

Application of Saluja-Karpovsky Compactors to Test Responses with Many Unknowns*

Janak H. Patel, Steven S. Lumetta
University of Illinois
Dept. of ECE
Urbana, Illinois 61801
{jhpatel,lumetta}@uiuc.edu

Sudhakar M. Reddy
University of Iowa
Dept. of ECE
Iowa City, Iowa
reddy@eng.uiowa.edu

Abstract

This paper addresses the problem of compacting test responses in the presence of unknowns at the input of the compactor by exploiting the capabilities of well-known error detection and correction codes. The technique, called i-Compact, uses Saluja-Karpovsky Space Compactors, but permits detection and location of errors in the presence of unknown logic (X) values with help from the ATE. The advantages of i-Compact are: 1. Small number of output pins from the compactors for a required error detection capability; 2. Small tester memory for storing expected responses; 3. Flexibility of choosing several different combinations of number of X values and number of bit errors for error detection without altering the hardware compactor; 4. Same hardware capable of identifying the line that produced an error in presence of unknowns; 5. Use of non-proprietary codes found in the literature of 1950s; and 6. Independent of the circuit and the test generator.

1 Introduction

Full scan has found wide acceptance as a cost-effective Design For Test (DFT) methodology in circuits with millions or more gates. The scan chain is often broken up into hundreds of smaller chains to reduce the test time through parallel loading and unloading of data. However, a large number of parallel scan chains implies the use of a large number of pins for scan-in and scan-out operations as well as a large number of scan channels on testers. One DFT method—the Illinois Scan Architecture [1999]—simultaneously addresses the problem of pins, test time, and test data volume. However, the data compaction suggested in Illinois Scan is the well-

known multiple-input shift register (MISR), which is most suitable for circuits that do not generate many unknown logic values (X's). To get around the problem of X's, many of the hardware-based solutions use masks to block out the X's, use test point insertion to force X's to certain specific logic values, or read the MISR out before an X arrives and then reset after the X is gone. Most of these techniques require circuit-specific and test-vector-specific hardware modifications. Most recently, Mitra and Kim [2002] addressed this problem and gave an elegant solution, called X-Compact, to handle X's with a compactor that is independent of the circuit and test vectors. They gave a novel coding technique to account for a limited number of X's in the input stream of a compactor while maintaining the desired error detection capability. In this paper, we give an alternative solution to the same problem; our solution does not require the creation of new codes and is also independent of the circuit and the test vectors. Our method, i-Compact, requires comparable tester memory but fewer pins than X-Compact. However, the biggest advantage of i-Compact is that the same compactor can be used for varying degrees of error detection capability in the presence of a varying number of unknowns.

The first use of error coding in the context of test response compaction was proposed by Benowitz [1975] and his colleagues. Other early pioneers of compaction by coding are Frohwerk [1977], who popularized the use of linear feedback shift registers (LFSRs) and Konemann [1979], who popularized the use of MISRs. All of these techniques did compaction of test responses in time and space. Space only compaction using error codes was first proposed by Saluja and Karpovsky [1983]. In these and many other compaction techniques, the problem of compaction in the presence of unknowns was largely handled by ad-hoc techniques of ignoring the unknowns with masks provided by either the tester or the storage on-chip. To the best of our knowledge, Mitra and Kim were the first to provide a com-

*This paper is based in part upon work supported by National Science Foundation grants ACI 99-84492 CAREER and CCR 00-97905 as well as SRC grants 99-TJ-717 and 2001-TJ-949. The content of the information does not necessarily reflect the position or the policy of these organizations.

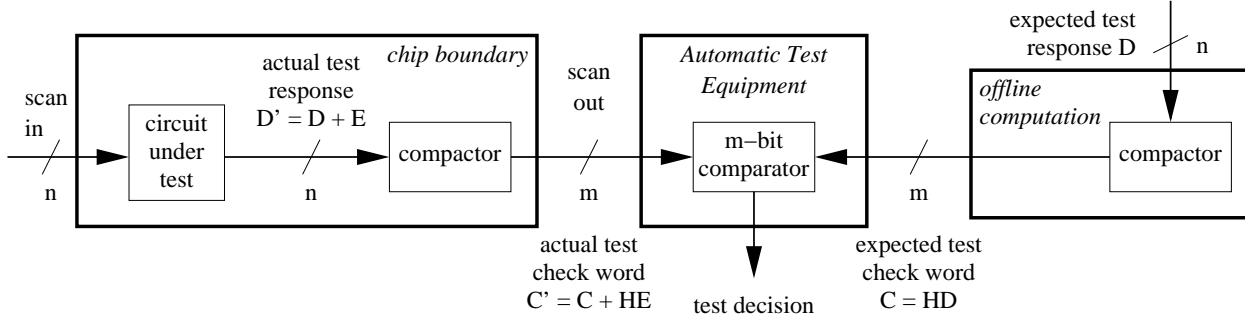


Figure 1. The i-Compact system model.

paction technique that didn't require a mask before the data enters the compactor. Their paper in ITC-2002 gives an excellent background on many other compaction techniques. We follow their lead in exploring the problem of providing mask-free compaction in the presence of unknowns. However, our work is primarily based on Saluja and Karpovsky, thus we term our compactors Saluja-Karpovsky Compactors. We extend their technique to account for a variable number of unknowns going in to a compactor and to explore tradeoffs between tester memory requirements and test time.

Our work relies on the coding theory results of Epstein [1958] for error detection and correction in the presence of erasures. An erasure in the testing context is a known bit-line that carries an unknown logic value. Unlike a bit-error, the location of the offending bit-line is known a priori at the time of decoding. We show that the unknown logic values coming into a space compactor can be treated identically to the classical erasure definition with additional support from a tester.

In the next section, we show how any linear binary code can be used to construct a compactor. The following section illustrates our hardware implementation with an example. Finally, we discuss some practical issues involving testers.

2 Preliminaries

For the purposes of this paper, we consider only binary linear codes. A binary code (n, k) is a set of 2^k code words of length n bits. Generally, a binary linear code is represented by an $m \times n$ check matrix H ($m = n - k$), and the relation $HD = 0$ defines the valid code words (the code word D is an $n \times 1$ vector). For codes with reasonable error control capabilities, m is a small factor times $\log_2 n$.

Consider the test system diagram in Figure 1, which illustrates the i-Compact approach in the absence of unknowns. As shown by Saluja and Karpovsky [1983], in the context of test compaction, any n -bit data word D can be entered into the compactor; the compactor applies the code matrix H to the data word to produce a check word $C = HD$ of m bits that serves as the compacted test

response. The expected test response and check word (as produced by a logic simulator) are unprimed in the figure (and text), while the actual test response and check word (as produced by a chip under test) are primed. The actual response may contain errors, which are included by adding (XOR'ing) an error term E to the expected response D , as shown in the figure.

Theorem 1 (Saluja-Karpovsky) If a code used for compaction provides a Hamming distance of d between valid code words, no distinct test responses closer than distance d can produce the same check words.

Proof: A proof by contradiction is straightforward. Let D and D' be separated by a Hamming distance smaller than d but have the same check word C . As the code is linear, $H(D + D') = C + C = 0$, implying that $D + D' = E$ is a valid code word with a Hamming distance of less than d from the code word of all 0s, contradicting the assumption that the original code provides distance d .

Using a check matrix H with Hamming distance d as a compactor, all distance-based coding theory results are directly applicable. The differences between such applications and the use of i-Compact arise primarily from the fact that i-Compact provides the Automatic Test Equipment (ATE) with only the compacted form of the actual chip response to a test; the response itself is not available. Information such as the bit positions of unknowns (erasures) must thus be stored explicitly on the ATE. Consider the following theorem, analogous to that of classical coding theory.

Theorem 2 An i-Compact compactor matrix with distance d can do any one of the following:

- (1) detect up to $d - 1$ single-bit errors,
- (2) detect up to e errors in the presence of up to x unknowns, where $e + x < d$, or
- (3) correct up to t errors and identify up to x unknowns, where $2t + x < d$.

Proof: Part (1) simply restates Theorem 1. For the remainder of the proof, we use the notation of Figure 1, and write $\Delta(D, D')$ for the Hamming distance between D and D' .

For part (2), consider the set S of correct test responses generated by replacing the unknowns in D with all possible

combinations of 0s and 1s. Let R be the set of all possible test responses: in this case, the set generated by injecting all members of S with arbitrary combinations of up to e errors. To prove part (2), it suffices to show that no response in S has the same check word as a test response with errors (in $R \setminus S$), allowing error detection through comparison with the check words of all correct test responses. Let $X \in S$ be an arbitrary correct test response, and assume that $D' \in R \setminus S$ contains at least one error. Let $P \in S$ be the correct test response formed by correcting the bits of D' . As D' can have no more than e errors, we have $\Delta(P, D') \leq e$. Similarly, as any two correct responses differ only in the values assigned to the unknown bits in D , we have $\Delta(X, P) \leq x$. By the triangle inequality, $\Delta(X, D') \leq \Delta(X, P) + \Delta(P, D') \leq x + e < d$. We then apply Theorem 1 to see that the check words for D' and X cannot be the same, completing the proof, since the choices of D' and X were arbitrary. As is the case with classical coding theory, the check words of two responses with errors are not necessarily unique, and the values of the unknowns cannot be determined unless the number of errors is further restricted, as in part (3).

To prove part (3), we take an approach analogous to that used for part (2). The set S is defined in the same manner as before, and the set R in an analogous manner (with the number of bits in error limited to t). Observe that it suffices to show that no two responses in R produce the same check words, allowing error correction and unique identification of unknown values. Let $D'_1 \in R$ and $D'_2 \in R$ be two arbitrary possible responses with up to t errors in each, and let $P_1 \in S$ and $P_2 \in S$ be the correct test responses formed by correcting the errors in D'_1 and D'_2 , respectively. We apply the triangle inequality twice to obtain $\Delta(D'_1, D'_2) \leq \Delta(D'_1, P_1) + \Delta(P_1, P_2) + \Delta(P_2, D'_2) \leq t + x + t \leq d$. Theorem 1 thus guarantees that the check words for D'_1 and D'_2 are unique, allowing the ATE to match the observed check word and to identify both the observed values of unknowns and the bits in error in the observed response D' .

Consider for a moment the nature of the comparator in the ATE in Figure 1. In the absence of unknowns in the expected response, this comparator simply matches the pre-calculated expected check word stored in memory against the actual check word received from the circuit under test. When some number of unknowns are present in the expected response, the unknown values can take either logic value (0 or 1) in the actual response, requiring that several possible results be considered. Towards this end, the two check words are bitwise XOR'd to produce an *error syndrome*. If we restrict the actual response D' to a Hamming distance of d from the expected response, this syndrome serves as the starting point for most of the error control applications suggested by Theorem 2. The theorem provides only bounds on the capability of a code, however. There is

no guarantee that cost-effective practical error control techniques exist for compaction with an arbitrary code of minimum distance d . We discuss the difficulty and practical tradeoffs of making use of the syndrome later in the paper, after providing a few examples.

3 Examples

We now explore the error control capabilities of Saluja-Karpovsky compactors through a series of examples. We use the check matrix for the (7,4) Hamming Code extended to the (8,4) SEC-DED code by adding a parity bit over all data bits, given by

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Denote the expected test response D by $d_1d_2d_3d_4d_5d_6d_7d_8$, and the expected check word C by $c_1c_2c_3c_4$. The following equations specify C :

$$\begin{aligned} c_1 &= d_1 \oplus d_6 \oplus d_7 \oplus d_8 \\ c_2 &= d_2 \oplus d_5 \oplus d_7 \oplus d_8 \\ c_3 &= d_3 \oplus d_5 \oplus d_6 \oplus d_8 \\ c_4 &= d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_8 \end{aligned}$$

These bits are pre-computed and stored in ATE memory. The same encoder is used to compact the actual test response D' on-chip, and only the actual check word $C' = HD'$ is delivered to the ATE. The code used is a distance-4 code, so it can correct up to $x \leq 3$ erasures; or detect up to e errors in the presence of up to $x \leq 3 - e$ erasures; or correct one error and one erasure. We demonstrate these capabilities in the following three examples.

3.1 Three unknowns and no errors

Let $D = X_1X_2X_301101$ and $D' = 01101101$. The actual check word is then $C' = HD' = 0101$. Keeping the symbolic names for X's, we find C in terms of the X_i . We XOR C with C' to find the error syndrome, which we then equate to 0 and solve for the X_i . The equations in this case are:

$$\begin{aligned} 0 &= X_1 \\ 0 &= \overline{X_2} \\ 0 &= \overline{X_3} \\ 0 &= X_1 \oplus X_2 \oplus X_3 \end{aligned}$$

The solution gives the unique values: $X_1 = 0$, $X_2 = 1$, and $X_3 = 1$, but note that some errors cannot be detected. For example, changing d'_8 to 0 does not prevent the equations from being solved. In general, for large codes and large number of erasures, one needs to solve a set of linear equations as originally suggested by Epstein [1958]. However, in our situation, the number of X's is small and the

code distance is small. So we can use either trial-and-error, or use an exhaustive search for a solution, by trying all possible binary combinations of X's. In the above case, for example, we can generate all eight possible check words C and pick the combination that matches C' . Other approaches are possible, trading tester memory for test time, as we discuss later.

3.2 Two unknowns and one error

When only two inputs are unknown, the same distance-4 code can detect any single error. Let $D = 001X_101X_20$ and $D' = 00101110$. The actual check word is then $C' = HD' = 0010$. Computing all possible values for C by setting $X_1X_2 = 00, 01, 10, 11$, we find respectively $C = 1000, 0101, 1001, 0100$. As none of these matches C' , we declare an error.

3.3 One error and one unknown

The same code can be used to identify a single error in the test response in the presence of an unknown. Such a capability is useful in failure analysis and diagnosis of chips.

Let $D = 0100100X_1$ and $D' = 01011001$. The actual check word is then $C' = HD' = 1100$. Possible values for C include only 0010 and 1101, giving $C + C' = 1110$ when $X_1 = 0$ and $C + C' = 0001$ when $X_1 = 1$. Only the second matches a column of H , and the column matched is that of bit position 4, indicating that $X_1 = 1$ and d_4 was observed in error. Since the diagnosis is usually a slow process, off of the main test floor, storage and computational complexity for error correction are usually not a serious concern.

4 Space-Time Tradeoffs

The error analysis in the previous sections illustrates the capabilities of Saluja-Karpovsky compactors, but does not provide any insight as to a practical implementation. In this section, we discuss three approaches through which one can trade between space in the form of ATE memory and time in the form of additional computation to handle unknowns.

The first approach is the simplest and favors the use of additional ATE memory in exchange for fast and simple operations. When an expected test response contains x unknowns, this approach simply stores all 2^x check words, allowing the ATE to compare them one by one against the actual test check word and to avoid any complexity.

The second approach reduces the memory requirements on the ATE by requiring the ATE to produce linear combinations of check words. We produce only $x + 1$ check words for an expected test response containing x unknowns: one in which all unknown bits are set to zero, and a compactor matrix column h for each unknown in the expected test response. Check words for comparison are then simply sums of the check words provided. As an example, with three unknowns, the check words $C_{000}, h_{001}, h_{010}$,

and h_{100} are written into the ATE memory. The remaining check words—corresponding to all non-zero combinations for unknowns—are then calculated by the ATE for comparison with the actual test check word using the following equations:

$$\begin{aligned} C_{001} &= C_{000} \oplus h_{001} \\ C_{010} &= C_{000} \oplus h_{010} \\ C_{011} &= C_{000} \oplus h_{010} \oplus h_{001} \\ C_{100} &= C_{000} \oplus h_{100} \\ C_{101} &= C_{000} \oplus h_{100} \oplus h_{001} \\ C_{110} &= C_{000} \oplus h_{100} \oplus h_{010} \\ C_{111} &= C_{000} \oplus h_{100} \oplus h_{010} \oplus h_{001} \end{aligned}$$

A third option replaces the check word basis with the positions of the unknown values within the expected test response. Each position requires $\lceil \log_2 n \rceil$ bits to specify, which is fewer than m if d is large. The compactor matrix is also stored on the ATE, allowing the check word basis used in the last approach to be looked up using the positions of the unknowns as column indices. More complex approaches based on error correction capabilities are also possible, but cannot in general provide significant reductions in storage.

The tradeoffs between these three approaches are compared in Table 1. The memory required to store the actual test check words are included as separate terms in each entry in the table; in practice, only those check words corresponding to expected responses with unknowns need be stored. Note that the unknown and error detection capabilities of each of the approaches are equivalent: $x + e < d$.

5 Implementation of i-Compact

In this section, we present the overall organization of i-Compact, including the encoder (compactor), the memory organization on the ATE side, and the basic operations necessary to perform error detection. Since the encoder cost must be small in any practical application of i-Compact, we mention here only two types of codes among many. The first is the distance-4 Hamming Code [1950] and the second is the BCH code by Bose and Ray-Chaudhuri [1960] and independently by Hocquengham, of distance five or higher. More powerful codes are unlikely to be necessary for most chip testing applications. A distance-4 Hamming code not only detects up to three errors, but also detects any odd number of errors and several even numbers of errors as well. The error escape probability in the chip testing context is extremely small. The defect escape probability is even smaller, as a defect may produce error bits several different times in the test session, and is unlikely to produce error patterns that all escape detection by the Hamming code. So it is unlikely that a user of i-Compact needs a larger Hamming distance code such as a BCH code. We mention them

Table 1. ATE storage requirements for an n -to- m compactor using a distance d code, ℓ test cycles, and x_i unknowns in test cycle i . The rightmost terms account for storage of actual responses.

approach	bits stored per test	bits stored per session	computational complexity
store all check words	$m2^{x_i} + m$	$m \sum_{i=1}^{\ell} 2^{x_i} + m\ell$	word-by-word comparison
store basis for check words	$m(x_i + 1) + m$	$m\ell + m \sum_{i=1}^{\ell} x_i + m\ell$	calculate $2^{x_i} - 1$ linear combinations in ATE
store indices of unknowns and compactor matrix	$m + x_i \lceil \log_2 n \rceil + m$	$m\ell + \lceil \log_2 n \rceil \sum_{i=1}^{\ell} x_i + mn + m\ell$	look up matrix columns, then calculate $2^{x_i} - 1$ linear combinations in ATE as above

only for those who need a guarantee of minimum detection of errors greater than three.

The overall organization of i-Compact appeared in Figure 1. Test responses are made up of n bits, which are compacted into m bits. With a distance-4 Hamming code, $n \leq 2^{m-1} - 1$. Implementation of the encoder as XOR trees and its testability is covered by Reddy, Saluja and Karpovsky [1985]. If the delay through XOR tree exceeds the maximum allowable delay for a specified scan-out rate, one can easily pipeline the XOR tree to meet the specified rate.

On the ATE side a suggested data structure for storing the expected responses are shown as $(m + 2)$ -bit memory words. Each word has a 2-bit tag and an m -bit check word, with tag values assigned the following meanings:

- 00:** (No X) *One Expected response.*
- 01:** (One X) *Two Alternate Correct responses.*
- 10:** (Two X's) *Four Alternate Correct responses.*
- 11:** (Too many X's) *Ignore some bits of the response.*

With the last entry above, the ATE ignores those bits of the check word affected by the unknowns, either by using existing masking techniques or by explicitly storing one mask word and one comparison check word.

If we were to use a distance-6 BCH code, we can detect an error in the presence of 4 unknowns. Such a scheme requires the ATE to compare against 2, 4, 8 or 16 possible correct check words. An arbitrary number of these check words can be stored as a contiguous list in memory using 2-bit tags, with list structure encoded in the tag as shown below.

- 0x:** Null list (*Ignore some bits of the response.*)
- 10:** Start list
- 11:** End list

6 Handling More X's

Suppose we face a situation where a particular chip produces more than two X's on many more scan-out cycles that

we cannot ignore. Or that certain faults cannot be detected without producing several X's at the outputs. Do we need to use more expensive codes? The short answer is, not necessarily. We can continue to use Hamming distance-4 code to detect errors in presence of more than two unknowns. However, we cannot guarantee that we always detect such errors. We could for example store 32 alternate check words in the ATE for a response with five X's. Let us take the example of circuit with 500 scan outputs. The Hamming check word width m is 10, resulting in compaction by a factor of 50. There are 1024 possible check words. The probability of a random error mapping in to the 32 stored responses is just 32/1024, providing a 96.8% detection probability of an error in the presence of five X's. We can improve this coverage by using less efficient compaction. For example, we can use 12 instead of 10 bits for the check word width, which still compacts responses by a factor greater than 40. With a 12-bit check word, we increase the random error detection probability to $(1 - 32/4096) = 99.2\%$. Similarly, we can store 64 or 128 responses to handle six or seven X's, with a detection probabilities of 98.4% and 96.8%, respectively. There is not much impact on the ATE memory requirement, since we need to store multiple responses only a few times out of several hundred thousand scan cycles.

Another method for handling more X's is by finding the relation between various X's. Say for example, during logic simulation of a test vector, we find that a bus is producing unknowns but no other logic gate is producing any unknowns. If we do a logic simulation with the X at the bus as a Boolean variable, all outputs will now be 0, 1, X or X'. No matter how many fan-outs the bus has, the output contains only one true unknown from the point of view of our decoder. If we have two nodes in the circuits producing unknowns, we can carry two logic symbols X_1 and X_2 through logic simulation. If these two variables converge at an AND gate, for example, we can declare the output $X_1 X_2$. However, to simplify the logic simulation code, one can also declare the output as X, a true unknown. In either

case, fewer true unknowns reach the outputs when expressions based on X_1 and X_2 are propagated. Logic simulation with many unknowns is covered by Carter, Rosen, Smith and Pitchumani [1989].

7 ATE Considerations

Most of today's ATE architectures have a dedicated controller per pin. Each pin is monitored independently of other pins. During a test session, an ATE cannot make an immediate decision on several alternate possible responses. A pin monitor can only decide if the expected response is 0 or 1, or it can ignore the response, as in the case of an expected X. It cannot, for example, process a command that says that both 0000 and 1111 are correct responses. Mitra and Kim's X-Compact [2002], which requires about twice as many output pins as i-Compact, has an advantage over i-Compact method in this regard. Each bit from the X-Compact can be independently monitored, since the construction of the code guarantees that an error reaches some pin while other pins are ignoring X's. In i-Compact, there is no concept of an X reaching the compactor output pins. Therefore, with i-Compact, the ATE must first collect the responses and then analyze them later. For all test responses that do not have any X's, the i-Compact approach permits pin-by-pin checking, and a failing response can be declared immediately. For all other responses, the ATE must respond later. However, this aspect is not really a disadvantage for good chips, since most test sessions involve some processing after the ATE has collected the data. In other words, a die under test is not instantly declared "go" immediately after it has passed the Boolean test. There are a number of parametric measurements that are analyzed later. In some defect-based testing, for example, the die position with respect to the wafer is part of the decision-making process. In this case, the go/no go decision is postponed until after all dies on the wafer have been tested. In summary, the needs of i-Compact data processing are compatible with modern test practices.

8 Concluding Remarks

We presented i-Compact, a space compaction organization that is effective in reducing say 500 outputs by a factor of 50, or 1,000 outputs by a factor of 90, while detecting errors in the presence of unknowns. The compactor itself is based on well-known codes such as Hamming or BCH.

We showed that the erasure and error detection-correction properties are preserved in our ATE-based compactor. We did so by using Saluja-Karpovsky compactors and classical coding theory to create i-Compact, a compactor assisted by ATE. In the process, we also demonstrated that an erasure is equivalent to an X in the testing domain with the appropriate assistance from an ATE. We also considered several possible approaches to trade off be-

tween test time and ATE memory requirements. The proposed technique i-Compact is circuit and ATPG independent.

9 Acknowledgements

We thank Jeff Rearick of Agilent Corp. for valuable input on ATE considerations. We also thank Subhasish Mitra and Kee Sup Kim of Intel Corp. for their paper at the ITC, which inspired us to work on this problem.

References

Historic Coding Theory References:

- [1] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, 29:147–160, 1950.
- [2] M. A. Epstein, "Algebraic decoding for a binary erasure channel," in *IRE Natl. Conv. Rec.*, volume 6, pages 56–69, 1958, Part 4.
- [3] R. C. Bose, D. K. Ray-Chaudhuri, "On a class of error correction binary group codes," *Inf. Control*, 3:68–79, 1960.

Historical Compaction References:

- [4] N. Benowitz, D. F. Calhoun, G. E. Alderson, J. E. Bauer, C. T. Joeckel, "An advanced fault isolation system for digital logic," *IEEE Transactions on Computers*, C-24(5):489–497, May 1975.
- [5] R. A. Frohwerk, "Signature analysis: A new digital field service method," *HP Journal*, pages 2–8, May 1977.
- [6] G. Konemann, J. Mucha, G. Zwiehoff, "Built-in logic block observation technique," in *Proc. IEEE Intl. Test Conf.*, pages 37–41, October 1979.

Compaction References used for this work:

- [7] K. K. Saluja, M. Karpovsky, "Testing computer hardware through data compression in space and time," in *Proc. IEEE Intl. Test Conf.*, pages 83–89, 1983.
- [8] S. Mitra, K. S. Kim, "X-compact: an efficient response compaction technique for test cost reduction," in *Proc. IEEE Intl. Test. Conf.*, pages 311–320, 2002.

Other References:

- [9] S. M. Reddy, K. K. Saluja, M. Karpovsky, "A data compression technique for built-in self test," in *Proc. of Fault-Tol. Comp. Symp.*, pages 294–299, June 1985.
- [10] J. L. Carter, B. K. Rosen, G. L. Smith, V. Pitchumani, "Restricted symbolic evaluation is fast and useful," in *IEEE Intl. Conf. on Comp.-Aided Design (ICCAD)*, pages 38–41, Nov. 1989.
- [11] I. Hamzaoglu, J. H. Patel, "Reducing test application time for full scan embedded cores," in *Proc. on Fault-Tolerant Comp. Symp.*, pages 260–267, June 1999.

General Coding Reference:

- [12] R. E. Blahut, *Theory and Practice of Error Control Codes*. Addison-Wesley, Reading, Mass., 1983.