Hierarchical Clustering Driven Test Case Selection in Digital Circuits

Conor Ryan[1][a], Meghana Kshirsagar[1][b], Krishn Kumar Gupt[2][c], Lukas Rosenbauer[3] and Joseph P. Sullivan[2][d]

[1] *Biocomputing and Development System Lab, University of Limerick, Ireland*

[2] *Department of Electrical & Electronic Engineering, Limerick Institute of Technology, Ireland*

[3] *BSH Home Appliances, Im Gewerbepark B10, Regensburg, Germany*

Keywords: Combinational Circuit, Functional Testing, Hamming Distance, Hierarchical Clustering, Machine Learning, Test Case, Unsupervised Learning.

Abstract: The quality assurance of circuits is of major importance as the complexity of circuits is rising with their capabilities. Thus a high degree of testing is required to guarantee proper operation. If, on the other hand, too much time is spent in testing then this prolongs development time. The work presented in this paper proposes a methodology to select a minimal set of test cases for validating digital circuits with respect to their functional specification. We do this by employing hierarchical clustering algorithms to group test cases using a hamming distance similarity measure. The test cases are selected from the clusters, by our proposed approach of distance-based selection. Results are tested on the two circuits viz. Multiplier and Galois Field multiplier that exhibit similar behaviour but differ in the number of test cases and their implementation. It is shown that on small fraction values, distance-based selection can outperform traditional random-based selection by preserving diversity among the chosen test cases.

1 INTRODUCTION

Testing is a crucial part of hardware design and manufacturing process. It evaluates the functionality of the *Circuit Under Test (CUT)* and ensures it fully matches the device specification and is fault free. In order to ensure a top quality device, detailed conditions must be defined that are specific to the expected behavior of the design in terms of inputs and expected outputs known as a test case[1] *(tc)*. A test case is a set of possible inputs, various execution conditions, expected outputs and testing procedures (Sapna and Mohanty, 2010). It helps in improving design quality by providing good test coverage (Gupt et al., 2021b) and hence decreases the post design maintenance budgets. Due to process variation in the manufacturing of digital circuits, it is impossible to achieve a 100% production yield and it is common for testing to be performed throughout the manufacturing process. Typically it is done in a clean room at Probe-Stage before packaging, and again at Final-Test after packaging. The equipment use for this testing is tremendously expensive and every test case adds cost to the final product as well as causing process delays. In this environment it is vital that testing is as efficient as possible with good coverage at minimum cost. This is every more important as the circuit's complexity increases, when testing can become a key constraint or a bottleneck in manufacture (Bushnell and Agrawal, 2002).

For a given CUT, testing methods can be summarised as follows: (1) Define test objective; (2) Identifying a set of test cases; (3) Initiating CUT with test case; and (4) Result analysis via simulation (Gupt et al., 2021b). One fundamental testing approach involves running through all possible test cases to comprehensively cover the CUT's functionality; which is known as exhaustive testing (McCluskey, 1987). The problem can become intractable as the volume of test cases increase with devices complexity. As the CUT's complexity and requirements evolve, the volume of test cases increases explosively making exhaustive testing impractical as it consumes substantial time and resources. Hence, optimisation and reduction of test cases become fundamental for the testing

[a] https://orcid.org/0000-0002-7002-5815
[b] https://orcid.org/0000-0002-8182-2465
[c] https://orcid.org/0000-0002-1612-5102
[d] https://orcid.org/0000-0003-0010-3715

[1] The keyword 'test case', 'test vectors', and 'test patterns' are used interchangeably throughout this paper.

process.

One approach is to select a subset of test cases (by using some algorithms/characteristics) from the entire set of the test case domain. The testing time of a CUT can be calculated as,

$$time(T) = I \times f \quad (1)$$

where $time(T)$ is the testing time, I denotes the number of input test vectors, and f is frequency/clock rate provided during testing. The actual problem arises with an increase in the volume of test cases.

The test cases crucial in identifying errors are just a fraction of the complete test data (Narciso et al., 2014), as shown in Figure 1. For example, in a simple

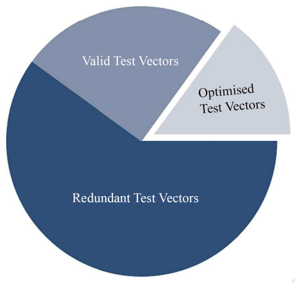


Figure 1: Distribution of test cases (Narciso et al., 2014).

combinational circuit with n inputs and m outputs, a total of 2^n input test vectors (I) would be required to perform exhaustive testing. Considering a circuit having primary inputs n as 32, if a total of 2^{32} test vectors are applied during an exhaustive test process at a frequency of 10MHz, it would take $2^{32} \times 10^{-7}$ seconds, which is 7.15 minutes of testing. For a 32 bit adder (without carry bit), it would take about 58.5k years to complete testing (Bushnell and Agrawal, 2002). Of course, this testing also involves comparing $m \times 2^n$ output signals against the expected output test vectors. However, as noted above, the testing of a CUT does not require exhaustively testing the entire test cases. Testing generally focuses on some selected set of test vectors and *corner cases*, which are cases a designer believes are most likely to cause failures. In such cases, it becomes important to select test cases that are well distributed over tc space while removing the test cases that are redundant and error free.

This work presents a test case selection approach using agglomerative hierarchical clustering (AHC) on combinational circuits. A *distance based selection (DBS)* methodology is used with the objective of minimizing the *selected test case (ts)* such that, $ts \subsetneq tc$. During *functional testing*, it is important to have diversity in ts for good test coverage. By diversity, we mean how dissimilar the test cases are. In order to maintain diversity, the proposed DBS algorithm performs selection based on the distance between two given test cases. The diversity of the test cases selected using DBS is compared with a *random based selection (RBS)* approach.

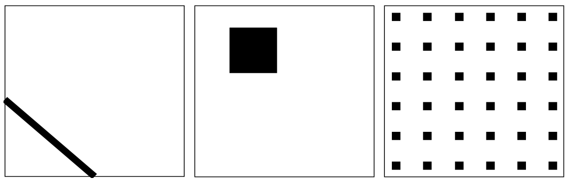
2 BACKGROUND

With the growing complexity of digital circuits, testing has always been a major concern in hardware design industries (Narciso et al., 2014). Exponential growth in the volume of test vectors can be time-consuming while testing. Hence, the selection of appropriate test cases becomes important.

Other than exhaustive testing (McCluskey, 1987), one fundamental testing approach involves simply selecting test cases in a random manner from the test case space (the set of all possible inputs), called random testing (RT) (Orso and Rothermel, 2014). Several sufficient methods have been developed for automatic test case generations. A method of generating test cases for some circuits used in Galois field (GF) and prime field operations is presented in (Gupt et al., 2021a,b), but test case minimization with high coverage is always desired, as the test-cost is measured in terms of the test data volume, and test time (Pradhan and Bhattacharya, 2021). A black-box test case selection method is presented in (Rosenbauer et al., 2021). There have been many concerns about the effect of minimizing test case volumes on fault detection. Some of the empirical studies are mentioned in (Wong et al., 1995; Rothermel et al., 1998).

(Thamarai et al., 2010) in their work, proposed to minimize the number of test cases based on the number of faults the test cases can detect. They proposed heuristic methods for identifying essential tests for circuits with smaller test sets. Motivated by the challenges of the ever difficulty in identifying defects in silicon chips due to the volume of test data, (Pradhan and Bhattacharya, 2021) in their comprehensive survey, presented insights on various machine learning methods for digital logic testing and diagnostics. (Muselli and Liberati, 2000) proposed hamming clustering as a solution to the classification of problems with binary inputs. In their work, they formulated clusters by comparing input patterns and their closeness to one another based on hamming distance. (Tamasauskas et al., 2012) evaluated the performance of hierarchical clustering methods for binary data type. They concluded that complete linkage, Flexible-beta, and Ward's methods have the best clustering performance on binary data. (Bushnell and Agrawal, 2002) provides insights on algorithms such as Automatic test-pattern generation (ATPG) for generating patterns for structural testing of digital circuits which can find redundant or unnecessary circuit logic, and are useful to compare whether one circuit's implementation matches another circuit's implementation. They discuss how functional ATPG programs are useful to generate test patterns to completely exercise the circuit function. They also highlight the impracticability of performing exhaustive testing of a 64-bit ripple-carry adder having 129 inputs and 65 outputs, suggesting that such testing can be suitable for only small circuits, whereas modern-day circuits tend to have an exponential volume of test data.

(Chan et al., 1996) identified three patterns for error/failure causing test cases as shown in Figure 2.



(a) Strip pattern. (b) Block pattern. (c) Point pattern

Figure 2: Erroneous test patterns.

The outer boundaries represent borders of the input domain and the filled regions denote the failure-causing inputs. A *strip pattern* is when erroneous test cases form the shape of a narrow strip, Figure 2a. Failing test cases concentrated in a closed region are termed as *block pattern*, Figure 2b, while those widely dispersed across the input domain are categorised as *point pattern*, Figure 2c. Previous studies (Ammann and Knight, 1988; Bishop, 1993; White and Cohen, 1980) indicate that strip and block pattern are encountered more often than point pattern. So, a sample of test cases should contain both, quality to provide good test coverage, and ability to detect failure causing inputs.

3 METHODOLOGY

The proposed architecture for test case selection on digital circuits is shown in Figure 3. The hierarchical clustering algorithm, distance metric used in experimentation, and approaches for test case selection are discussed in the subsequent sections.

3.1 Hierarchical Clustering for Grouping Binary Data Test Cases

Clustering algorithms are a type of unsupervised machine learning method (Bindra et al., 2021). The hierarchical clustering method helps in processing the easy-to-acquire features based on certain similarity

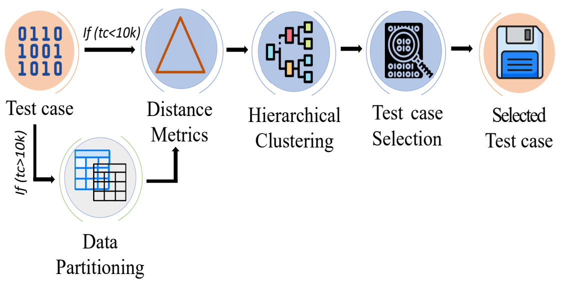


Figure 3: Key processes of the proposed methodology.

metrics and group processed test vectors into distinct clusters. The AHC, as discussed in Algorithm 1, starts with considering every test vector as a separate cluster.

```
Algorithm 1: Agglomerative hierarchical clustering algorithm.
  Input: Partition $p$ with $N$ sampled test cases
         as $T_1 \dots T_N$.
1 Compute distance matrix $disMtx$ for all T.
2 Let $T_1 \dots T_N$ be $N$ clusters.
3 repeat
4     Merge the two closest clusters.
5     Update the distance matrix.
6 until no. of cluster=1;
```

By employing the idea of complete linkage, (Equation 2), they are repeatedly merged together based on similarity, until all the test vectors are assigned to some group.

$$d(X,Y) = \max_{T_1 \in X, T_2 \in Y} d(T_1,T_2) \tag{2}$$

Where $d(T_1,T_2)$ is the distance between test vectors $T_1 \in X$ and $T_2 \in Y$; and X, Y are two sets (clusters) of test vectors.

In this work, the Hamming distance (Hamming, 1950) is used as similarity measure, as,

$$d_H(T_1,T_2) = \sum_{i=1}^{n} |T_{1_i} - T_{2_i}| \tag{3}$$

where T_1 and T_2 and two test vectors with n binary bits. However, in order to get meaningful information while selecting distance metrics applied to test vectors $(T_1,T_2,T_3,\dots,T_n)$, the following conditions (Akman et al., 2019) are adhered to:

1. *distance* $(T_i,T_j) =$ *distance* (T_j,T_i),
2. *distance* $(T_i,T_j) \geq 0$,
3. *distance* $(T_i,T_j) = 0$ *iff* T_i is same as T_j.

Here i, j $\geq$ 1. Condition (1) is symmetry, i.e. the distance between two test vectors should always be the same irrespective of the order used in measurement. Condition (2) restricts that the distance between two test vectors should always be non-negative, and condition (3) enforces that distance from a test vector to itself is always zero.

The result is visualized in the form of a tree-based structure, dendrogram as shown in Figure 4 where T is the test vectors.

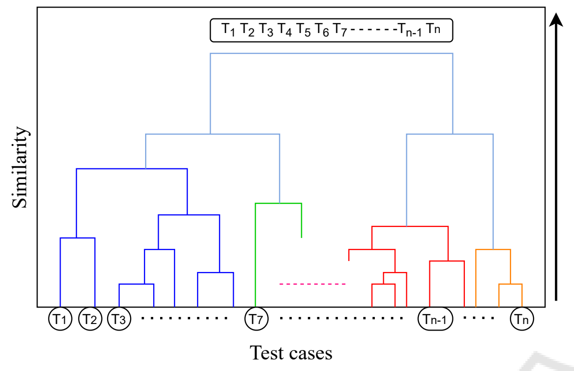

T_1 T_2 T_3 T_4 T_5 T_6 T_7 - - - - - -T_{n-1} Tn
Similarity
T1
T2
T3
T7
Tn-1
Tn
Test cases

Figure 4: Dendrogram obtained with AHC.

The number of clusters depends on the level (height) at which we cut the dendrogram. In our experimental setup we obtain clusters by cutting the dendrogram at heights 65% and 75%.

3.2 Methods for Test Case Selection

After clustering of test cases, the next objective is to obtain a reduced number of test cases while maintaining diversity. Hence the test case selection is done on the input vectors of underlying digital circuits. We select the test cases following random based selection and distance based selection strategies as discussed below.

3.2.1 Random based Selection

A random based selection is a popular approach used in testing where test cases are selected randomly and tested (Duran and Ntafos, 1984). This method selects a fraction of test cases from each cluster based on random generation. Experimentation is done on three different fractions values, such as 0.07, 0.10, and 0.15. The number of test cases to be selected from each cluster is obtained as,

$$s = f \times N \tag{4}$$

where s is number of test cases to be selected, N is number of test cases in the cluster and f is the fraction value applied. The approach is discussed in Algorithm 2.

Algorithm 2: Random based selection.

```
Input: $k$ clusters of $N$ test cases as $T_1 \ldots T_N$
Input: $f$ fraction to be selected
Output: Selected test cases $(ts)$
begin
    for $i$ in $k$ do
        $s \leftarrow N \times f$
        Temp = $\emptyset$
        for $j$ in $s$ do
            Select unique random $T$
            $uniqT \leftarrow T$
        end
        Temp$\leftarrow$ $uniqT$
    end
    $ts$ $\leftarrow$Temp
end
```

3.2.2 Distance based Selection

This approach begins with measuring hamming distance between all test cases within each cluster. The distances are then sorted in descending order, and we choose the test cases employing greedy strategy based on their distance. Thus, we select the two test cases corresponding to highest distance first, afterwards the second most distant test cases etc. as shown in Algorithm 3. (Note that the duplicate test cases are ignored during selection).

Algorithm 3: Distance based selection.

```
Input: $k$ clusters of $N$ test cases as $T_1 \ldots T_N$
Input: $f$ fraction to be selected
Output: Selected test cases (ts)
begin
    for $i$ in $k$ do
        s $\leftarrow N \times f$
        Get distance matrix $disMtx$ of $T_1 .. T_{N_i}$
        // by distance, descending order
        L = sort $\{(T_j, T_l) | 0 \leq j, l \leq N_i\}$
        Temp = $\emptyset$
        for $(a, b)$ in $L$ do
            // add most distant remaining tests
            Temp = Temp $\cup \{a, b\}$
            if $|Temp| == s$ then
                break
            end
        end
        $Temp \leftarrow (T_1 \ldots T_s)$
    end
    $ts \leftarrow Temp$
end
```

Three different fraction values for f at 0.07, 0.10 and 0.15 are used. The number of test cases to be selected is calculated using Equation 4. The selected test cases size (*Size(ts)*) is calculated as shown in Equation 5,

$$Size(ts) = \sum_{i=1}^{k} f \times N_i \tag{5}$$

where N_i is number of test vectors in i^{th} cluster, k is total number of clusters, and f is fraction value.

4 RESULTS AND DISCUSSION

In this section we present the digital circuits used in our experimentation and illustrate the clusters obtained after running the AHC algorithm on our circuits. We introduce a new performance measure, named diversity, that is useful to calculate the uniqueness and coverage of test cases within each cluster.

4.1 Experimental Setup

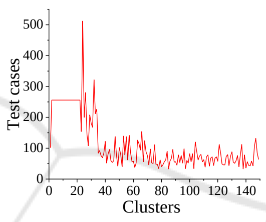
The test vectors used in the experiment are considered for *two* different circuits popular in the digital circuit domain, as discussed. A binary *multiplier* is a combinational logic circuit built-up using binary adders and is used to multiply two binary numbers. A *Galois field (GF) multiplier* is a finite field operational circuits used in cryptography (Benvenuto, 2012). An automatic test case generation approach for various Galois field arithmetic operational circuits was presented recently (Gupt et al., 2021b). A short description of both the benchmark circuits are mentioned in Table 1. Prior to the experiment, a complete set of test cases *(tc)* are generated using expert knowledge of each circuits' behaviour.

4.2 Complexity Measure

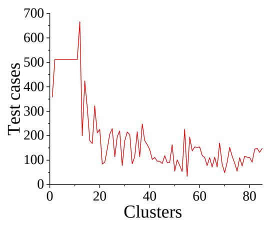
Since the distance matrix is stored in the RAM during hierarchical clustering, for 2^n test cases, space complexity being $O((2^n)^2)$ becomes too high. In addition to that, hierarchical clustering performs iterations while adding data to clusters, and updates the distance matrix in each iteration, hence the time complexity is $O((2^n)^3)$. To overcome this issue, instead of loading the entire distance matrix into memory, we follow the approach of splitting the test cases into different partitions (p). It is worth mentioning that the partitions are done in sequence, and after clustering the obtained clusters are merged in the same order. In this way, the total number of *selected test cases (ts)* representing the huge volume of *test cases (tc)* is the sum of samples obtained from clusters of every partition. Due to the high volume of test cases, we split *multiplier*'s test vector into 3 partitions, and *GF multiplier*'s test vectors in 11 partitions where each p have almost equal size.

4.2.1 Results of AHC

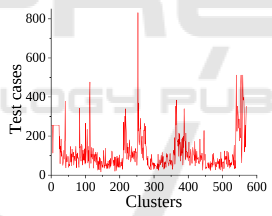
After merging the clusters obtained from all 3 partitions of the multiplier test cases, a total of 149 clusters (Figure 5a) and 85 clusters (Figure 5b) were obtained at 65% and 75% respectively, whereas the total number of clusters obtained from GF multiplier test cases are 569 (Figure 5c) and 306 (Figure 5d) at heights 65% and 75% respectively. The number of test cases per cluster is shown below.



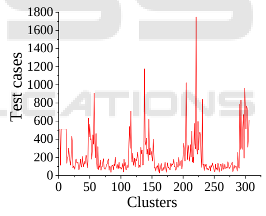
(a) Multiplier (H=65%)



(b) Multiplier (H=75%)



(c) GF Multiplier (H=65%)



(d) GF Multiplier (H=75%)

Figure 5: Clusters at different heights for the two circuits.

Figure 5 shows the number of clusters obtained for both the benchmark circuits' test cases, using the height parameter as 65% and 75%. After clustering, the test case samples are chosen from every distinct p, using RBS and DBS.

4.2.2 Diversity Measure

Test cases with a larger distance between them are more diverse relative to each other. To calculate the diversity of the selected test cases from each clusters $div_{ts}(C)$, we compute the distance matrix between them and do an average sum as given in Equation 6 & 7.

$$div_{ts}(C) = \frac{\sum_{i=1}^{s}\sum_{j=1}^{s} dist(T_i, T_j)}{s} \tag{6}$$

Table 1: Circuits used in experimentation.

Circuits	**Input (n)**	**tc**	**Description**
Multiplier	14	16384	7-bit binary multiplier implemented using basic longhand algorithm.
GF multiplier	16	65536	GF multiplication in GF(2^8) and *irreducible polynomial*=11100010.

Where s is the number of selected test cases. The diversity of ts can be calculated as,

$$div(ts) = \frac{\sum_{C=1}^{k} Div_{ts}(C)}{k} \tag{7}$$

where k is number of clusters from which the test cases are being selected.

Figure 6 shows the diversity comparison of the test cases selected across 85 clusters, obtained with height 75%, and *f=0.07* using RBS and DBS.

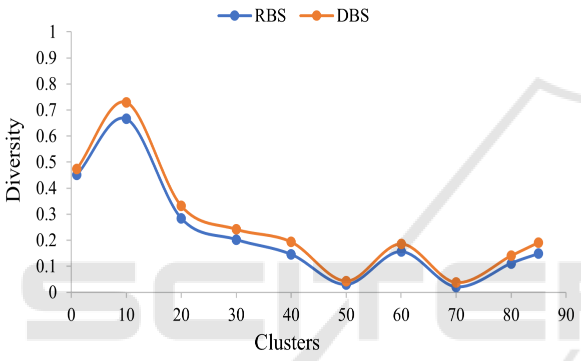


Figure 6: Comparative analysis of diversity scored for *ts* using RBS and DBS on multiplier circuit.

Figure 7 represents the diversity of RBS and DBS across 306 clusters for the GF multiplier test cases with *f=0.07* using RBS and DBS.

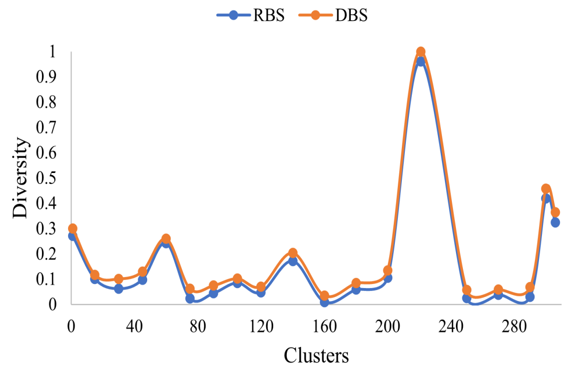


Figure 7: Comparative analysis of diversity scored for *ts* using RBS and DBS on GF multiplier circuit.

Due to the random nature of the RBS selection algorithm, five independent runs for test case selection on each cluster were done for all fraction values, and an average value is considered. The diversity measure for DBS shown is obtained using single run on each clusters as it produces same output. The diversity obtained on other fraction (f) value and height parameter (65%) have similar performance.

5 STATISTICAL VALIDATION

Test case selection based on two different selection approaches are performed on two widely used digital circuits. While using RBS, there are many chances of getting test cases that are close/similar to each other, DBS overcomes this problem as it selects test cases based on maximum distance. Along with having good diversity in ts, it also eases the selection of corner cases as test cases at two corners have maximum distance between them. For the performance analysis of the proposed method, we examine the null hypothesis as 'on average, the RBS selects test cases having better diversity than DBS for selected circuits'. To test the hypothesis, we rely on one sided Wilcoxon tests with a significance level of 0.05. The p-values are shown in Table 2. The row represent the *circuits' test case* on which the test case selection algorithms are performed using height parameters *(H)* of 65% and 75% during clustering, and the column *test case selection algorithm* to compare with. The p-value of the null hypothesis examines that the RBS with the given fractions f gives better diversity than DBS during test case selection for the given circuits.

Table 2: P-values for one-sided Wilcoxon tests.

Height	Circuit	RBS (0.07)	RBS (0.1)
65%	multiplier	4.08e-14	2.76e-10
	GF multiplier	4.085-12	2.58e-12
75%	multiplier	8.88e-16	6.66e-16
	GF multiplier	5.55e-16	0

The p-values are not only below our significance value but basically zero ($10^{-10} \approx 0$). Thus the likelihood that the null hypothesis is true given our data is zero. Hence we reject the null hypothesis and infer that DBS leads to more diverse test cases.

6 CONCLUSIONS

This paper investigated hierarchical clustering as a design approach for obtaining minimal set test cases for functional verification of combinational circuits. The authors propose a novel distance-based selection method for choosing a fraction of test cases from the clusters obtained as an output of the hierarchical clustering process. Furthermore, the quality of clusters is evaluated based on the score of the proposed diversity metric. The diversity score is computed as the ratio of an average value of diversity score across each individual cluster to the total number of test cases distributed across all clusters. The results indicate hierarchical clustering with the distance-based selection approach can be a promising strategy for obtaining a wide number of unique test cases with maximum coverage. In the preliminary work undergone in this paper, the proposed approach is performed on the circuits with specific behavior. For example, a multiplier does multiplication on all test cases. For circuits like an arithmetic logic unit (ALU) where a circuit's behavior changes by changing the control bits, test cases should be grouped on the basis of control bits before clustering and selection. In the future, the authors will investigate this approach for grouping test cases for sequential circuits, where several factors like the order of test vectors, previous state, etc. need to be considered. The extended work by the authors would also test the feasibility of this approach on other problem domains such as identifying minimal test cases for real-world regression datasets.

ACKNOWLEDGEMENTS

We express our sincere and heartfelt gratitude to all the reviewers, with special mention of reviewer #5, for their valuable comments and insights which has greatly helped towards organization and restructuring of the paper.

This work was supported with the financial support of the Science Foundation Ireland (SFI) grants #16/IA/4605.

REFERENCES

Akman, O., Comar, T., Hrozencik, D., and Gonzales, J. (2019). Chapter 11 - data clustering and self-organizing maps in biology. In Robeva, R. and Macauley, M., editors, *Algebraic and Combinatorial Computational Biology*, MSE/Mathematics in Science and Engineering, pages 351–374. Academic Press.

Ammann, P. and Knight, J. (1988). Data diversity: an approach to software fault tolerance. *IEEE Transactions on Computers*, 37(4):418–425.

Benvenuto, C. J. (2012). Galois field in cryptography. *University of Washington*, 1(1):1–11.

Bindra, P., Kshirsagar, M., Vaidya, G., Gupt, K. K., Ryan, C., and Kshirsagar, V. (2021). Insights into the advancements of Artificial Intelligence and Machine Learning, the present state of art, and future prospects: Seven decades of digital revolution.

Bishop, P. G. (1993). The variation of software survival time for different operational input profiles (or why you can wait a long time for a big bug to fail). In *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*, pages 98–107. IEEE.

Bushnell, M. L. and Agrawal, V. D. (2002). *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*, volume 17 of *Frontiers in Electronic Testing*. Springer US, Boston, MA.

Chan, F., Chen, T., Mak, I., and Yu, Y. (1996). Proportional sampling strategy: guidelines for software testing practitioners. *Information and Software Technology*, 38(12):775–782.

Duran, J. W. and Ntafos, S. C. (1984). An evaluation of random testing. *IEEE Transactions on Software Engineering*, SE-10(4):438–444.

Gupt, K. K., Kshirsagar, M., Sullivan, J. P., and Ryan, C. (2021a). Automatic test case generation for prime field elliptic curve cryptographic circuits. In *2021 IEEE 17th International Colloquium on Signal Processing Its Applications (CSPA)*, pages 121–126.

Gupt, K. K., Kshirsagar, M., Sullivan, J. P., and Ryan, C. (2021b). Automatic test case generation for vulnerability analysis of galois field arithmetic circuits. In *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, pages 32–37.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.

McCluskey, E. J. (1987). Exhaustive and pseudo-exhaustive testing. Technical report, Stanford Univ CA Center for Reliable Computing.

Muselli, M. and Liberati, D. (2000). Training digital circuits with hamming clustering. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(4):513–527.

Narciso, E. N., Delamaro, M. E., and De Lourdes Dos Santos Nunes, F. (2014). Test case selection: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, 24(4):653–676.

Orso, A. and Rothermel, G. (2014). Software Testing: A Research Travelogue (2000–2014). In *Future of Software Engineering Proceedings*, FOSE 2014, pages 117–132, New York, NY, USA. Association for Computing Machinery.

Pradhan, M. and Bhattacharya, B. B. (2021). A survey of digital circuit testing in the light of machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(1):e1360.

Rosenbauer, L., Stein, A., and Hähner, J. (2021). An Artificial Immune System for Black Box Test Case Selection. In *EvoCop: 21st European Conference on Evolutionary Computation in Combinatorial Optimisation as part of evostar 2021, April 2021, Seville, Spain*.

Rothermel, G., Harrold, M. J., Ostrin, J., and Hong, C. (1998). Empirical study of the effects of minimization on the fault detection capabilities of test suites. In *Conference on Software Maintenance*, pages 34–43. IEEE.

Sapna, P. G. and Mohanty, H. (2010). Clustering test cases to achieve effective test selection. In *Proceedings of the 1st Amrita ACM-W Celebration of Women in Computing in India, A2CWiC'10*, pages 1–8, New York, New York, USA. ACM Press.

Tamasauskas, D., Sakalauskas, V., and Kriksciuniene, D. (2012). Evaluation framework of hierarchical clustering methods for binary data. In *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 421–426. IEEE.

Thamarai, S., Kuppusamy, K., and Meyyappan, T. (2010). Heuristic approach to optimize the number of test cases for simple circuits. *arXiv preprint arXiv:1009.6186*.

White, L. J. and Cohen, E. I. (1980). A Domain Strategy for Computer Program Testing. *IEEE Transactions on Software Engineering*, SE-6(3):247–257.

Wong, W. E., Horgan, J. R., London, S., and Mathur, A. P. (1995). Effect of test set minimization on fault detection effectiveness. In *Proceedings - International Conference on Software Engineering*, pages 41–50. IEEE.