applied sciences

MDPI

Article

An Approach for Controlled Random Tests with a Given Hamming Distance Generation

Ireneusz Mrozek [1,*], Marek Kopczewski [1] and Vyacheslav N. Yarmolik [2]

[1] Faculty of Computer Science, Bialystok University of Technology, 15-351 Białystok, Poland; marek.kopczewski@pb.edu.pl
[2] Faculty of Computer Science, Belarusian State University of Informatics and Radioelectronics, 220013 Minsk, Belarus; yarmolik10ru@yahoo.com
* Correspondence: i.mrozek@pb.edu.pl

Abstract

This paper addresses the challenges of testing computing systems and their hardware components, especially memory devices. It highlights the limitations of traditional random testing. Such methods often fail to use available information about the system under test and previously generated test patterns. The potential of controlled random testing, which incorporates knowledge of prior patterns, is therefore explored. A class of controlled random tests with a limited number of test patterns is identified and analyzed, including existing standard approaches. The paper introduces a novel measure of dissimilarity between test patterns. This measure is based on calculating Hamming distances for binary patterns after mapping them into different numeral systems, including quaternary, octal, and hexadecimal. We propose a method for generating controlled random tests with a guaranteed minimum Hamming distance. It is based on representing binary patterns as symbols from non-binary numeral systems. In this way, ensuring a specific Hamming distance in the symbolic domain also guarantees at least the same distance in the binary representation. We evaluate the effectiveness of the proposed method through simulations, particularly in the context of memory testing and the detection of multicell faults, i.e., errors caused by interactions between multiple memory cells. This approach can enhance the efficiency and reliability of test procedures in embedded systems, memory diagnostics, and safety-critical applications.

Keywords: test; test pattern; computing systems testing; random test; controlled random test; Hamming distance; pattern-sensitive faults; March tests

Academic Editor: Christos Bouras

Received: 29 July 2025
Revised: 3 September 2025
Accepted: 9 September 2025
Published: 11 September 2025

Citation: Mrozek, I.; Kopczewski, M.; Yarmolik, V.N. An Approach for Controlled Random Tests with a Given Hamming Distance Generation. *Appl. Sci.* **2025**, *15*, 9951. https://doi.org/10.3390/app15189951

1. Introduction

Testing is still the main method used to verify the quality of software, hardware, memory devices, and applications. Even though many design-for-test and analysis methods have been proposed, testing remains a process that requires considerable effort and time. Therefore, systematic and automated procedures for generating test data are very important. Since the 1960s, probabilistic approaches, usually called random testing, have been widely used because they are simple to understand and easy to apply [1–3]. However, the efficiency of purely random methods is low: they usually ignore information about the system under test and about earlier test patterns, which often results in weaker fault detection [4,5].

To overcome this limitation, different controlled versions of random testing have been developed. In this paper, we use the term *controlled random testing* (CRT) [6] for methods

that select the next pattern based on its difference from the previous ones. The Hamming distance is most often used as a measure of this difference. A related idea, well known in software testing, is called *adaptive random testing* (ART). A more detailed overview of these approaches is given in Section 2.1, while the theoretical background relevant for CRT is presented in Section 2.2.

The main problem studied in this paper is the choice of a difference measure that (i) can properly describe the diversity between test patterns and (ii) can be computed with low complexity [5,7]. The classical Hamming distance is widely used, but it often treats different patterns as equally distant, even when their structures are sufficiently different. To address this, we propose to extend the idea of Hamming distance by interpreting binary patterns in other numeral systems and by defining a new vector-based dissimilarity measure. This allows us to construct controlled tests that remain efficient to compute but can better distinguish between candidate patterns.

Contributions. The paper offers the following contributions. First, we introduce a framework in which an n-bit binary pattern can be mapped to sequences over alphabets of higher radix while still allowing for distance calculations in a consistent way. Second, we define a new vector-based dissimilarity measure based on these representations and discuss its main properties. Third, we propose a method for generating controlled random tests with a given minimum Hamming distance that reduces the need for expensive candidate checks. Finally, we show in experiments with memory-oriented examples that our method provides a more effective selection of patterns at a reasonable computational cost.

Paper organization. Section 2 is divided into two parts. Section 2.1 reviews earlier work on controlled random testing. Section 2.2 presents the coding-theoretic background and shows bounds that limit the construction of CRT. Section 3 introduces the representation-based extensions of the Hamming distance and the new dissimilarity measure. Section 4 describes the method for generating tests with a given minimum Hamming distance. Section 5 presents the experimental results, and Section 6 gives the conclusions and future directions.

2. Controlled Random Tests Analysis

2.1. Related Work

The related work can be grouped into three major streams: (i) pseudo-random and pseudo-exhaustive testing, (ii) diversity-driven methods such as Antirandom Testing and its extensions, and (iii) adaptive random testing (ART) and its numerous variants. Below, we summarize key contributions within each stream.

Random testing has long been applied in hardware and memory verification due to its simplicity and low implementation cost. In practice, purely random vectors are seldom used directly; instead, pseudo-random sequences are commonly employed, often generated by linear feedback shift registers (LFSRs) within built-in self-test (BIST) schemes [8]. Exhaustive and pseudo-exhaustive approaches have also been investigated to guarantee high fault coverage [9]. Early seminal work by McCluskey [10] introduced a pseudo-exhaustive testing methodology that became a foundation for later approaches. Fujiwara [11] provided further theoretical and practical developments, placing pseudo-exhaustive methods within a broader framework of logic testing and design for testability. Karpovsky, Yarmolik, and van de Goor [12] subsequently applied pseudo-exhaustive techniques to RAM testing, demonstrating their potential for memory devices. While such methods ensure thorough fault coverage, they are impractical for large memories due to excessive computational requirements. Such drawbacks motivated the development of more advanced approaches that explicitly control the diversity of generated test patterns.

A significant breakthrough was the introduction of Antirandom Testing by Malaiya [13–15], who first defined and demonstrated the method as a distance-based black-box testing strategy. Yin [16] subsequently developed a practical tool for generating hardware test sequences based on the principle of maximizing dissimilarity between test vectors, providing one of the earliest implementations of the antirandom concept. The idea of maximizing dissimilarity was later refined through extensions, including, among others, Fast Antirandom Testing (FAR) [17], Scalable Antirandom Testing (SAT) [18], and Pseudo-Ring Testing (PRT) [19]. Notably, most of these extensions were developed in the context of hardware and memory testing.

In parallel, the concept of selecting tests according to their distance from previous ones was generalized into what became known as adaptive random testing (ART) [20–22]. Unlike the earlier hardware-oriented approaches, ART and its numerous variants were primarily investigated in software testing. These include Good Random Testing [23], Restricted Random Testing [24,25], Maximum Distance Testing [26], Mirror Random Testing [27], Orderly Random Testing [28], hybrid adaptive random testing [29], and Evolutionary Random Testing [30], among others. A variety of distance metrics have been explored in this context, including minimum, average, maximum, centroid-based distances, discrepancy, and membership grade, as well as Hamming distance [7]. None of these metrics can be regarded as predominant across all ART variants, but they share the common goal of enforcing diversity when selecting new test cases. Comprehensive surveys, such as Anand et al. [5] on automated test case generation, Grindal et al. [31] on combination and diversity-driven strategies, Chen et al. [32] and Huang et al. [7] on adaptive random testing, and Feldt [33] on quantifying test diversity, provide broader overviews of these approaches and underline the central role of diversity metrics in general.

Despite these contributions, most existing methods still rely on evaluating specific characteristics of previously generated test sets. The vast majority of the approaches presented share the common goal of maximizing diversity among test patterns. However, this goal is most often achieved at the expense of increased computational overhead [34]. While such costs may be acceptable in some software testing contexts, they become a serious limitation in hardware- and memory-oriented testing, where the efficiency of test generation is a critical requirement. Therefore, there is a clear need for methods that ensure sufficient diversity of test patterns while significantly reducing the computational burden.

2.2. Formal Analysis of Controlled Random Tests

Following the discussion of related work, this subsection provides a theoretical background on controlled random tests (CRTs). The analysis emphasizes the role of Hamming distance as the principal diversity metric and introduces fundamental bounds and constructions that shape the efficiency of CRT generation. The presented considerations are rooted in the classical results of coding theory, including Hamming's seminal work on error-detecting and error-correcting codes [35], the comprehensive treatment by Peterson and Weldon [36], and the Plotkin bound [37], while also extending our earlier research on Multi-Run Memory Tests [38] and optimal controlled random tests [39].

In the following discussions, we consider a sequence T_i of data $t_{i,0}, t_{i,1}, \ldots, t_{i,n-1}$ as a test pattern $T_i = t_{i,0}t_{i,1}\ldots t_{i,n-1}$ consisting of n elements $t_{i,l}$, where $l \in \{0, 1, 2, \ldots, n-1\}$, generally represented in an arbitrary alphabet. As shown in [7], the next test pattern T_i in a controlled random test is designed to differ as much as possible from the previously generated patterns $T_0, T_1, \ldots, T_{i-1}$. The hypothesis assumes that for two test patterns with the maximum difference, the number of faults (errors) detected by the second pattern will also be maximized. The Hamming distance $HD(T_i, T_j)$ for $j \in \{0, 1, \ldots, i-1\}$ is

often used as a criterion to distinguish the test pattern T_i from the previous patterns $T_0, T_1, \ldots, T_{i-1}$ [7,32].

For the general case, the Hamming distance is calculated by comparing two sequences of data, $T_i = t_{i,0}t_{i,1}\ldots t_{i,n-1}$ and $T_j = t_{j,0}t_{j,1}\ldots t_{j,n-1}$ each consisting of n characters $t_{i,l}$ and $t_{j,l}$ from an arbitrary alphabet [35,36].

The Hamming distance $HD(T_i, T_j)$ between T_i and T_j is defined as the number of positions at which $t_{i,l}$ and $t_{j,l}$ differ, and it can be expressed as

$$HD(T_i, T_j) = \sum_{l=0}^{n-1} \delta(t_{i,l}, t_{j,l}), \tag{1}$$

where

$$\delta(t_{i,l}, t_{j,l}) = \begin{cases} 1, & \text{if } t_{i,l} \neq t_{j,l}, \\ 0, & \text{if } t_{i,l} = t_{j,l}. \end{cases}$$

When comparing n characters in the patterns T_i and T_j, the minimum value of the Hamming distance, $minHD(T_i, T_j)$, is 0 if all characters match, and the maximum value, $maxHD(T_i, T_j)$, is n if all n characters differ. For example, in the case of binary number systems, the Hamming distance $HD(T_i, T_j)$ between $T_i = 0110$ and $T_j = 1100$ is 2, as they differ at two positions.

Most commonly, binary test patterns are considered, but they can also be interpreted as sets of characters from other alphabets corresponding to different numeral systems. For example, quaternary, octal, hexadecimal, and other alphabets can be used, where a fixed number of consecutive bits in the original binary patterns represent the binary code of a character in the corresponding alphabet. For instance, the binary pattern $T_i = 01011011_2$, when divided into groups of two consecutive bits, can be represented in the quaternary number system, which uses an alphabet of four characters (0, 1, 2, and 3), as $T_i = 1123_4$. In the hexadecimal system, the same pattern T_i takes the form $T_i = 5B_{16}$.

The main idea behind most approaches to controlled random test generation is to select, from a given set of test candidates, the pattern T_i that has the maximum Hamming distance with respect to the previously included patterns $T_0, T_1, \ldots, T_{i-1}$. Various criteria can be used for selecting T_i; however, the most common is to maximize the value of $minHD(T_i, T_j)$, where $j \in \{0, 1, \ldots, i-1\}$. In this case, the generated test will be characterized by the minimum Hamming distance between any two test patterns included in the test [7]. As a result, the controlled random test is defined by the value of the minimum Hamming distance, as described in the following definition.

Definition 1. *The value $minHD(T_i, T_j)$ for a controlled random test $T \in \{T_0, T_1, \ldots, T_{q-1}\}$ is equal to the minimum Hamming distance between two arbitrary test patterns T_i and T_j, where $i \neq j$ and $i, j \in \{0, 1, 2, \ldots, q-1\}$.*

In terms of coding theory, the $minHD(T_i, T_j)$ characteristic can be regarded as the code distance d of the code $T \in \{T_0, T_1, \ldots, T_{q-1}\}$, which represents the smallest Hamming distance between different pairs of code words $T_0, T_1, \ldots, T_{q-1}$. Therefore, based on the fundamental principles of coding theory, several useful conclusions can be drawn that must be considered when generating controlled random tests.

In particular, a significant feature of controlled random tests is their limited length. This follows from the fact that the larger the minimum distance $minHD(T_i, T_j)$, used as a criterion for including T_i in the test, the fewer patterns exist that satisfy this criterion. This relationship is described by the Hamming bound [35,36].

Hamming Bound. The estimation of the Hamming bound for $d = minHD(T_i, T_j) = 2r + 1$, where r is an integer, can be expressed as the inequality:

$$q \leq \frac{b^n}{\sum_{k=0}^{r} \binom{n}{k}(b-1)^k}. \quad (2)$$

Here, the Hamming bound denotes the maximum possible size q of a b-ary block code T of length n and minimum Hamming distance d between code words. In the context of controlled random tests, the value $d = minHD(T_i, T_j)$ directly affects the test length. For example, in the case of binary patterns ($b = 2$) with $n = 8$ and $d = minHD(T_i, T_j) = 7 = 2 \times 3 + 1$, the Hamming bound can be calculated as

$$q \leq \frac{2^8}{\sum_{k=0}^{3} \binom{8}{k}(2-1)^k}.$$

As shown in this example, increasing the Hamming distance $minHD(T_i, T_j)$ to a value of 7 reduces the estimate of q to 2. This means that the controlled random test T for $n = 8$ and $d = minHD(T_i, T_j) = 7$ will consist of no more than two patterns: T_0 and T_1. It is important to note that the pattern T_0 is generated randomly and can take any of $2^n = 2^8$ binary values, while the second pattern T_1 is selected to satisfy the criterion $minHD(T_0, T_1) \geq 7$. Thus, there is a large variety of controlled random tests T with $minHD(T_0, T_1) \geq 7$, but each test consists of only two patterns: T_0 and T_1. In practice, this result shows that for short memory words, only a very limited number of maximally distant test patterns can be constructed, which restricts the applicability of such strict distance requirements.

Let us consider approaches for constructing controlled random tests consisting of a minimal number q of test patterns, for which $minHD(T_i, T_j)$ takes the maximum possible value.

For the synthesis of controlled random tests with a small number of patterns q, we first examine classic codes with $minHD(T_i, T_j) \geq n/2$ [37]. It is known that the Plotkin theorem allows for determining the maximum possible number q of code words in a binary code of length n with $minHD(T_i, T_j) \geq n/2$. The Plotkin bound provides an upper limit for this value [37,39].

Plotkin Bound. If $d = minHD(T_i, T_j) \geq n/2$ and n is even, the following inequality holds for q:

$$q \leq \begin{cases} 2\left\lfloor \frac{d}{2d-n} \right\rfloor, & \text{for } 2d - n > 0; \\ 4d, & \text{for } 2d - n = 0. \end{cases} \quad (3)$$

For odd values of n, the Plotkin bound is expressed as

$$q \leq \begin{cases} 2\left\lfloor \frac{d+1}{2d+1-n} \right\rfloor, & \text{for } 2d + 1 - n > 0; \\ 4d + 4, & \text{for } 2d + 1 - n = 0. \end{cases} \quad (4)$$

Based on the application of the Plotkin bound, a formal algorithm for synthesizing controlled random tests MMHD(q), characterized by a small number q of patterns with the maximum–minimum Hamming distance ($max\_minHD(T_i, T_j)$) between test patterns T_i and T_j, is proposed in [38].

For $q = 2$, based on (3) and (4), the maximum possible value $max\_minHD(T_i, T_j)$ of the distance can be estimated. This result, $max\_minHD(T_i, T_j) = n$, and the corresponding test MMHD(2) $= \{T_0, T_1\} = \{T_0, \overline{T_0}\}$, is supported by previous findings for the optimal random test consisting of two inverse patterns, T_0 and $\overline{T_0}$ [35,36].

In the case of $q = 3$, according to the Plotkin bound, $d = max\_minHD(T_i, T_j) \leq 3n/4$. As shown in [38], the closest optimal solution MMHD(3) can be achieved only for $max\_minHD(T_i, T_j) = 2n/3$, where $2n/3 < 3n/4$. For $q = 4$, the test MMHD(4) can be constructed with $d = max\_minHD(T_i, T_j) \leq 2n/3$ [38].

By generalizing the heuristic procedure for constructing MMHD(q) for small values of q, a formal algorithm for synthesizing the MMHD(q) test for a given $q \geq 4$ was presented in [38]. According to this algorithm, the MMHD(q) test consists of q patterns with

$$max\_minHD(T_i, T_j) = (2^{q-3}) \cdot n/(2^{q-2} - 1).$$

It should be noted that for any integer q, the distance $(2^{q-3}) \cdot n/(2^{q-2} - 1)$ is greater than $n/2$; however, as q increases, it approaches $n/2$.

A very important remark concerns the size n of the test patterns, which must be considered. In order to generate MMHD(q) tests with $q \geq 4$, the value of n must be divisible by $2^{q-2} - 1$, and its minimal value is $2^{q-2} - 1$. For example, in the case of $q = 4$, one variant of the MMHD(4) test is $T = \{000, 011, 101, 110\}$ with the minimal value $n = 2^{q-2} - 1 = 2^{4-2} - 1 = 3$.

Based on the Hamming distance $HD(T_i, T_j)$ for test patterns T_i and T_j, and their Cartesian distance $CD(T_i, T_j)$ as described in [39], a method for synthesizing optimal controlled random tests (OCRTs) is considered. These tests are characterized by the conditions $HD(T_i, T_j) \geq n/2$ and $minHD(T_i, T_j) = n/2$. In the general case, the number of OCRT patterns is defined as $q = 2(\lceil \log_2 n \rceil + 1)$. A constructive algorithm for generating test patterns is presented in [39]. For the specific case when $n = 2^m$, where m is an integer, the number q of OCRT patterns $T_0, T_1, \ldots, T_{q-1}$ is given by $q = 2(m + 1)$. For example, when $n = 4$, the number of OCRT patterns is $q = 6$, and for $n = 8$, the number of patterns is $q = 8$.

The example of the MMHD(4) test with $HD(T_i, T_j) = 2$ for $n = 3$ presented in [38] and the OCRT for $n = 4$ are shown in Table 1.

Table 1. Examples of MMHD(4) for $n = 3$ and OCRT with $n = 4$ tests.

	MMHD(4)				**OCRT**			
Pattern	**1**	**2**	**3**	**Pattern**	**1**	**2**	**3**	**4**
T_0	0	0	0	T_0	0	0	0	0
T_1	0	1	1	T_1	1	1	1	1
T_2	1	1	0	T_2	0	0	1	1
T_3	1	0	1	T_3	1	1	0	0
				T_4	0	1	0	1
				T_5	1	0	1	0

These small examples demonstrate how theoretical bounds directly limit the number of feasible patterns in controlled random tests, especially for memory diagnostics where compact but diverse test sets are needed. At the same time, the MMHD(4) and OCRTs shown in Table 1 can be interpreted as templates for generating patterns of similar tests. A specific MMHD(4) or OCRT can be defined by a randomly chosen initial test pattern T_0, based on which subsequent patterns are generated by inverting the bits of T_0 according to the given templates. For example, in the case of MMHD(4) shown in Table 1, if the random initial pattern is chosen as $T_0 = 101$, the corresponding new MMHD(4) test will consist of the patterns $\{101, 110, 011, 000\}$. Therefore, in the following text, the abbreviation MMHD(q) is used to denote a family of tests with q test patterns and the corresponding value of $max\_minHD(T_i, T_j)$.

A common drawback of both approaches to generating controlled random tests, MMHD and OCRT, is the limitation and restriction on the size of their test patterns. Four main algorithms are known for constructing a set of test patterns (code words) with given properties based on an initial test (code) [36]. These algorithms utilize the following four properties, which we formulate for the case of MMHD(q) [36].

Property 1. *The result of permuting the bits $t_{i,l}$ simultaneously in all q test patterns $T_i = t_{i,0}t_{i,1}\dots t_{i,n-1}$ of the test MMHD$(q) = \{T_0, T_1, \dots, T_{q-1}\}$ is the test MMHD(q).*

Property 2. *The result of inverting the bits $t_{i,l}$ in all q test patterns $T_i = t_{i,0}t_{i,1}\dots t_{i,n-1}$ of the test MMHD$(q) = \{T_0, T_1, \dots, T_{q-1}\}$ is also the test MMHD(q).*

Property 3. *The test MMHD$(q) = \{T_0, T_1, \dots, T_{q-1}\}$ with test patterns*

$$T_i = t_{i,0}t_{i,1}\dots t_{i,u\cdot n-1},$$

consisting of $u \times n$ bits, is obtained from the test pattern $T_i = t_{i,0}t_{i,1}\dots t_{i,n-1}$ of the original test MMHD(q) by concatenating it (repeating) u times. For example, if $u = 2$, then

$$T_i = t_{i,0}t_{i,1}\dots t_{i,2n-1} = t_{i,0}t_{i,1}\dots t_{i,n-1}t_{i,0}t_{i,1}\dots t_{i,n-1}.$$

Property 4. *The result of scaling (increasing) by s times the test MMHD$(q) = \{T_0, T_1, \dots, T_{q-1}\}$ is the test MMHD(q), consisting of test patterns $T_i = (t_{i,0})^s(t_{i,1})^s\dots(t_{i,n-1})^s$, where $T_i = t_{i,0}t_{i,1}\dots t_{i,n-1}$ is the test pattern of the original test MMHD(q).*

The results of applying the above properties to the test MMHD(4) shown in Table 1 are illustrated in Table 2.

Table 2. MMHD(4) test extension examples.

Original MMHD(4)	**Property 1 (Bit Permutation)**	**Property 2 (Bit Inversion)**
0 0 0	0 0 0	1 1 1
0 1 1	1 0 1	0 1 0
1 1 0	1 1 0	0 0 1
1 0 1	0 1 1	1 0 0
Original MMHD(4)	**Property 3 (Repetition, u = 3)**	**Property 4 (Bit Scaling, s = 2)**
0 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0
0 1 1	0 1 1 0 1 1 0 1 1	0 0 1 1 1 1
1 1 0	1 1 0 1 1 0 1 1 0	1 1 1 1 0 0
1 0 1	1 0 1 1 0 1 1 0 1	1 1 0 0 1 1

The presented analysis of controlled random tests with a small number of test patterns demonstrates the feasibility of generating such tests without significant computational costs. The examples provided in Table 2 illustrate the derivation of new controlled random tests using formal methods, which enable the generation of a new set of test patterns as well as the adjustment of the pattern size n. It should be noted that the given properties apply not only to the MMHD(q) and OCRT but also to any controlled random tests.

As noted above, the key characteristic of tests with a small number of test patterns is the relationship between the value of the $max\_minHD(T_i, T_j)$ Hamming distance and the number q of test patterns. Increasing the required minimum Hamming distance $minHD(T_i, T_j)$—essentially maximizing it—reduces the number q of patterns in the generated test. It is evident that both test parameters, namely, the Hamming distance $minHD(T_i, T_j)$ and the number of test patterns, influence the efficiency and quality of the test. Intuitively, one

might conclude that increasing both parameters improves the test properties. Indeed, the more test patterns that are maximally distant from each other, the more efficient the test becomes. However, as the analysis above has shown, it is impossible to simultaneously increase both parameters. Therefore, in the subsequent discussion, we will consider an approach that focuses on increasing the number of test patterns while maintaining the minimum Hamming distance $minHD(T_i, T_j)$ at an acceptable level.

In summary, the theoretical analysis highlights a fundamental trade-off: increasing the minimum Hamming distance between test patterns inevitably reduces the number of patterns in the test. These coding-theoretic constraints motivate the search for alternative distance measures, which can better balance diversity and efficiency. Section 3 introduces representation-dependent interpretations of the Hamming distance, aimed at achieving this balance.

3. Modified Approach to Hamming Distance Calculation

The Hamming distance has significant limitations as a dissimilarity metric, as it only distinguishes fully matching patterns T_i and T_j with $HD(T_i, T_j) = 0$ while treating all other non-identical patterns equally. One argument that confirms the indistinguishability of non-matching sequences is the case of binary patterns T_i and $\overline{T_i}$, for which the Hamming distance is always constant and equal to n. For example, $HD(10000000, 01111111) = HD(10101010, 01010101) = HD(00001111, 11110000) = 8$. As seen above, the Hamming distance $HD(T_i, \overline{T_i})$ in all the given examples is equal to $n = 8$, indicating the same maximum difference across all pairs of patterns. However, the structural differences between these pairs of sequences are significant. An even greater structural difference exists in the following character sequence pairs, $00000000, 11110000$; $11111111, 00001111$; $01011010, 11010100$, for which the Hamming distance is $HD(T_i, T_j) = 4$. These examples highlight the need for alternative dissimilarity measures capable of capturing not only the number of differing bits but also the spatial and structural relationships within the sequences.

Let us examine the potential for extending the use of Hamming distance in comparing finite sequences of characters $T_i = t_{i,0}t_{i,1}\ldots t_{i,n-1}$ and $T_j = t_{j,0}t_{j,1}\ldots t_{j,n-1}$, which represent test patterns consisting of n characters (elements) $t_{i,l}$ and $t_{j,l}$, where $l \in \{0, 1, \ldots, n-1\}$. The alphabet of characters $t_{i,l}$ and $t_{j,l}$ can be arbitrary, as well as the number n of elements in the patterns T_i and T_j. Without loss of generality, we assume that the test pattern T_i is initially a binary pattern, meaning that the characters $t_{i,l} \in \{0, 1\}$.

The primary objective of the existing modifications to the Hamming distance calculation is to select, from among potential test pattern candidates, a test pattern T_i that is most different from the previously included pattern T_j.

The first modification assumes that the length of a binary test pattern is restricted to $n = 2^w$, where w is an integer. Such constraints frequently occur in practice when addressing diagnostic problems in computer systems. Under this condition, the original binary sequence

$$T_i = t_{i,0}t_{i,1}\ldots t_{i,n-1}$$

can be represented in $w + 1$ different ways denoted as $T_i(2^v)$, where $v \in \{0, 1, \ldots, w\}$. The index 2^v specifies the number of consecutive bits that form each character in the new alphabet. For $v = 0$ ($2^0 = 1$), we obtain the binary alphabet:

$$T_i(1) = t_{i,0}(1)t_{i,1}(1)\ldots t_{i,n-1}(1).$$

For $v = 1$ ($2^1 = 2$), we obtain the quaternary alphabet:

$$T_i(2) = t_{i,0}(2)t_{i,1}(2)\dots t_{i,n/2-1}(2),$$

where each character is formed from two consecutive bits of $T_i(1)$. For larger values of v, the construction continues in the same manner, producing

$$T_i(4),\ T_i(8),\ \dots,\ T_i(n/2),\ T_i(n) = T_i.$$

In the general case, the sequence $T_i(2^v)$ consists of 2^{w-v} characters. Each character of this alphabet is obtained by concatenating two neighboring characters of the previous representation $T_i(2^{v-1})$. For instance, for $T_i(2)$:

$$t_{i,0}(2) = t_{i,0}(1)\, t_{i,1}(1), \quad t_{i,1}(2) = t_{i,2}(1)\, t_{i,3}(1), \quad \dots$$

and, more generally,

$$t_{i,l}(2^v) = t_{i,2l}(2^{v-1})\, t_{i,2l+1}(2^{v-1}), \quad l = 0, 1, \dots, n/2^v - 1.$$

Thus, each representation $T_i(1), T_i(2), T_i(4), \dots, T_i(2^w)$ defines a sequence over a different alphabet, offering multiple perspectives on the same original binary pattern.

The given interpretation of the original binary patterns does not prevent the determination of the Hamming distance between the patterns T_i and T_j. Just as in the case of binary vectors, Equation (1) can also be applied here, provided that both patterns are expressed in the same chosen alphabet. Let us illustrate this with the following example for the case where $n = 2^3$.

Example 1. *As an example of binary test patterns, consider* $T_i = 01100011_2$ *and* $T_j = 01011011_2$, *for which the condition* $n = 2^w = 2^3$ *is satisfied. For each pattern of binary characters* $T_i = 01100011_2$ *and* $T_j = 01011011_2$, *in accordance with the above-described definitions, there are* $w + 1 = 4$ *representations in the form of sequences of characters belonging to different alphabets (see Table 3).*

Table 3. Hamming distance computation in multiple alphabets for $n = 8$.

	$w = 0$	$w = 1$	$w = 2$	$w = 3$
T_i	$T_i(1) = 01100011$	$T_i(2) = 1203$	$T_i(4) = 63$	$T_i(8) = c = (99)_{256}$
T_j	$T_j(1) = 01011011$	$T_j(2) = 1123$	$T_j(4) = 5B$	$T_j(8) = [= (91)_{256}$
$HD(T_i, T_j)$	3	2	2	1

In Table 3, the Hamming distance for the original binary patterns $T_i(1)$ and $T_j(1)$, as well as for their representations in different alphabets with their respective characters, is presented. In this example, ASCII codes are used to represent $T_i(8)$ and $T_j(8)$. For all cases, the value of the Hamming distance has been calculated based on Equation (1). The resulting characteristic $HD(T_i, T_j)$, represented by the four components $\{3, 2, 2, 1\}$, provides a more accurate assessment of the differences between these test patterns.

The requirement that the dimension $n = 2^w$ of a binary pattern T_i, where w is an integer, may not always be satisfied in practice. Consequently, for cases where $n \neq 2^w$, when mapping the original pattern T_i into the sequences $T_i(1), T_i(2), T_i(4), \dots$, the required number of bits equal to 2^v may be insufficient for the last character of the sequence $T_i(2^v)$, where $v \in \{0, 1, 2, \dots, w\}$. For example, considering the pattern $T_i = 0110001_2$, where $n = 7$, it can be represented as the sequences $T_i(1), T_i(2), T_i(4)$, and $T_i(8)$. However, in three cases—$T_i(2)$, $T_i(4)$, and $T_i(8)$—the required number of bits is insufficient for the

last character of the corresponding alphabet; specifically, one bit is missing for $T_i(2)$, and one bit is missing in both $T_i(4)$ and $T_i(8)$. An obvious solution to overcome this limitation is a cyclic interpretation of the original pattern $T_i = t_{i,0}t_{i,1}\dots t_{i,n-1}$. This interpretation assumes that the bit following the last bit $t_{i,n-1}$ is the first bit $t_{i,0}$, thereby using the initial bits of the pattern to obtain the required number of bits for the last character of $T_i(2^v)$. For the pattern $T_i = 0110001_2$, such an interpretation allows us to obtain $T_i(1) = 01100010_2$ $T_i(2) = 01100010_2 = 1202_4$ $T_i(4) = 01100010_2 = 62_{16}$ $T_i(8) = 01100010_2 = b_{256} = (98)_{256}$.

The notation b_{256} above, as well as the symbols "c" and "[" in Table 3, represent values in the base-256 numeral system. In each case, a group of 8 consecutive bits is interpreted as a single element of a 256-ary alphabet. Thus, $01100010_2 = 98_{10}$ is represented by b_{256} (ASCII code for the letter b), while $01100011_2 = 99_{10}$ and $01011011_2 = 91_{10}$ correspond to the ASCII symbols "c" and "[", respectively. It should be emphasized that these ASCII representations are used only as illustrative examples, since the base-256 system also includes non-printable and control characters. The purpose of this notation is to demonstrate that every 8-bit block can be treated as one symbol of a base-256 alphabet.

Removing the restriction on the size n of the binary pattern T_i by extending it to the required number of bits allows for an expansion in the number of alphabets available for different mappings of the original pattern. Naturally, considering the possibility of extending the original binary pattern to the required number of bits, the number of alphabets can be increased up to n. These alphabets consist of characters specified by one bit, two bits, three bits, four bits, and so on, up to the alphabet in which each character is determined by n consecutive bits. For example, considering the original pattern $T_i = 01100_2$ with $n = 5$ and its cyclic extensions, it can be represented in the form of sequences obtained for $n = 5$ different alphabets. The sequential representations are as follows: $T_i(1) = T_i = 01100_2$, $T_i(2) = 011000_2 = 120_4$, $T_i(3) = 011000_2 = 30_8$, $T_i(4) = 01100011_2 = 63_{16}$, and $T_i(5) = 01100_2 = C_{32}$.

Another approach to representing the original test pattern in various numerical systems with different character sets is to expand the last character of the pattern by appending, for example, all zero values. Consider the example of a test pattern $T_i = 01100$, which can be represented in five different numerical systems, each with its own alphabet. To avoid potential conflicts related to the absence of a complete set of characters (or their graphical representation) in alphabets containing a large number of symbols, each character in all numerical systems will be represented in binary form and separated by spaces. Thus, the test pattern $T_i = 01100$ can be represented in five different numerical systems as follows: $T_i(1) = 0\ 1\ 1\ 0\ 0_2$ $T_i(2) = 01\ 10\ 00_4$ $T_i(3) = 011\ 000_8$ $T_i(4) = 0110\ 0000_{16}$ $T_i(5) = 011000_{32}$.

Let us define the binary n-bit test pattern T_i as a pattern in a numerical system other than binary.

Definition 2. *The test pattern T_i, consisting of n binary characters, can be interpreted in a 2^r numerical system with 2^r characters as the pattern $T_i(r)$, where $r \in \{1, 2, \dots, n\}$. This pattern consists of $\lceil n/r \rceil$ characters, where T_i is expanded to a size of $\lceil n/r \rceil \times r$ bits by adding $\lceil n/r \rceil \times r - n$ zeros.*

For example, the test pattern $T_i = 0110001$ with $n = 7$ can be represented in the octal (2^3) numerical system with $\lceil n/r \rceil = \lceil 7/3 \rceil = 3$ characters as $T_i(3) = 011\ 000\ 100_8 = 304_8$. To achieve this representation, $\lceil n/r \rceil \times r - n = \lceil 7/3 \rceil \times 3 - 7 = 2$ zeros have been added.

Note that the above examples of interpreting the pattern T_i and Definition 2 allow us to consider binary test patterns in various number systems. Using the last example of representing the test pattern $T_i = 01100$ in $n = 5$ different number systems, let us illustrate the determination of the Hamming distance $HD(T_i, T_j)$ (Equation (1)) for each interpretation of two patterns: $T_i = 01100$ and $T_j = 01011$.

The below example (see Table 4) of determining the Hamming distance demonstrates the possibility of obtaining, based on Equation (1), several numerical assessments of the relationship between the original binary patterns T_i and T_j.

Table 4. Example of the Hamming distance calculation.

	$T_i(1)$	$T_i(2)$	$T_i(3)$	$T_i(4)$	$T_i(5)$
T_i	0 1 1 0 0	01 10 00	011 000	0110 0000	01100
T_j	0 1 0 1 1	01 01 10	010 110	0101 1000	01011
$HD(T_i, T_j)$	3	2	2	2	1

Let us now define a new measure of dissimilarity between the binary test patterns T_i and T_j, which consists of a set of numerical characteristics represented by the Hamming distances.

Definition 3 (Dissimilarity Measure $MD(T_i, T_j)$)**.** *The dissimilarity measure $MD(T_i, T_j)$ between two binary test patterns $T_i = t_{i,0}t_{i,1}\ldots t_{i,n-1}$ and $T_j = t_{j,0}t_{j,1}\ldots t_{j,n-1}$, where $t_{i,l}, t_{j,l} \in \{0,1\}$ and $l \in \{0,1,\ldots,n-1\}$, is defined as an n-component vector composed of the Hamming distances*

$$HD_1 = HD[T_i(1), T_j(1)], \quad HD_2 = HD[T_i(2), T_j(2)], \quad \ldots, \quad HD_n = HD[T_i(n), T_j(n)]$$

calculated according to Equation (1).

The analyzed characters $t_{i,l}$ and $t_{j,l}$ of the test patterns $T_i(r)$ and $T_j(r)$, according to Definition 2, are represented by $r \in \{1,2,\ldots,n\}$ binary bits. Accordingly, using Equation (1), the numerical values of the components $HD_1, HD_2, \ldots, HD_n$ of the dissimilarity measure $MD(T_i, T_j)$ are determined. Table 5 presents examples of calculating $MD(T_i, T_j)$ for various pairs of test patterns T_i and T_j in the case where $n = 5$.

Table 5. Example of the dissimilarity measure $MD(T_i, T_j)$ calculation.

	$T_i(1)$	$T_i(2)$	$T_i(3)$	$T_i(4)$	$T_i(5)$
T_i	0 1 1 0 0	01 10 00	011 000	0110 0000	01100
T_j	1 0 0 0 0	10 00 00	100 000	1000 0000	10000
$HD(T_i, T_j)$	3	2	1	1	1
T_i	0 1 1 0 0	01 10 00	011 000	0110 0000	01100
T_j	1 1 0 0 1	11 00 10	110 010	1100 1000	11001
$HD(T_i, T_j)$	3	3	2	2	1

Note that in all three examples presented in Tables 4 and 5, the same pattern $T_i = 01100$ was used as the test pattern T_i, while three different patterns T_j were selected to determine the value of the measure $MD(T_i, T_j)$. Accordingly, for the three cases shown in Tables 4 and 5, the measure of dissimilarity $MD(T_i, T_j)$ takes the following values: $MD(01100, 01011) = \{3,2,2,2,1\}$, $MD(01100, 10000) = \{3,2,1,1,1\}$, $MD(01100, 11001) = \{3,3,2,2,1\}$.

The examples presented in Tables 4 and 5 demonstrate the indistinguishability of all three patterns T_j with respect to the reference pattern $T_i = 01100$ when using the classical measure of difference—the Hamming distance—since in all three cases $HD(T_i, T_j) = HD_1 = 3$. At the same time, applying the new measure of dissimilarity (see Definition 3) reveals different degrees of difference between the patterns T_j and T_i, as expressed by the varying values of the components HD_2, HD_3, and HD_4 of the measure $MD(T_i, T_j)$.

The measure of dissimilarity $MD(T_i, T_j)$ for the binary test patterns T_i and T_j has the following obvious properties.

Property 1. The minimum value of all components $HD_1, HD_2, \ldots, HD_n$ of the measure $MD(T_i, T_j)$ is zero, that is,

$$minHD_1 = minHD_2 = \cdots = minHD_n = 0.$$

This condition occurs when the test patterns are identical, i.e., $T_i = T_j$.

Property 2. If one component HD_r, where $r \in \{1, 2, \ldots, n\}$, equals zero, then all the others are also equal to zero. Conversely, if any component $HD_r > 0$, then all other components are greater than zero as well.

Property 3. The maximum values of the components $HD_1, HD_2, \ldots, HD_n$ depend on the number of characters in the representations $T_i(r)$ and $T_j(r)$. Specifically,

$$\begin{aligned} &maxHD_1 = n, \quad maxHD_2 = \lceil n/2 \rceil, \quad maxHD_3 = \lceil n/3 \rceil, \\ &maxHD_{\lceil n/2 \rceil - 1} = \left\lceil \frac{n}{\lceil n/2 \rceil - 1} \right\rceil = 3, \\ &maxHD_{\lceil n/2 \rceil} = \cdots = maxHD_{n-1} = 2, \quad maxHD_n = 1. \end{aligned}$$

The maximum difference between test patterns T_i and T_j in terms of the new dissimilarity measure $MD(T_i, T_j)$ is achieved when T_j is the bitwise inverse of T_i. In this case, all components $HD_1, HD_2, HD_3, \ldots, HD_n$ of the measure $MD(T_i, T_j)$ reach their maximum values.
For example, for $T_i = 01100$ and its inverse pattern $T_j = \overline{T_i} = 10011$, the corresponding component values are

$$\begin{aligned} &HD_1 = maxHD_1 = n = 5, \\ &HD_2 = maxHD_2 = \lceil n/2 \rceil = \lceil 5/2 \rceil = 3, \\ &HD_3 = maxHD_3 = \lceil n/3 \rceil = \lceil 5/3 \rceil = 2, \\ &HD_4 = maxHD_4 = \lceil n/4 \rceil = \lceil 5/4 \rceil = 2, \\ &HD_5 = maxHD_5 = 1. \end{aligned}$$

Property 4. The components of $MD(T_i, T_j)$ satisfy the following relation:

$$HD_1 \geq HD_2 \geq HD_3 \geq \cdots \geq HD_n.$$

The fulfillment of this property is explained by the fact that when calculating HD_{r+1}, the number of characters included in the patterns $T_i(r+1)$ and $T_j(r+1)$ is less than or equal to the number of characters within the patterns $T_i(r)$ and $T_j(r)$. Therefore, the following inequality holds: $HD_r[T_i(r), T_j(r)] \geq HD_{r+1}[T_i(r+1), T_j(r+1)]$.

As noted in [7,13,32], the idea of controlled random tests is as follows: the next test pattern T_i is generated to be as different (or distant) as possible from the previously generated patterns $T_0, T_1, \ldots, T_{i-1}$ in terms of predetermined measures of dissimilarity. For this purpose, at each step of forming the next test pattern, a candidate is selected from a set of potential test patterns [7,13,32]. The main operation of the selection procedure is to determine the numerical value of the chosen measure of dissimilarity between two patterns: T_i, which is one of the test patterns, and T_j, which is one of the candidate test patterns. As a result, the candidate test pattern for which the measure (or measures) of dissimilarity attains the maximum value is selected as the next test pattern.

Let us explain the procedure for generating a controlled random test using the examples presented in Tables 4 and 5 for the case where the Hamming distance is applied as a measure of dissimilarity. Assume that the first pattern of the controlled random test is $T_i = 01100$, and three randomly generated candidates for the next test pattern are

$T_j = 01011$, $T_j = 10000$, and $T_j = 11001$. For each candidate pattern T_j, the value of the dissimilarity measure, as defined in Equation (1), is calculated with respect to the test pattern T_i. As shown in Tables 4 and 5, the value of HD_1 is equal to 3 in all three cases. The classical technique for generating controlled random tests assumes that any of the three candidate patterns—$T_j = 01011$, $T_j = 10000$, or $T_j = 11001$—can be selected as the next test pattern.

In cases where multiple test pattern candidates yield the maximum value of HD_1, the new measure of dissimilarity $MD(T_i, T_j)$, introduced by the authors (see Definition 3), provides a more comprehensive way to distinguish between test pattern candidates T_j with respect to the test pattern T_i. To achieve this, it is necessary to analyze the values of the next component, HD_2, of the dissimilarity measure. As demonstrated in the given example, the maximum value $HD_2 = 3$ is obtained for the pattern $T_j = 11001$, which can then be selected as the next test pattern in the controlled random test.

Based on the above example and following the classical strategy for generating random tests, we will formulate one of the rules for applying the new dissimilarity measure.

***MD*(T_i, T_j) Application Rule.** The test pattern candidate T_j is selected as the next test pattern T_i if it is the only candidate, among the entire set of test pattern candidates, that has the maximum value HD_r for the minimum value of $r \in \{1, 2, \ldots, n\}$ in the dissimilarity measure $MD(T_i, T_j)$, specifically among the components $HD_1, HD_2, \ldots, HD_n$. Otherwise, if multiple candidates have the same maximum value of HD_r, one of them is selected randomly.

Other strategies for generating controlled random tests are possible, differing from the given $MD(T_i, T_j)$ application rule for the new dissimilarity measure. For example, instead of selecting the next test pattern based on a single component of the measure, one can use an integral measure of dissimilarity, $MD_{\text{Total}}(T_i, T_j)$, defined as the arithmetic sum of its components, i.e., $MD_{\text{Total}}(T_i, T_j) = HD_1 + HD_2 + \cdots + HD_n$.

$$MD_{\text{Total}}(T_i, T_j) = \sum_{r=1}^{n} HD_r(T_i, T_j). \tag{5}$$

Table 6 presents the results of calculations, based on Equation (1), of the components $HD_1, HD_2, HD_3, \ldots, HD_8$ of the dissimilarity measure $MD(T_i, T_j)$ for the binary pattern $T_i = 00000000$ and for four test pattern candidates T_j: 11110000, 00110011, 11100010, and 10010101. The last column of Table 6 contains the value of the integral measure $MD_{\text{Total}}(T_i, T_j)$ for all four candidate patterns T_j.

Table 6. Numerical values for dissimilarity measure $MD(T_i, T_j)$.

T_j	HD_1	HD_2	HD_3	HD_4	HD_5	HD_6	HD_7	HD_8	MD_{Total}
11110000	4	2	2	1	1	1	1	1	13
00110011	4	2	3	2	2	2	2	1	18
11100010	4	3	2	2	2	2	1	1	17
10010101	4	4	3	2	2	2	2	1	20

As can be seen from Table 6, according to both criteria, namely, the $MD(T_i, T_j)$ application rule and its integral value $MD_{\text{Total}}(T_i, T_j)$, the pattern $T_j = 10010101$ will be selected as the next test pattern.

An analysis of the data presented in Table 6 shows that as r increases, the significance of the HD_r component decreases significantly. This can be explained by the fact that for $r \geq \lceil n/2 \rceil$ (see $MD(T_i, T_j)$ Property 3), all HD_r components, except for the last HD_n, take only three possible values: 0 if $T_i = T_j$, and either 1 or 2 if $T_i \neq T_j$.

The given measure of dissimilarity $MD(T_i, T_j)$ demonstrates its effectiveness in generating controlled random tests. It enables the selection of an optimal pattern T_j from a set of candidates that share the same Hamming distance from the previously included test pattern T_i. However, its application is associated with the same drawbacks as classical approaches, requiring significant computational costs. Most notably, it necessitates the determination of dissimilarity measures between candidate test patterns and previously selected test patterns.

4. Controlled Random Test Generation with the Given Hamming Distance

The significant computational complexity of generating controlled random tests has led to the development of methods for constructing such tests that do not require selecting the next test pattern from a set of possible candidates. The core idea behind these methods is to use a small number of test patterns that are maximally distant from each other in terms of the Hamming distance while avoiding the computationally expensive process of candidate selection and enumeration.

As noted in previous sections, there are approaches for constructing controlled random tests with a small number of test patterns based on formal procedures that eliminate computational costs, such as MMHD(q) and OCRTs [38]. The key characteristic of such tests is the relationship between the maximum–minimum Hamming distance, $max\_minHD(T_i, T_j)$, and the number of test patterns, q. Increasing the required minimum Hamming distance, $minHD(T_i, T_j)$, effectively maximizing it for the generated test, results in a reduction in the number of test patterns, q. Unfortunately, a simultaneous increase in both parameters—namely, the required $minHD(T_i, T_j)$ and the number of test patterns q—is not possible.

As an alternative to existing approaches, we propose a method based on increasing the number of test patterns q while maintaining the value of $minHD(T_i, T_j)$ at a moderate level. The result of implementing the proposed approach is a controlled random test consisting of binary patterns $T_i = t_{i,0}t_{i,1}\dots t_{i,n-1}$, where $t_{i,l} \in \{0,1\}$ for $l \in \{0,1,\dots,n-1\}$, and where $minHD(T_i, T_j)$, for $j \neq i$, takes given values from the set $\{0,1,\dots,q-1\}$. The main feature of the proposed approach is the use of a new measure of dissimilarity, $MD(T_i, T_j)$ (see Definition 3), introduced by the authors, which is defined for an arbitrary alphabet of test patterns. This measure allows for the estimation of the n components $HD_1, HD_2, \dots, HD_n$ that quantify the dissimilarity between two arbitrary binary patterns $T_i = t_{i,0}t_{i,1}\dots t_{i,n-1}$ and $T_j = t_{j,0}t_{j,1}\dots t_{j,n-1}$. Property 4 of this measure states that the components are related according to the following inequality: $HD_1 \geq HD_2 \geq HD_3 \geq \dots \geq HD_n$, where $HD_r = HD_r[T_i(r), T_j(r)]$. According to Definition 2, the patterns $T_i(r)$ and $T_j(r)$ represent the binary patterns T_i and T_j in a base - 2^r numerical system consisting of 2^r distinct characters.

Based on Property 4 of the new measure of dissimilarity $MD(T_i, T_j)$, we formulate a statement that serves as the foundation for generating controlled random tests with a small number q of test patterns while maintaining a given $minHD(T_i, T_j)$ value.

Statement 1. A controlled random test consisting of $q = 2^r$ binary patterns, where $r \in \{1,2,\dots,n\}$, is the minimum value of r for which $HD_r[T_i(r), T_j(r)] = maxHD_r[T_i(r), T_j(r)]$ for all $i \neq j \in \{0,1,\dots,q-1\}$ and $n \mod r = 0$ has $minHD(T_i, T_j) = n/r$.

The limited number of test patterns, $q = 2^r$, is determined by the restricted number of characters in the alphabet, which is also equal to 2^r, in which the test patterns $T_i(r) = t_{i,0}(r)t_{i,1}(r)\dots t_{i,n/r-1}(r)$ and $T_j(r) = t_{j,0}(r)t_{j,1}(r)\dots t_{j,n/r-1}(r)$ are represented. Only in this case can the characters at the same positions in all q test patterns assume different values without repetition. This is the necessary condition for achieving the maximum value $HD_r[T_i(r), T_j(r)]$ of the Hamming distance for all pairs of test patterns $T_i(r)$ and

$T_j(r)$, where $i \neq j \in \{0, 1, \ldots, q-1\}$. To illustrate the meaning of this statement, let us consider the following example of a controlled random test.

Example 2. *In the case of $n = 6$, the controlled random test consisting of $q = 4$ patterns has the following form in binary ($r = 1$), quaternary ($r = 2$), and octal ($r = 3$) number systems (see Table 7).*

As can be seen from Table 7, there are no repeating characters in any digit of the quaternary and octal representations of the test patterns. This indicates that in both cases, the Hamming distance between the test patterns, according to Equation (1), takes its maximum values. Indeed, for any two patterns T_i and T_j in the test, $HD_2[T_i(2), T_j(2)] = maxHD_2[T_i(2), T_j(2)] = n/2 = 3$, as well as $HD_3[T_i(3), T_j(3)] = maxHD_3[T_i(3), T_j(3)] = n/3 = 2$. Moreover, in the quaternary case, all four characters (0, 1, 2, and 3) are used in each digit of the test patterns without repetition.

Following the above statement, we can conclude that a test consisting of $q = 2^2$ binary patterns with a minimum value of $r = 2$, for which $HD_2[T_i(2), T_j(2)] = maxHD_2[T_i(2), T_j(2)]$ for all $i \neq j \in \{0, 1, 2, 3\}$, satisfies the condition $HD(T_i, T_j) \geq minHD(T_i, T_j) = n/r = 6/2 = 3$. Indeed, as can be observed, $HD_1[T_0(1), T_1(1)] = HD_1[T_0(1), T_2(1)] = HD_1[T_1(1), T_3(1)] = HD_1[T_2(1), T_3(1)] = 3$ and $HD_1[T_0(1), T_3(1)] = HD_1[T_1(1), T_2(1)] = 6$. All values of $HD(T_i, T_j)$ are greater than or equal to 3, which confirms that the condition stated in the statement is fulfilled.

Table 7. Binary controlled random test for $n = 6$ and its representation in quaternary and octal notation.

$T_i(r)$	$t_{i,0}^{(1)}$	$t_{i,1}^{(1)}$	$t_{i,2}^{(1)}$	$t_{i,3}^{(1)}$	$t_{i,4}^{(1)}$	$t_{i,5}^{(1)}$	$t_{i,0}^{(2)}$	$t_{i,1}^{(2)}$	$t_{i,2}^{(2)}$	$t_{i,0}^{(3)}$	$t_{i,1}^{(3)}$
T_0	0	0	1	1	0	1	0	3	1	1	5
T_1	0	1	1	0	0	0	1	2	0	3	0
T_2	1	0	0	1	1	1	2	1	3	4	7
T_3	1	1	0	0	1	0	3	0	2	6	2

Based on the statement, we propose a formal procedure for constructing controlled random tests with $q = 2^r$ binary patterns and a given value of $minHD(T_i, T_j) \geq \lfloor n/r \rfloor$. The possible values of $minHD(T_i, T_j)$ depend on the number n of bits in the binary test patterns T_i and T_j. For example, for $n = 16$, the possible test configurations with a given value of $minHD(T_i, T_j) \geq \lfloor n/r \rfloor$ and the number q of test patterns are presented in Table 8.

Table 8. Dependence between the number of bits $n = 16$ of binary patterns and $minHD(T_i, T_j)$.

r	2	3	4	5	6	7	8	9	10	...	15	16
$minHD(T_i, T_j) = \lfloor 16/r \rfloor$	8	5	4	3	2	2	2	1	1	...	1	1
q	4	8	16	32	64	128	256	512	1024	...	32,768	65,536

As can be seen from Table 8, the fixed value n of the test pattern bit length determines the possible values of $minHD(T_i, T_j)$ for which a test can be constructed based on the statement. Naturally, the most interesting cases are those where $minHD(T_i, T_j)$ attains acceptably large values, which correspond to the smallest values of r.

The algorithm for generating binary controlled random tests with a given Hamming distance consists of the steps outlined in Algorithm 1. An extension of this algorithm (Algorithm 1) can involve selecting not necessarily consecutive r bits of the patterns but any arbitrary r out of n bits to specify the binary code of characters. The only limitation is the requirement to select non-overlapping blocks of r bits.

Algorithm 1 Generation of Binary Controlled Random Tests with a Given Hamming Distance

Input data: the size n of the test patterns (in bits) and the required value of $Rec\_minHD(T_i, T_j)$, which denotes the minimum Hamming distance between any two test patterns.

1. From the inequality
$$Rec\_minHD(T_i, T_j) \leq \left\lfloor \frac{n}{r} \right\rfloor,$$
determine the largest possible value of $r \in \{1, 2, \ldots, n\}$. Based on this condition, compute the number of test patterns as $q = 2^r$. The minimum Hamming distance between any two patterns will then satisfy
$$minHD(T_i, T_j) \geq \left\lfloor \frac{n}{r} \right\rfloor, \quad \text{for all } i \neq j,\ i, j \in \{0, 1, \ldots, 2^r - 1\}.$$
2. Assign the first r bits of each test pattern $T_0, T_1, \ldots, T_{q-1}$ to distinct binary codes selected randomly from an alphabet of 2^r possible r-bit combinations. Each code is assigned without repetition, starting from T_0 to T_{q-1}. As a result, each pattern contains in the first r bits a unique binary combination corresponding to one of the 2^r possible codes.
3. Repeat step 2 for the next $\left\lfloor \frac{n}{r} \right\rfloor - 1$ blocks of r bits. In each iteration, assign the next r bits of all test patterns (e.g., bits r to $2r - 1$, $2r$ to $3r - 1$, etc.) to new sets of unique binary codes of length r, again selected randomly without repetition.
4. If the pattern length n is not divisible by r, i.e., $n - \left\lfloor \frac{n}{r} \right\rfloor \cdot r > 0$, then assign the remaining bits randomly for all test patterns.

The described algorithm generates test patterns with a guaranteed minimum Hamming distance between any pair of test patterns. By partitioning each test pattern into independent r-bit blocks and ensuring that each block contains a unique binary code selected from a maximally distinct set, the method guarantees that the resulting test set is both compact and diverse. The final step introduces randomness in the unused bit positions, further enhancing the variability of the test without violating the distance constraint. It should be emphasized that the guaranteed minimum Hamming distance is determined solely by the disjoint allocation of unique codes in the complete r-bit blocks. When the pattern length n is not divisible by r, the remaining bits are filled by random padding. This step only affects the residual part of the patterns and does not reduce the guaranteed minimum Hamming distance between them. On the contrary, it adds additional variability to the generated tests while fully preserving the distance constraint.

The computational complexity of Algorithm 1 is $\mathcal{O}(q \cdot n)$, where $q = 2^r$ denotes the number of generated patterns and n is the pattern length. This is significantly more efficient than classical candidate–selection approaches, which usually require $\mathcal{O}(q^2 \cdot n)$ operations due to pairwise comparisons.

The following example demonstrates the operation of Algorithm 1 for a specific input configuration, highlighting the structure of the generated patterns and validating the achieved minimum Hamming distance.

Example 3. *Let the size n of the test patterns be 7, and let the required value of* $Rec\_minHD(T_i, T_j) = 3$.

1. *Based on the inequality* $Rec\_minHD(T_i, T_j) \leq \lfloor n/r \rfloor$, *we obtain* $r = 2$. *This is the largest value of r for which the inequality holds:* $3 \leq \lfloor 7/2 \rfloor = 3$. *Therefore, the generated test T will consist of* $2^r = 4$ *patterns,* T_0, T_1, T_2, T_3, *with a guaranteed minimum Hamming distance* $minHD(T_i, T_j) \geq 3$.
2. *The first two bits* $t_{i,0}$ *and* $t_{i,1}$ *of the test patterns are assigned binary values corresponding to four distinct characters from the quaternary alphabet: 00, 01, 10, and 11. These binary*

codes are assigned randomly, without repetition, starting from T_0 to T_3. As a result, each test pattern contains a unique 2-bit prefix: 10, 11, 00, and 01.

3. *Step 2 is repeated $\lfloor 7/2 \rfloor - 1 = 2$ times for the next two r-bit blocks, i.e., $(t_{i,2}, t_{i,3})$ and $(t_{i,4}, t_{i,5})$. For each block, values are assigned using random permutations of the quaternary alphabet.*
4. *The remaining bit $t_{i,6}$, since $7 - \lfloor 7/2 \rfloor \times 2 = 1$, is assigned randomly for all patterns.*

The resulting controlled random test is presented in Table 9.

Table 9. Controlled random test with $minHD(T_i, T_j) = 3$.

T	$t_{i,0}$	$t_{i,1}$	$t_{i,2}$	$t_{i,3}$	$t_{i,4}$	$t_{i,5}$	$t_{i,6}$
T_0	1	0	0	0	0	1	1
T_1	1	1	1	1	0	0	0
T_2	0	0	0	1	1	1	1
T_3	0	1	1	0	1	0	0

All pairwise Hamming distances between patterns satisfy the required minimum value:

$$HD(T_0, T_1) = 5, \quad HD(T_0, T_2) = 3, \quad HD(T_0, T_3) = 6,$$
$$HD(T_1, T_2) = 6, \quad HD(T_1, T_3) = 3, \quad HD(T_2, T_3) = 5.$$

Since all values are greater than or equal to 3, the condition $Rec\_minHD(T_i, T_j) \geq 3$ is fulfilled.

It should be noted that the proposed algorithm was intentionally formulated in the binary domain, since it directly corresponds to the digital world at the low level of hardware implementation, where the binary alphabet is natural and fundamental. Although the theoretical framework allows for the use of higher-radix alphabets and non-binary symbols, our focus on binary patterns reflects the practical context of memory testing and built-in self-test environments. Extending the method to real non-binary alphabets remains an interesting direction for future research.

5. Experimental Investigation

This section presents a comparative analysis of the effectiveness of two types of tests: controlled random tests with a given Hamming distance (CRTs), generated using the proposed algorithm, and standard random patterns. The comparison is conducted in the context of their ability to detect multicell faults, particularly Pattern-Sensitive Faults (PSFs) occurring in RAM. Due to the size of the test patterns and the vast number of their permutations, the comparisons are based on the average values obtained from the generated test collections.

The first test collection consists of patterns generated using the proposed algorithm, based on the controlled random test generation method described earlier. Using this approach, a controlled random test of length 1024 bits was generated with $minHD(T_i, T_j) = 256$. For the input parameters $n = 1024$ and $Rec\_minHD(T_i, T_j) = 256$, the value of r was determined to be 4, resulting in the generation of $2^r = 16$ patterns per test. The average value of the metric $MD_{\text{Total}}(T_i, T_j)$ (Equation (5)) for these patterns is 287,092, with a standard deviation of 34.54.

In contrast, the second test collection consists of 16 purely random patterns of the same size. The average value of $MD_{\text{Total}}(T_i, T_j)$ for these test sets is 273,815, with a standard deviation of 871.

The basic statistical parameters of the generated test collections are summarized in Table 10.

Table 10. Comparison of statistical parameters between CRT and random tests.

Parameter	CRT	Random Tests
Number of tests in collection	1000	2000
Patterns per test	16	16
Bits per pattern	1024	1024
Average $MD_{Total}(T_i, T_j)$	261,052.62	248,361.75
Standard deviation	34.54	455.55
Coefficient of variation (CV)	0.0132%	0.1834%
Relative error (E_{rel})	0.00082%	0.00804%
Confidence level	95%	95%

The generated test collections confirm their statistical reliability, as evidenced by low relative errors (E_{rel}) and consistent coefficient of variation (CV) values.

Similar test collections of 1024-bit size and comparable statistical parameters were generated for r = 2, 3, and 5 and will be used in further analyses.

In Table 11, the detailed results for individual values of $HD_1, HD_2, \ldots, HD_8$ and MD for $r = 4$ are compared.

Table 11. Average results for $HD_1, HD_2, \ldots, HD_8$ and $MD_{Total}(T_i, T_j)$ for $r = 4$.

Test	HD_1	HD_2	HD_3	HD_4	HD_5	HD_6	HD_7	HD_8	$MD_{Total}(T_i, T_j)$
CRT	65,536	49,152	37,711	30,720	24,436	20,520	17,616	15,360	287,092
Random	61,316	45,978	35,612	28,637	23,824	20,211	17,477	15,295	274,354
% Diff	6.88%	6.90%	5.87%	7.27%	2.57%	1.53%	0.80%	0.42%	4.64%

The results presented in Table 11 highlight the comparative performance of the analyzed CRT and standard random tests across individual HD_n values (HD_1 to HD_8) and the overall metric $MD_{Total}(T_i, T_j)$ for $r = 4$. On average, the CRT outperforms random tests across all tested HD values, with percentage differences ranging from 0.42% to 7.27%. The highest difference (7.27%) was observed for HD_4, which aligns with the parameter $r = 4$ used in generating the CRTs. This correlation underscores the effectiveness of the proposed algorithm in targeting specific test conditions based on the selected r parameter. Although the percentage differences in Table 11 may appear moderate, they are systematic across all evaluated parameters. More importantly, the subsequent experiments (Table 12 and Figure 1) confirm that these differences translate into noticeable improvements in memory fault coverage.

In the next set of experiments, conducted in a simulation environment, the focus was on evaluating the effectiveness of test patterns generated using the proposed algorithm in detecting multicell RAM faults. Multicell memory faults, such as Pattern-Sensitive Faults (PSFs), involve dependencies between any k out of N memory cells (N being the memory size). These faults are triggered when specific binary patterns are present in the related cells or when particular transitions occur based on predefined conditions. Consequently, effective detection of such faults requires generating the largest possible number of binary patterns during testing. These patterns activate the faults and enable their detection.

The simulations analyzed groups consisting of k memory cells for $k = 2 \ldots 10$. For each group, up to 2^k distinct k-bit binary patterns (i.e., values ranging from 0 to $2^k - 1$) could potentially appear. The objective was to determine the average number of unique k-bit patterns generated during a march test, with the memory being initialized in each iteration using test patterns from the CRT with a given Hamming distance. The obtained results were compared with the results for random tests presented in Table 8.12 in [38].

Each simulation-based test consisted of a specific number of iterations, determined by the value of r: 4 iterations for $r = 2$, 8 iterations for $r = 3$, and 16 iterations for $r = 4$. During each iteration, the simulated memory was initialized with a given test pattern from the analyzed set, followed by the execution of a transparent version of the MATS+ memory test. Throughout the simulation, the memory model was monitored, and the number of unique k-bit binary patterns observed in individual groups of k-cells was recorded. The results are presented in Table 12.

Table 12. Fault coverage [%] comparison for random tests and CRTs with a given Hamming distance for different memory fault sizes k and different numbers of iterations (2^r).

k	3	4	5	6	7	8	9	10
$r = 2$ *(4 iterations)*								
random [38]	93.74	77.67	56.42	37.07	22.75	13.34	7.59	4.31
CRT with given HD_r	97.27	81.49	58.42	37.79	22.93	13.35	7.58	4.22
$r = 3$ *(8 iterations)*								
random [38]	99.69	95.01	81.03	60.41	40.33	24.91	14.61	8.28
CRT with given HD_r	99.98	97.52	84.58	62.89	41.54	25.38	14.77	8.33
$r = 4$ *(16 iterations)*								
random [38]	100.00	99.75	96.47	84.39	64.44	43.64	27.18	15.88
CRT with given HD_r	100.00	99.95	97.85	86.78	66.36	44.64	27.55	16.03

Based on the results presented in Table 12, it can be concluded that CRTs consistently achieve better results than random tests in most cases. The difference is most noticeable for lower values of r and k, where the CRT outperforms random tests by several percentage points. For instance, for $r = 2$ and $k = 3$, the CRT achieves a fault coverage of 97.27%, while random tests reach 93.74%. Fault coverage decreases as the value of k increases. This is expected, since the number of possible binary combinations 2^k grows exponentially, making full coverage harder to achieve. However, the results in Table 12 indicate that CRTs perform slightly better for larger k compared to random tests, highlighting the greater ability of a CRT to generate diverse test patterns.

In the final experiment, the average number of unique k-bit test patterns generated in arbitrary groups of k out of N memory cells using the proposed algorithm was compared with that obtained using traditional CRT generation methods, including native antirandom tests [13], concatenated antirandom tests [13], and STPG [40]. The comparison was carried out for fault groups of size $k = 3$, using tests generated for $r = 3$ (i.e., 8 iterations). During the simulation, the number of distinct k-bit patterns generated in each iteration was recorded to assess the performance of the proposed method relative to the standard techniques. The outcomes of this analysis are presented in Figure 1, which illustrates the differences in the number of generated k-bit patterns across the tested methods.

The results show that the CRT method with a given Hamming distance consistently outperforms other test generation methods in terms of fault coverage, with one exception in the second iteration, where it achieves a slightly lower result (76.13%) compared to native antirandom (77.77%). However, starting from the third iteration, the CRT surpasses all other methods, demonstrating a faster increase in fault coverage (e.g., between iterations 2 and 3, the CRT rises from 76.13% to 90.05%). In the later iterations (7 and 8), the CRT approaches near-complete fault coverage, reaching 99.92% and 99.99%, respectively. Although the differences between the CRT and other methods diminish with a higher number of iterations, the CRT consistently demonstrates superior effectiveness, confirming its ability to generate diverse and efficient test patterns even in advanced stages of testing.

In summary, the experimental evaluation demonstrates that the proposed method consistently provides superior results compared to both purely random tests and classical controlled random tests. The improvements are systematic across all examined cases, particularly in terms of fault coverage and test diversity, while being achieved with significantly reduced computational effort.

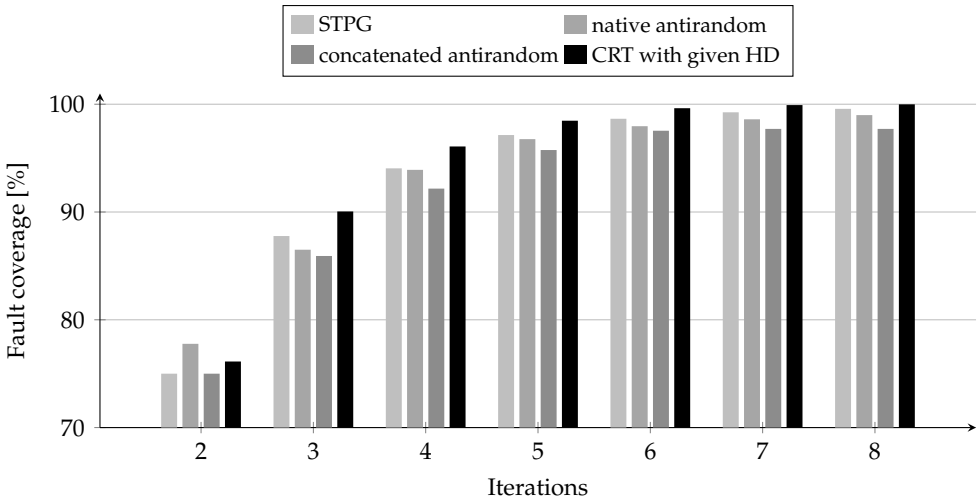

Figure 1. Fault coverage comparison for different test generation methods for $k = 3$ and $r = 3$ (8 iterations).

6. Conclusions

This paper presented a method for generating controlled random tests with a given Hamming distance, aimed at improving the diversity and effectiveness of test sets used in computing systems, particularly memory devices. A new dissimilarity measure was introduced, based on Hamming distances calculated for binary patterns represented in various numeral systems. This extended measure allows for a more detailed assessment of differences between patterns compared to the classical Hamming distance alone.

We proposed an algorithm that generates test sets with a predefined minimum Hamming distance, without selecting patterns from large pools of candidates. This approach reduces computational effort while ensuring sufficient diversity in the generated patterns.

The effectiveness of the proposed method was evaluated through a series of comparative experiments. The results showed that the generated tests outperform not only purely random test sets but also traditional controlled random tests (CRTs) in several aspects. Specifically, tests created using the proposed method achieved higher total dissimilarity values and better coverage of multicell memory faults, particularly for lower numbers of iterations and smaller fault group sizes. Although some improvements observed in the experiments may appear moderate, they are systematic across all evaluated cases. More importantly, the obtained results demonstrate that these differences translate into tangible practical benefits, as the proposed method consistently achieves higher fault coverage than random and classical controlled random testing, especially in scenarios with smaller fault groups and lower iteration counts.

These results suggest that the method may be a practical alternative in contexts where test diversity and efficiency are important. Future work may include extending the approach to more complex fault models or exploring its use in different types of systems.

Author Contributions: Conceptualization, V.N.Y.; methodology, V.N.Y.; software, I.M.; validation, V.N.Y., I.M. and M.K.; formal analysis, V.N.Y. and I.M.; investigation, V.N.Y. and I.M.; writing—original draft preparation, V.N.Y. and I.M.; writing—review and editing, V.N.Y., I.M. and M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by grant WZ/WI-ITI/3/2023 from the Faculty of Computer Science at Bialystok University of Technology, Ministry of Science and Higher Education, Poland.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Malaiya, Y.K.; Yang, S. The Coverage Problem for Random Testing. In Proceedings of the IEEE International Test Conference (ITC '84), Philadelphia, PA, USA, 16–18 October 1984; pp. 237–245.
2. Duran, J.W.; Ntafos, S.C. An Evaluation of Random Testing. *IEEE Trans. Softw. Eng.* **1984**, *10*, 438–444. [CrossRef]
3. Arcuri, A.; Iqbal, M.Z.; Briand, L. Random Testing: Theoretical Results and Practical Implications. *IEEE Trans. Softw. Eng.* **2012**, *38*, 258–277. [CrossRef]
4. Renfer, G. Automatic program testing. In Proceedings of the 3rd Conference of the Computing and Data Processing Society of Canada, Toronto, ON, USA, 2–3 June 1962.
5. Anand, S.; Burke, E.K.; Chen, T.Y.; Clark, J.; Cohen, M.B.; Grieskamp, W.; Harman, M.; Harrold, M.J.; Mcminn, P. An Orchestrated Survey of Methodologies for Automated Software Test Case Generation. *J. Syst. Softw.* **2013**, *86*, 1978–2001. [CrossRef]
6. Yarmolik, S.V.; Yarmolik, V.N. Controlled random tests. *Autom. Remote Control* **2012**, *73*, 1704–1714. [CrossRef]
7. Huang, R.; Sun, W.; Xu, Y.; Chen, H.; Towey, D.; Xia, X. A survey on adaptive random testing. *IEEE Trans. Softw. Eng.* **2019**, *47*, 2052–2083. [CrossRef]
8. Bardell, P.H.; McAnney, W.H.; Savir, J. *Built-In Test for VLSI: Pseudorandom Techniques*; John Wiley & Sons: New York, NY, USA, 1987.
9. Das, D.; Karpovsky, M. Exhaustive and Near-Exhaustive Memory Testing Techniques and their BIST Implementations. *J. Electron. Test.* **1997**, *10*, 215–229. [CrossRef]
10. McCluskey, E.J. Verification Testing—A Pseudoexhaustive Test Technique. *IEEE Trans. Comput.* **1984**, *C-33*, 541–546. [CrossRef]
11. Fujiwara, H. *Logic Testing and Design for Testability*; MIT Press: Cambridge, MA, USA, 1985. [CrossRef]
12. Karpovsky, M.G.; van de Goor, A.J.; Yarmolik, V.N. Pseudo-Exhaustive Word-Oriented DRAM Testing. In Proceedings of the European Design and Test Conference (ED&TC), Paris, France, 6–9 March 1995; pp. 126–132. [CrossRef]
13. Malaiya, Y.K. Antirandom Testing: Getting the Most out of Black-Box Testing. In Proceedings of the International Symposium on Software Reliability Engineering (ISSRE), Toulouse, France, 24–27 October 1995; pp. 86–95. [CrossRef]
14. Wu, S.H.; Malaiya, Y.K.; Jayasumana, A.P. Antirandom vs. pseudorandom testing. In Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors, Austin, TX, USA, 5–7 October 1998; p. 221.
15. Wu, S.H.; Jandhyala, S.; Malaiya, Y.K.; Jayasumana, A.P. Antirandom Testing: A Distance-Based Approach. *VLSI Des.* **2008**, *2008*, 1–9. [CrossRef]
16. Yin, H. Antirandom Test Patterns Generation Tool. In *Technical Report CS-98-101*; Computer Science Department, Colorado State University: Fort Collins, CO, USA, 1996; Fall 1996.
17. von Mayrhauser, A.; Bai, A.; Chen, T.; Anderson, C.; Hajjar, A. Fast Antirandom (FAR) Test Generation. In Proceedings of the 3rd IEEE International Symposium on High-Assurance Systems Engineering, Washington, DC, USA, 13–14 November 1998; HASE '98, pp. 262–269.
18. Sahari, M.S.; A'ain, A.K.; Grout, I.A. Scalable Antirandom Testing (SAT). *Int. J. Innov. Sci. Mod. Eng.* **2015**, *3*, 33–35.
19. Bodean, G.; Bodean, D.; Labunetz, A. New Schemes for Self-Testing RAM. In Proceedings of the Design, Automation and Test in Europe (DATE), Munich, Germany, 7–11 March 2005; Volume 2, pp. 858–859.
20. Chen, T.Y.; Leung, H.; Mak, I.K. Adaptive Random Testing. In Proceedings of the 9th Asian Computing Science Conference, Chiang Mai, Thailand, 8–10 December 2004; pp. 320–329.
21. Zhou, Z.Q. Using Coverage Information to Guide Test Case Selection in Adaptive Random Testing. In Proceedings of the 34th Annual IEEE Computer Software and Applications Conference Workshops (COMPSACW), Seoul, Republic of Korea, 19–23 July 2010; pp. 208–213. [CrossRef]
22. Jiang, B.; Zhang, Z.; Chan, W.K.; Tse, T.H. Adaptive Random Test Case Prioritization. In Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE), Auckland, New Zealand, 16–20 November 2009; pp. 233–244.
23. Chan, K.P.; Chen, T.Y.; Towey, D. Good Random Testing. In Proceedings of the 9th Ada-Europe International Conference on Reliable Software Technologies, Palma de Mallorca, Spain, 14–18 June 2004; Llamosí, A., Strohmeier, A., Eds.; 2004; pp. 200–212. [CrossRef]

24. Chan, K.P.; Chen, T.Y.; Towey, D. Restricted Random Testing. In Proceedings of the 7th European Conference on Software Quality, Helsinki, Finland, 9–13 June 2002; Kontio, J.; Conradi, R., Eds.; pp. 321–330. [CrossRef]
25. Chan, K.P.; Chen, T.Y.; Towey, D., Normalized Restricted Random Testing. In *Lecture Notes in Computer Science, Proceedings of the 8th Ada-Europe International Conference on Reliable Software Technologies (Ada-Europe 2003), LNCS, Toulouse, France, 16–20 June 2003* ; Springer: Toulouse, France, 2003; Volume 2655, pp. 368–381. [CrossRef]
26. Xu, S.; Chen, J. Maximum Distance Testing. In Proceedings of the Asian Test Symposium, Guam, GU, USA, 18–20 November 2002; pp. 15–20.
27. Kuo, F. An Indepth Study of Mirror Adaptive Random Testing. In Proceedings of the Ninth International Conference on Quality Software, QSIC 2009, Jeju, Republic of Korea, 24–25 August 2009; Choi, B., Ed.; IEEE Computer Society: Washington, DC, USA, 2009; pp. 51–58.
28. Xu, S. Orderly Random Testing for Both Hardware and Software. In Proceedings of the 14th IEEE Pacific Rim International Symposium on Dependable Computing, Washington, DC, USA, 15–17 December 2008; pp. 160–167.
29. Nikravan, E.; Parsa, S. Hybrid adaptive random testing. *Int. J. Comput. Sci. Math.* **2020**, *11*, 209–221. [CrossRef]
30. Tappenden, A.; Miller, J. A Novel Evolutionary Approach for Adaptive Random Testing. *IEEE Trans. Reliab.* **2009**, *58*, 619–633. [CrossRef]
31. Grindal, M.; Offutt, J.; Andler, S.F. *Combination Testing Strategies—A Survey*; Technical Report ISE-TR-04-05, GMU Technical Report; George Mason University: Fairfax, VA, USA, 2004.
32. Chen, T.Y.; Kuo, F.C.; Merkel, R.G.; Tse, T.H. Adaptive Random Testing: The ART of test case diversity. *J. Syst. Softw.* **2010**, *83*, 60–66. [CrossRef]
33. Feldt, R.; Poulding, S.; Clark, D.; Yoo, S. Test Set Diameter: Quantifying the Diversity of Sets of Test Cases. In Proceedings of the IEEE International Conference on Software Testing, Verification and Validation (ICST), Chicago, IL, USA, 11–15 April 2016; pp. 223–233. [CrossRef]
34. Arcuri, A.; Briand, L.C. Adaptive random testing: An illusion of effectiveness? In Proceedings of the 20th International Symposium on Software Testing and Analysis (ISSTA 2011), Toronto, ON, Canada, 17–21 July 2011; pp. 265–275. [CrossRef]
35. Hamming, R.W. Error detecting and error correcting codes. *Bell Syst. Tech. J.* **1950**, *29*, 147–160. [CrossRef]
36. Peterson, W.W.; Weldon, E.J. *Error-Correcting Codes*, 2nd ed.; MIT Press: Cambridge, MA, USA, 1972.
37. Plotkin, M. Binary codes with specified minimum distance. *IIRE Trans. Inf. Theory* **1960**, *6*, 445–450. [CrossRef]
38. Mrozek, I. *Multi-Run Memory Tests for Pattern Sensitive Faults*; Springer International Publishing: Cham, Switzerland, 2019. [CrossRef]
39. Mrozek, I.; Yarmolik, V.N. Optimal Controlled Random Tests. In *Proceedings of the Computer Information Systems and Industrial Management: 16th IFIP TC8 International Conference, CISIM 2017, Białystok, Poland, 16–18 June 2017*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10244, pp. 27–38. [CrossRef]
40. Yiunn, D.B.Y.; Bin A'ain, A.K.; Khor Ghee, J. Scalable test pattern generation (STPG). In Proceedings of the IEEE Symposium on Industrial Electronics Applications (ISIEA '10), Penang, Malaysia, 3–5 October 2010; pp. 433–435. [CrossRef]