set union of S_1 and S_2. Only faults on signals in S are considered as secondary faults. For example, fault $G2$ stuck-at 0 need not be considered as a secondary fault since Theorem 4 rules out the possibility of detecting it by extending the partially specified sequence of Table 3. Note that if the first vector alone is considered, it can be extended so that the fault effect reaches the next state variable $G13$. However, this fault effect cannot be propagated to the primary output $G17$ by any extension of the second vector. For this circuit, the target fault list has 32 faults. After applying Theorem 4, it can be verified that 12 of the 32 target faults cannot be detected by any extension of the given sequence.

Table 4: Vector used for illustrating dynamic fault equivalence.

$G0$	$G1$	$G2$	$G3$	$G5$	$G6$	$G7$
0	0	0	0	0	x	x

4. DYNAMIC FAULT EQUIVALENCE

Two faults, $f1$ and $f2$ are *equivalent* if any test sequence that detects $f1$ also detects $f2$ and vice-versa. By computing such equivalent faults, we can reduce the number of faults that have to be considered for test generation and fault simulation. If a partially specified vector has to be fault simulated or further extended (as is the case in dynamic compaction), the partially specified values in the circuit may lead to new equivalent faults. To see this, consider the circuit of Figure 6. This figure also shows the signal values for the input vector given in Table 4. Consider faults $G7$ stuck-at 0 and $G12$ stuck-at 1. These faults are not equivalent. However, since $G1$ assumes a logic 0, these two faults now become equivalent. This is because any test for $G12$ stuck-at 1 that is derived by extending the partial vector of Table 4 must set $G7$ to a 1 and propagate the fault effect from $G12$ to a primary output. These are also the conditions for any test vector for $G7$ stuck-at 1. Therefore, the faults $G7$ stuck-at 0 and $G12$ stuck-at 1 are *dynamically equivalent*. Given a partially specified input vector, the target fault list can be further collapsed based on signals that assume known values. This situation occurs often in dynamic compaction. For example, the initial collapsed fault list for the circuit shown in Figure 6 consists of 32 faults. Given the vector of Table 4, we can compute additional equivalent faults. For this example, we can collapse the fault list to contain only 27 faults. Dynamic equivalences contribute to a reduction in the number of faults considered for both test generation and fault simulation.

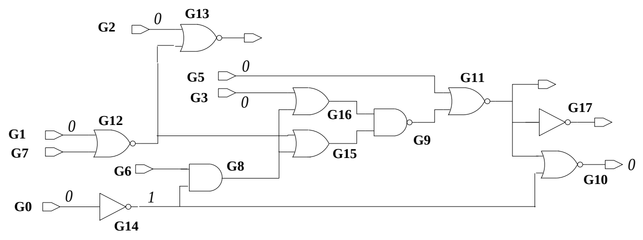

Figure 6: Circuit response for the vector in Table 4.

5. UNTESTABILITY ANALYSIS USING X-PATH CHECK

The concept of X-path [15] check can be used to further reduce the number of secondary faults processed. Consider a combinational circuit C, a test vector T that may be partially specified, and a candidate fault g stuck-at v (g is a signal in the circuit and v is either 0 or 1). The X-path check assumes a knowledge of the good circuit response to T. The X-path check starts from the fault site, g, and searches for a path P to a primary output such that for each gate g' on the path, one of the following conditions is true:

Either The output of g' is at a logic value of X in the good circuit.

Or Each side input s of g for which there is no path from the fault site to s has either a non-controlling value or a value of X.

The X-path check was used in PODEM [15] for guiding the branch and bound process during test generation. We use it in a different context, to efficiently identify faults that cannot be detected by any extension of a given partial vector. Knowledge of the good circuit response to the vector T is a pre-requisite for the X-path check. However, this does not impose any overhead in practice as these values are readily available from the fault simulation performed for T before the next extension of T is attempted. It is important to note that the presence of an X-path is not sufficient to guarantee that a test can be derived for the fault by extending the partial vector T. However, failure of the X-path check does imply that it is futile to attempt an extension of T to detect the fault being considered. This significantly reduces the number of unnecessary test generation attempts, thus making the compaction procedure more efficient.

6. TARGET FAULT SWITCHING

If the target fault list has an untestable fault f, then f may be considered during the extension of several test sequences before it is proven untestable. This happens because a failure to generate a test for f by extending a partially specified sequence T_1 does not say anything about the testability of f for another partial sequence T_2. In order to prove f as untestable, it is necessary to consider f as a primary target fault. Thus, a fault could be considered several times as a secondary target fault before it is detected or proven undetectable by considering it as a primary target fault. This contributes to an increase in the number of faults that are processed by the test generator. We use a simple heuristic to avoid repeated test generation attempts for such faults. If f is proven to be untestable by any extension of the current test sequence, then we consider f as the next primary target fault. It is important to note that since target fault switching occasionally selects a different primary target fault than would have been used otherwise, it nominally alters the fault ordering used by the dynamic compaction system. As a result, a potentially different test set may result when target fault switching is used. However, the change in the fault ordering thus caused is limited to very few faults, and does not affect the test set size significantly because most of the faults selected by this heuristic turn out to be indeed untestable. Hence, known fault ordering methods [4] can still be used along with target fault switching. In practice, it was observed that the slight re-ordering caused by the use of target fault switching has a negligible impact on the size of the test sets (see Section 7).

The target fault switching strategy is also useful for cases when the fault list has no untestable faults. If the underlying test generator generates tests by considering each primary output separately, then it is useful to establish if a given fault is untestable with respect to a particular primary output. If this is the case, then the fault does not have to be considered for all tests and their extensions that are derived by considering this primary output. We use the following heuristic to quickly identify such faults. If a fault f cannot be detected by any extension of an input vector for a given primary output O, then we consider f as the next primary target fault. The test generator attempts to establish f as untestable with respect to the primary output O. If f is established to be untestable with respect to O, we need not target f as a secondary fault for detection at O for the remaining vectors.

7. EXPERIMENTAL RESULTS

We have currently implemented the following acceleration techniques as part of the BECCS system: (1) use of support sets for fault simulation and secondary fault selection, (2) target fault switching strategy, and (3) X-path check for secondary fault selection for combinational circuits.

Table 6: Improvement in computation times.

Circuit	ATG	FS	Total
s5378	1.50	1.65	1.57
s9234	1.10	1.05	1.08
s13207	1.14	2.44	1.61
s15850	1.29	1.98	1.38
s35932	1.0	2.24	2.14
s38417	1.25	2.8	2.32
s38584	3.13	2.86	2.95

7.1 Full scan designs

We performed two experiments to evaluate the effectiveness of proposed techniques for combinational circuits. For the first experiment, the target fault list includes all faults in the circuit. For the second experiment, only irredundant fault lists were used. These fault lists were obtained by removing faults that were proven to be redundant from the fault lists used in the first experiment. All experiments were performed on a Silicon Graphics Challenge L series machine that uses a MIPS 4400 processor.

To study both the individual and the net effects of the acceleration techniques on dynamic compaction, we considered three cases: (1) BECCS with only X-path check for secondary faults (this is referred to as the *Base* case), (2) BECCS with X-path check and target fault switching strategy (*Base + TFS* case), and (3) BECCS with X-path check, target fault switching strategy and use of support sets for fault simulation (*Base + TFS + Support* case).

Complete fault lists:

Computation times for the full scan versions of the larger ISCAS 89 benchmark circuits are given in Table 5. Test generation time, fault simulation time and total time are reported separately under columns *ATG*, *FS*, and *Total*, respectively. No numbers for fault coverage or test efficiency are reported, because both the base system and BECCS achieved 100% test efficiency (all detectable faults were detected, and all remaining faults were proven to be redundant) for all circuits, in all cases.

Table 6 shows the improvement in test generation, fault simulation and overall computation times obtained by using the proposed acceleration techniques. Column *ATG* (*FS*) shows the ratio of test generation (fault simulation) times for the base system to those for the case when target fault switching strategy and support sets are added to the base system. The improvement in overall computation times is shown in column *Total*. The improvements are computed as the ratio of the *Base* case to the *Base + TFS + Support* case.

Table 7: Test set sizes.

Circuit	Base	Base + TFS	Base + TFS + Support
s5378	117	116	116
s9234	164	155	155
s13207	237	238	238
s15850	103	107	107
s35932	15	14	14
s38417	109	108	108
s38584	133	130	130

Test set sizes are reported in Table 7. There is a marginal improvement in test set sizes over the base system for most circuits. This difference is due to the fact that the target fault switching

Table 5: Computation times for dynamic compaction.

Circuit	Base			Base + TFS			Base + TFS + Support		
	ATG	FS	Total	ATG	FS	Total	ATG	FS	Total
s5378	93.0	90.9	183.9	62.2	84.5	146.7	61.8	55.0	116.8
s9234	634.9	390.7	1025.6	582.5	463.2	1045.7	576.9	372.1	949.0
s13207	485.4	578.7	1064.1	467.0	592.0	1059.0	424.9	236.8	661.7
s15850	3829.3	926.4	4755.7	2928.0	1218.0	4146.0	2975.5	468.1	3443.6
s35932	139.9	3399.6	3539.5	137.6	3740.0	3877.6	139.4	1514.3	1653.7
s38417	1124.8	5642.4	6767.2	934.5	5602.4	6536.9	899.3	2012.6	2911.9
s38584	5021.7	8869.1	13890.8	1626.0	8691.1	10317.1	1603.8	3100.0	4703.8

Table 8: Production VLSI circuits.

Circuit	Inputs	Outputs	Gates	Collapsed Faults
ckt1	336	340	7803	8824
ckt2	551	654	4656	7424
ckt3	134	32	6025	12161
ckt4	1133	1106	31416	42744
ckt5	2131	2304	49623	63703

Table 10: Improvement in computation times (irredundant fault lists).

Circuit	ATG	FS	Total
s5378	1.15	1.43	1.28
s9234	1.12	1.22	1.16
s13207	1.09	2.11	1.52
s15850	2.55	1.89	2.33
s35932	1.03	2.17	2.07
s38417	1.20	2.62	2.21
s38584	2.03	2.76	2.58
Production VLSI Circuits			
ckt1	1.56	1.56	1.56
ckt2	1.08	1.91	1.50
ckt3	0.93	1.49	1.05
ckt4	0.60	3.19	1.84
ckt5	-	-	> 3.00

strategy causes faults to be considered in a slightly different order than the fault ordering in the base system. Note that the third and fourth columns of Table 7 are identical, indicating that test set size is unaffected by support sets.

Results in Tables 5, 6 and 7 are significant for the following reasons: (i) the proposed acceleration techniques can speed up the base system by as much as a factor of 2.95, (ii) the test set sizes produced by the base system are largely unaffected by the acceleration techniques, (iii) the overhead for support set computation and X-path check is negligible and their use results in a significant reduction in overall computation times, and (iv) speedups are higher for larger circuits. If support sets are also used in reducing secondary faults, we can expect a further improvement in computation times.

Irredundant fault lists:

The characteristics of the production VLSI circuits are shown in Table 8. These circuits consist of non-Boolean primitives like tristate buffers, bidirectional buffers (also called as I/O buffers) and bus configurations. We have extended the concept of support sets to circuits with non-Boolean primitives. Column headers of Table 9 are identical to the headers in Table 5. Again, test efficiency numbers are not presented because they are 100% for all circuits in all cases. Although fault lists do not have redundant faults, the target fault switching strategy can still be used, as explained in Section 6.

Table 11: Test set sizes (irredundant fault lists).

Circuit	Base	Base + TFS	Base + TFS + Support
s5378	116	118	118
s9234	168	156	156
s13207	238	238	238
s15850	104	106	106
s35932	13	15	15
s38417	107	109	109
s38584	128	129	129
Production VLSI Circuits			
ckt1	263	265	265
ckt2	126	122	122
ckt3	175	169	169
ckt4	56	57	57
ckt5	-	-	563

Table 9 reports computation times and Table 10 reports the improvements in CPU seconds for full scan versions of ISCAS 89 benchmark circuits and several production VLSI circuits. Again, there is an improvement in performance over the base system. The test generation times improve by a factor of upto 2.5. The reduction in fault simulation times is due to the use of support sets to eliminate undetectable faults. The acceleration techniques result in a system that is about two to three times faster than the base system. For circuit $ckt5$ that has about 50,000 gates, the base system *did not complete* in 2 days. However, using the proposed acceleration techniques, dynamic compaction was achieved in 11.7 hours, with 100% fault coverage. The improvements in computation times are summarized in Table 10.

Test set sizes are reported in Table 11. Again, a marginal fluctuation in test set sizes over the base system is seen. The small change in test set sizes is solely due to target fault switching, since support sets and the X-path check do not have any effect on test set size.

In summary, our results indicate that the proposed acceleration techniques can speed up the base system by two to three times, the test set sizes are largely unaffected by the acceleration techniques, and complete fault coverage was maintained. The overhead for support set computation and the X-path check was negligible and their use resulted in a significant reduction in overall computation times.

7.2 Sequential circuits

We use support sets for secondary fault selection during dynamic compaction for sequential circuits. Table 12 gives the computation times, test set sizes and fault coverages for several circuits. The total CPU time for compaction, the test set size, and the fault coverage are given under columns *CPU*, *Vec*, and *FC* respectively. Circuits whose names start with 'ps' are partial scan circuits that were obtained from the ISCAS 89 benchmarks by breaking all loops except self loops. Circuits whose names start with 'p' represent

Table 9: Computation times for dynamic compaction (irredundant fault lists).

Circuit	Base			Base + TFS			Base + TFS + Support		
	ATG	FS	Total	ATG	FS	Total	ATG	FS	Total
s5378	69.4	80.3	149.7	60.7	83.7	144.4	60.1	56.1	116.2
s9234	409.0	362.4	771.4	372.9	378.6	751.5	366.0	296.0	662.0
s13207	334.2	484.5	818.7	314.8	536.2	851.0	307.2	229.7	536.9
s15850	2462.0	876.2	3338.2	1026.3	1152.8	2179.1	964.9	463.0	1427.9
s35932	144.2	3109.4	3253.6	141.4	4435.1	4576.5	140.4	1429.9	1570.3
s38417	1036.0	5551.0	6587.0	878.6	5669.5	6548.1	864.6	2121.7	2986.3
s38584	1960.2	8288.4	10248.6	974.9	8758.5	9733.4	964.7	3000.6	3965.3
Production VLSI Circuits									
ckt1	1089.0	563.6	1652.6	675.8	669.7	1345.5	696.3	361.5	1057.8
ckt2	151.0	247.0	398.0	139.4	246.3	385.9	140.0	129.5	265.5
ckt3	1420.9	651.2	2072.1	1489.8	767.4	2257.2	1536.3	435.7	1972.0
ckt4	1353.4	6643.4	7997.8	2214.2	6330.7	8544.9	2253.8	2082.5	4336.3
ckt5	-	-	$>$ 2 days	-	-	-	29179.3	13198.3	42377.6

Table 12: Acceleration of dynamic compaction in sequential circuits.

Circuit	Base			Base + Support		
	CPU	Vec	FC	CPU	Vec	FC
s208	27.2	151	63.7	23.8	151	63.7
s344	318.7	127	94.1	175.0	168	94.1
s382	1728.2	1016	81.7	1575.1	1016	81.7
s386	1022.6	354	79.7	393.0	380	79.7
s838	2123.4	169	29.6	1833.0	169	29.6
s1196	1261.2	334	99.7	451.0	332	99.7
s1238	1308.8	332	94.5	513.1	333	94.5
s1423	16417.1	40	19.7	12102.1	35	20.2
ps510	68.2	133	100.0	28.0	137	100.0
ps526	3413.1	2023	82.7	1827.1	1874	82.7
ps526n	3213.9	2089	82.3	1979.0	2089	82.3
ps820	1096.5	349	100.0	147.0	370	100.0
ps832	1156.5	355	98.4	148.0	380	98.4
ps953	6989.6	245	100.0	600.1	262	100.0
ps1488	1623.1	307	87.1	180.0	334	86.8
ps1494	1569.2	315	100.0	181.1	341	100.0
p35932	$>$ 2 days	-	-	19741.4	271	44.44
p38417	$>$ 2 days	-	-	23614.3	415	45.92

pipelined versions of the corresponding ISCAS 89 benchmarks (all loops including self loops were broken through partial scan). Table 12 shows that the use of acceleration techniques improved the speed of the dynamic compaction process by up to a factor of 8.75. Circuits like $p35932$ and $p38417$ that the base compaction system was unable to process in over 2 days were successfully processed by using our acceleration techniques.

8. CONCLUSION

We have presented techniques for accelerating dynamic vector compaction. These techniques can be used for combinational or sequential circuits. They can be integrated into most dynamic or static vector compaction systems. Experimental results on several large production VLSI circuits show that our techniques can accelerate dynamic compaction methods by a factor of two to three. More significantly, the acceleration factors are higher for larger circuits, enabling the generation of compact test sets for large production circuits. The test set quality (size) was unaffected by our methods, and complete fault coverages were maintained for all our experiments. We demonstrated the use of support sets to significantly reduce the number of faults processed during dynamic compaction. Support sets have several other applications, including serial or parallel fault simulation for combinational and sequential circuits. Although not attempted here, our techniques can also be used in static compaction methods that rely extensively on fault simulation [2].

REFERENCES

[1] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Transactions on Computer-Aided Design*, vol. 7, pp. 126–136, January 1988.

[2] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, "On Compacting Test Sets by Addition and Removal of Test Vectors," in *VLSI Test Symposium*, pp. 202–207, April 1994.

[3] P. Goel and B. C. Rosales, "PODEM-X: An Automatic Test Generation System for VLSI Logic Structures," in *Proceedings of the 18th ACM/IEEE Design Automation Conference*, pp. 260–268, June 1981.

[4] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," in *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pp. 102–106, June 1993.

[5] S. T. Chakradhar and A. Raghunathan, "Bottleneck Removal Algorithm for Dynamic Compaction and Test Cycles Reduction," in *Proc. European Design Automation Conf.*, September 1995.

[6] T. M. Niermann, R. K. Roy, J. H. Patel, and J. A. Abraham, "Test Compaction for Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, vol. 11, pp. 260–267, February 1992.

[7] S. P. Morley and R. A. Marlett, "Selectable Length Partial Scan: A Method to Reduce Vector Length," in *Proceedings of the International Test Conference*, pp. 385–392, September 1991.

[8] S. Y. Lee and K. K. Saluja, "Sequential Test Generation with Reduced Test Clocks for Partial Scan Designs," in *VLSI Test Symposium*, pp. 220–225, April 1994.

[9] E. M. Rudnick and J. H. Patel, "A Genetic Approach to Test Application Time Reduction for Full Scan and Partial Scan Circuits," in *Proceedings of the 8th International Conference on VLSI Design*, January 1995.

[10] I. Pomeranz and S. M. Reddy, "On Generating Compact Test Sequences for Synchronous Sequential Circuits," in *Proc. European Design Automation Conf.*, September 1995.

[11] S. T. Chakradhar, V. D. Agrawal, and S. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 1015–1028, July 1993.

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

[13] S. B. Akers, B. Krishnamurthy, S. Park, and A. Swaminathan, "Why is Less Information from Logic Simulation more useful in Fault Simulation?," in *Proceedings of the International Test Conference*, pp. 786–800, 1990.

[14] A. Raghunathan and S. T. Chakradhar, "Acceleration Techniques for Dynamic Compaction," tech. rep., C&C Research Labs, NEC USA, Princeton, NJ, October 1994.

[15] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Transactions on Computers*, vol. C-30, pp. 215–222, March 1981.