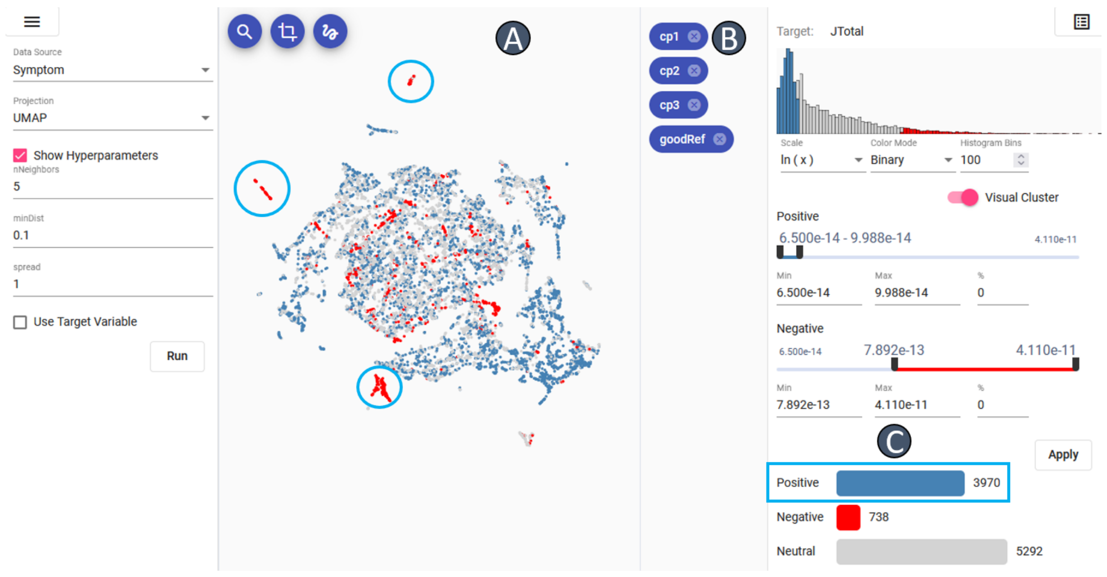
Interactive Analysis of Post-Silicon Validation Data

Andres Lalama, Johannes Knittel, Steffen Koch, Daniel Weiskopf, Thomas Ertl[*]
University of Stuttgart
Sarah Rottacker, Raphaël Latty, Jochen Rivoir[†]
Advantest Europe GmbH



Data Source: Symptom
Projection: UMAP
☑ Show Hyperparameters
nNeighbors: 5
minDist: 0.1
spread: 1
☐ Use Target Variable

Target: JTotal
Scale: ln (x) | Color Mode: Binary | Histogram Bins: 100

Positive: 6.500e-14 - 9.988e-14 (4.110e-11)
Min: 6.500e-14 | Max: 9.988e-14 | %: 0

Negative: 6.500e-14 | 7.892e-13 | 4.110e-11
Min: 7.892e-13 | Max: 4.110e-11 | %: 0

Positive	3970
Negative	738
Neutral	5292

Figure 1: Overview visualization of a post-silicon validation dataset. Based on a chosen target error attribute (here: JTotal), validation engineers can define thresholds on the right side of the interface to divide test cases into passed, neutral, or failed (C). Our system then projects a subspace of the data composed of all symptom attributes into a two-dimensional scatter plot in which each dot is colored according to its test outcome as defined by the specified thresholds. Users can select points of interests in the projection, for instance, clusters of mainly failing test cases (A), which could indicate that the source of the problem differs between these groups. In subsequent interactive analysis steps, engineers can refine these selections (B) to delineate problems and they can investigate the correlation of input attributes with such extracted problem cases to obtain an understanding of its causes.

Abstract

Analyzing post-silicon validation datasets to spot bugs and obtain hints for design improvements challenges validation engineers with complex and sometimes subtle problems. Through experience, engineers gain accuracy and speed in recognizing the cause of a problem, but analyzing the data remains a tedious and time-consuming process involving substantial manual effort. Sometimes, validation engineers just have a gut feeling regarding the cause, and in many cases they can find themselves at a loss in understanding the cause of a problem. This work presents a new approach that visually supports validation engineers in debugging chips during the post-silicon validation phase. It was developed as part of a collaboration between visualization researchers and post-silicon validation domain experts, aiming at tackling complex problems that present subtle differences between test case outcomes.

Index Terms: Visual analytics—Debugging—Multivariate Dataset—;

[*]e-mail: andres.lalama@vis.uni-stuttgart.de
[†]e-mail: firstname.lastname@advantest.com

1 Introduction

Post-silicon validation deals with testing and debugging integrated circuits (IC) chips on actual hardware. After a specific chip has been designed, it has to be tested in hardware to make sure that it works properly without critical bugs that may have been missed in earlier development and testing stages, to obtain hints about possible design improvements, and to specify a set of optimal parameters and operating conditions. To achieve this, the chips undergo several automatic test runs with different parameters and operating conditions. Engineers then analyze the produced validation dataset with regard to the testing objectives, which is typically a difficult, tedious, and time-consuming process that requires in-depth domain knowledge.

In our joint project, visualization researchers and domain experts from a leading provider of automatic IC test equipment collaborate to develop a visual analytics approach that aims to specifically support such post-silicon validation tasks. In this work, we first outline a typical validation workflow and describe the numerous challenges and requirements that arise when analyzing the obtained datasets. We then present an early version of our visualization approach (Fig. 1) that provides a straightforward yet flexible workflow specifically adapted to the complex testing objectives and that aims to support validation engineers during post-silicon validation (PSV) and ultimately help to lower the barrier of entry for PSV debugging experts. Our use case shows that the developed approach is effective for extracting and characterizing different problem cases in multidimensional validation datasets and for providing initial hints about underlying causes.

2 Related Work

While there is a considerable body of work related to the design and execution of post-silicon validation tests for different chips, holistic (published) approaches for the visual analysis of resulting validation datasets for debugging and error localization purposes are lacking to date. Many analysis steps to retrace and understand problems still require significant manual human effort [3]. To better support validation engineers, some approaches have thus proposed the use of clustering algorithms such as k-means for detecting anomalies to better understand whether failing tests constitute a bug [3, 11].

Visual analytics approaches that generally aim to support analyses for large multidimensional datasets in other domains are most relevant to our approach. In addition to simpler visualization techniques such as scatter plot matrices [1] and parallel coordinates [8], more advanced, interactive approaches have been developed to gain insights into multivariate datasets.

Several approaches [2, 5, 6] first reduce the data to two-dimensional scatter plots (e.g., with UMAP [13]) to provide an overview and enable analysts to explore how clusters of seemingly similar items differ to each other. Visual Neural Decomposition [9] applies a neural network to extract and visualize relationships separately for similar cases. Another strategy is to first extract subspaces of interest, which can then be visualized in more compact ways [7, 10, 18]. For finding relevant attributes or sets of attributes, we can rank attributes and pairs of attributes according to statistical correlation factors [4, 12, 15] or classification metrics such as separability [17]. Such rankings also motivate the use of iterative workflows in which analysts increase the complexity of relationships step-by-step, guided by suggestions based on previously performed selections [14, 19, 20]. The correlation map by Zhang et al. [21] visualizes pairwise correlations between attributes in a graph.

However, these approaches typically either aim to provide general methods for exploring multivariate datasets or focus on domains other than ours. In contrast to this, our approach is specifically tailored to the challenges and requirements of post-silicon validation analysis tasks.

3 Requirements

One of the main goals in post-silicon validation is to better understand in which situations the first prototypes of integrated circuit (IC) chips do not behave as expected or produce erroneous outputs. We, an interdisciplinary group composed of visualization researchers and domain experts from a leading provider of automatic IC test equipment, discussed the typical setting and workflow of post-silicon validation in a series of meetings, while iterating over prototypes based on real world data. We based this process on the design study methodology [16]. In particular, we were interested in the challenges validation engineers currently face when analyzing the collected data for spotting bugs and obtaining hints for design improvements.

3.1 Data Collection

The chips are tested in an automated way using different parameters and operating conditions, often in iterative batches. The resulting multivariate validation dataset captures several measurements in addition to the respective parameters and operating conditions for each test run.

The project's domain experts divide the attributes of the collected data into three categories. *Input attributes (inputs)* correspond to the different parameters and operating conditions under which the chip was tested. *Target attributes (targets)* indicate whether the chip performs in-spec or out-of-spec and are thus the most important quantities for determining whether the chip failed a specific test run. However, instead of binary pass/fail test results, they are often floating point numbers that, for instance, express the extent of deviations from the expected output, i.e., engineers want to find out in which situations such a target attribute has high values. *Symptom attributes (symptoms)* may contribute valuable information about the state of the device under test to gain an understanding of what is going on. They could indicate errors as well as unexpected or undesired behavior that may not necessarily lead to a failed test run in a strictly technical sense (out-of-spec behavior).

3.2 Analysis Goals

According to the domain experts in our project team, validation engineers analyze the data to investigate in which situations out-of-spec behavior occurs and to get clues about possible design improvements. Since there could be very well several problems that cause undesired behavior, the engineers typically first try to identify groups of problem cases that probably share a similar root cause. While validation engineers can quickly identify out-of-spec test cases based on the collected target attributes, they are interested in how they can characterize and delineate these issues based on symptom attributes and, for each identified group of problems, understand the relationships between input attributes and the respective problem.

One strategy that worked well in the past is to first identify all problematic cases based on one or several target attributes, for instance, test cases that exhibit an error rate above a certain threshold. In a next step, the engineers would look at the symptoms to discern different groups of problematic cases. For each such group, analysts would then investigate which input attributes and value ranges correlate with the respective undesired outcome to get a better understanding of what could cause the problem or which operating conditions should be avoided.

Hence, in contrast to more generic analyses of multivariate datasets, the goal here is two-fold. The first goal is to analyze a subspace of the dataset for extracting and characterizing individual problems. Based on and using this obtained knowledge, the subsequent goal is then to investigate the relationships between input attributes and the extracted problems.

3.3 Challenges

Validation engineers have to deal with numerous challenges when analyzing the data. They need to accomplish two major tasks. They must disentangle and characterize different problem cases solely based on the collected data, mainly involving symptom and target attributes, and they must gain a better understanding of the relationships between input attributes and undesired outcomes, involving all attributes.

Recognizing truly multidimensional and non-linear relationships in datasets is a difficult endeavor, particularly because of the growing sparsity of data points in high-dimensional spaces. Even if we can extract a limited set of interesting attributes for a specific problem case, it may still be challenging to visualize their multidimensional relationship with the respective target attribute(s) so that analysts can better understand possible problem causes.

We must also take into account that the occurrence of errors can be sporadic and that, depending on the test objective, there may well be a spectrum of results that sometimes cannot be clearly categorized as *passed* or *failed*. It also happens that errors only occur with an unfavorable combination of different input parameters and value ranges. Furthermore, problems that trace back to different causes can overlap. The applied methods must also deal with different attribute types and encodings, such as binary inputs, categories, floating point values, and integer values.

As of now, the process of analyzing validation datasets is a tedious, labor-intensive process that requires skilled engineers. They often start with testing initial, 'gut-driven' hypotheses and continue the analysis process in an iterative manner, employing statistical tests and straightforward visualizations such as scatter plot matrices.

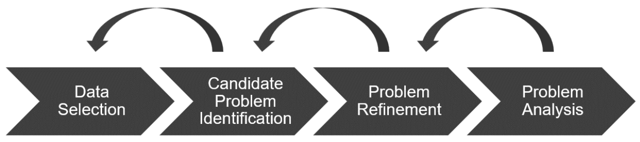
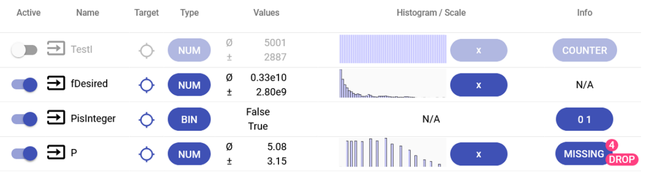


Figure 2: Interactive workflow with its phases, the main direction of analysis, and possible feedback loops.



Active	Name	Target	Type	Values	Histogram / Scale		Info
	TestI		NUM	Ø 5001 ± 2887		x	COUNTER
	fDesired		NUM	Ø 0.33e10 ± 2.80e9		x	N/A
	PisInteger		BIN	False True	N/A		0 1
	P		NUM	Ø 5.08 ± 3.15		x	MISSING 4 DROP

Figure 3: The tabular data selection interface, where users can inspect the preprocessed data and choose the attributes to be analyzed.

4 METHODOLOGY

As mentioned in the previous section, the analysis and debugging of multivariate data generated by post-silicon validation (PSV) tests is complex and ultimately relies on skilled validation engineers tediously analyzing datasets using straightforward visualizations and in-house developed tools. Thus, in our joint project, we are developing an interactive, integrated visualization approach for the analysis of PSV test data inspired by the design study methodology [16]. Our approach is intended to provide a basic, yet flexible workflow adapted to the complex objectives that supports validation engineers in finding, characterizing, and understanding problematic cases and failed test runs and helps lower the barrier of entry for PSV debugging experts. The focus of this paper is to present an early version of this first integrated visualization approach for PSV and its effectiveness for analyzing initial problems. The extent to which this reduces the learning curve and workload for experts has yet to be evaluated.

4.1 Workflow

Based on many conversations and discussions between PSV and visualization experts, we developed a workflow comprising several steps to tackle the aforementioned challenges in the analysis of PSV test data. Each step focuses on a specific set of objectives:

1. **Data Selection:** The selection of input and symptom attributes to be analyzed. Depending on the task, different targets or symptoms could be relevant, and other attributes in the dataset might not be of use during the analysis (e.g., test id).

2. **Candidate Identification:** An overview and first rough exploration of the test cases after the analyst has chosen one target attribute as main error indicator. This step should already provide first insights into possibly different groups of error cases.

3. **Problem Refinement:** The selection, comparison, and delimitation of possible groups of problems (subproblems), mainly involving symptom and target attributes, since the dataset may contain several not necessarily related issues that analysts want to get a grasp on.

4. **Problem Analysis:** An analysis of the causes of an identified (sub-)problem in terms of input attributes and value ranges, which is important for fixing bugs, designing future test runs, and determining suitable operating conditions.

As shown in Fig. 2, the phases already imply a general direction of the workflow. However, the individual objectives often cannot be achieved in one step right away, but can only be worked out successively through iterative analyses. Such iterations can be carried out within individual phases or encompass several phases. An example of this is the selection of a possible subproblem, which is initially performed in the second phase, but is extended or restricted accordingly after comparison with other problems and cases that have passed the test (third phase). The user interface of the integrated approach is also aligned to the different phases and offers separate interactive visualizations tailored to the respective objectives.

The prototype was created as a web-based application to simplify testing, provide access to intermediate stages of the development, and make sure that the developed approach scales well to many clients based on modern web technologies. For some computations, such as the UMAP projection, we still rely on server-side computing but aim at integrating those as well on the client-side in the long run. In the following, the visualization interfaces and the respective interaction possibilities are presented. An example of an analysis, and thus the interaction of the individual interfaces, is presented in Sect. 5.

4.2 Data Preprocessing and Selection

As indicated in Sect. 3.1, there are three categories of attributes in the collected validation dataset: inputs, symptoms, and targets. In addition to the actual dataset, our system can also load a separate file that indicates the category of each attribute. Even though this is optional, it greatly supports the workflow, since the different categories differ in relevance for each of the analysis steps that we outlined in the previous section. Users may also change the category associations (e.g., for a specific analysis task, the assignment of symptom and target attributes might differ).

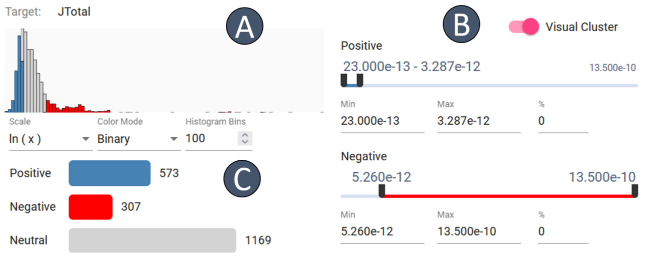
After loading both files, the system preprocesses the data. This includes an analysis of the attributes' values and potential problems. Attributes that show no variance in their values, are empty, or are deemed as a counter will be automatically deactivated. Fig. 3 shows the interface after the preprocessing step.

Since the data is organized into test cases, we show them in a table-based interface in which each row depicts one test result with respective input parameters, symptoms, and target values. The interface indicates whether attributes have some missing values. If this is the case, it automatically suggests dropping the rows containing the missing values, but the user can opt to fill those values with zero or with the attribute's mean value. After dealing with these detected issues, users select those attributes of the data they would like to analyze in subsequent steps, thereby filtering the test data to a suitable subset.

To visualize the attribute's value distribution, we show a compact histogram view that fits well to the tabular view due to its low height. It is possible to switch between a linear and a logarithmic scale. The latter improves the handling of skewed value distributions, both visually and computationally. The system additionally displays the attribute's mean value and variance.

4.3 Candidate Problem Identification

The second phase aims at providing an overview of the data while indicating latent similarity of the test cases at the same time. Showing an overview of passed and failed tests helps identify and select subsets of interesting test cases to be included in subsequent analysis steps. The validation engineer first sets one attribute as the chosen target that best fits their analysis goal (out of the set of targets) and specifies two thresholds of that target with a slider that maps the (continuous) value into one of three possible test results: passed ('good'), in-between ('neutral'), and failed ('bad'), as depicted in Fig. 4 B. The user interface typically marks passed cases in blue, in-between cases in gray, and failed cases in red. We visualize the


Target: JTotal

Scale: ln (x)
Color Mode: Binary
Histogram Bins: 100

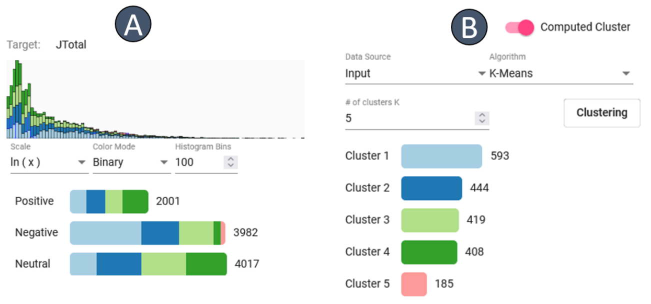
Positive 573
Negative 307
Neutral 1169

Visual Cluster

Positive: 23.000e-13 - 3.287e-12 (13.500e-10)
Min: 23.000e-13 Max: 3.287e-12 %: 0

Negative: 5.260e-12 – 13.500e-10
Min: 5.260e-12 Max: 13.500e-10 %: 0

Figure 4: Users can define ranges of positive, negative, and undefined test outcomes with sliders (B). These are depicted in a histogram view (A) and as a bar chart (C) to understand the numbers of test cases that fall into these categories. The latter can also be used for selecting categories in a convenient way.


Target: JTotal

Scale: ln (x)
Color Mode: Binary
Histogram Bins: 100

Positive 2001
Negative 3982
Neutral 4017

Computed Cluster

Data Source: Input
Algorithm: K-Means
# of clusters K: 5

Cluster 1 593
Cluster 2 444
Cluster 3 419
Cluster 4 408
Cluster 5 185

Figure 5: (A) Computed clusters mapped to user-defined ranges. (B) Bar chart of computed clusters with their associated color.

distribution of the chosen target attribute with a histogram (Fig. 4 A) in which we also indicate the respective thresholds using accordingly stacked bars to help analysts set suitable thresholds. Based on this classification, we provide several views for generating an overview of the dataset and enabling analysts to get first insights into possibly different groups of error cases, which we describe in the following sections.

4.3.1 Data Projection

We project a subspace of the dataset involving only symptom attributes into a two-dimensional scatter plot using UMAP [13] to provide analysts with a visual overview of the test cases (Fig. 1 A). Each dot represents one test case and is colored according to the test result (defined by the specified thresholds). As outlined in Sect. 4.1, validation engineers typically discern different groups of problems by looking at the symptoms. Hence, the idea is that similar test cases in terms of their symptoms should also be projected nearby, possibly forming visual clusters. Since the symptoms are used for the disambiguation of different problem cases in traditional workflows, projecting test cases based on them provides a good chance of indicating subproblems visually.

Based on this visualization, analysts can then interactively select groups of points of interest to investigate them further and save them for subsequent analysis steps. For instance, a visual cluster that mainly comprises red dots could indicate one group of test cases that exhibit a similar issue. Even though such a projection inevitably leads to information loss and may not always preserve data distances, it can nevertheless offer hints about possible groups of subproblems and serve as a starting point for further analysis.

4.3.2 Computed Clusters

Since the projection might not always succeed in separating groups of problematic cases as visually distinguishable clusters, the system also supports the computation of clusters in the multidimensional space by applying k-means clustering either using only symptoms or only inputs (Fig. 5). After the clusters have been computed, we can switch the coloring of the dots to the computed clusters, i.e., the color of a dot then indicates its associated computed cluster rather than its test result.

Analysts can then assess whether there is an overlap with the visually formed clusters, but they may also get a first hint whether a group of similar test cases in the input space corresponds to one of the interesting clusters. One important caveat is, though, that analysts have to specify the number of clusters, so they may need to run the clustering several times with different hyperparameters to obtain a useful clustering.

4.3.3 Clusters Selection

Our interface helps analysts select interesting clusters and save them for further analysis. We provide several ways to select clusters (also called *selections* in this work). Analysts can select them in the scatter plot using a circular or rectangular brush, they can choose all cases with one specific outcome (e.g., passed), or they can click on one of the computed clusters to select all associated test cases.

These clusters then typically represent either potential subproblems ('bad clusters') or clearly passing test cases ('good clusters'). The main advantage of separating the dataset into several such clusters is that it simplifies subsequent analysis steps in which analysts compare different 'bad clusters' to each other or contrast them to one or several 'good cluster(s)'. Operating on meaningful subsets of the data enhances the utility of our visualization views (e.g., less clutter and possibly easier-to-spot relationships in parallel coordinates plots) and it also greatly supports validation engineers in characterizing and understanding individual problems compared to analyzing entire datasets as a whole. Our system offers several interaction mechanisms so that users can refine these clusters iteratively, as presented in the next section.

4.4 Problem Refinement

In the next analysis step, the system allows exploring previously created selections so that analysts can delineate groups of problems and obtain a better understanding of the problem in terms of symptoms. For instance, a selected visual cluster may still contain two problems that should be investigated separately. We may also want to understand whether other test cases placed elsewhere in the projection still exhibit similar symptoms. Analogously to our workflow, users can click on the *Refine* tab, which provides several views to support the refinement and debugging of selections and problems.

4.4.1 Re-Projection

Each row in the *Refine* view corresponds to one selection, as depicted in Fig. 6. We highlight the associated points with a black outline in the compact projection overview to the very left. The system also computes a local UMAP projection using only the data items in the respective selection, which is then visualized next to the main, global projection. With the two projections side-by-side, analysts can assess whether there are any larger differences between the projections, for instance, the occurrence of additional visual clusters in the local projection, which are not visible in the global projection (e.g., due to competing projection stresses). Users can interact with this local projection to modify selections as well as adding new ones.

4.4.2 Attribute Scoring and Difference Assessment

Next to the two projections, we show a scoring of the attributes with small colored dots from white to purple based on how different the histogram of the respective attribute is to our reference selection (which could be another selection or the entire dataset). The scoring

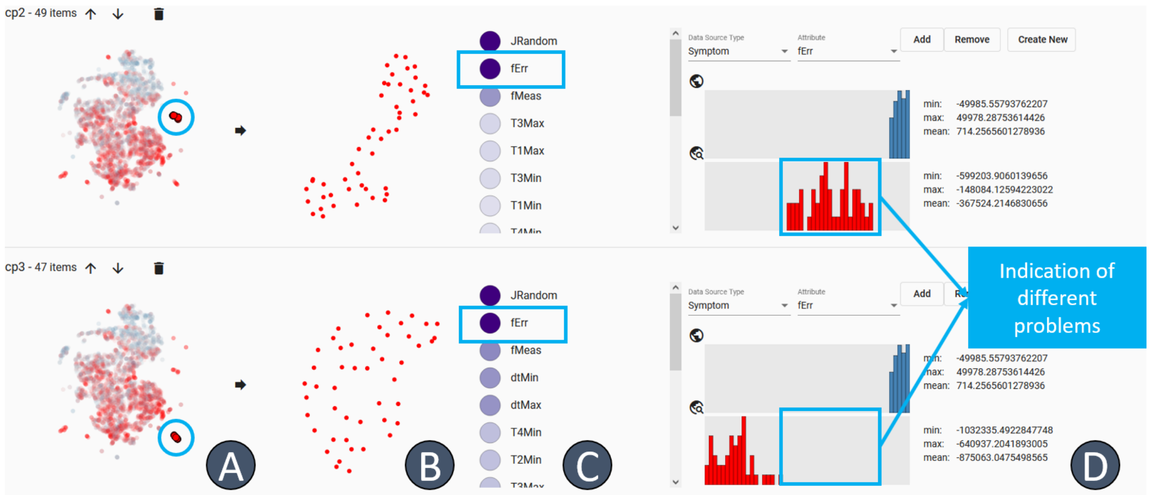


Figure 6: The *Refine* view allows investigating and modifying clusters of interest that are visualized in a row-based layout for easier comparison. This example shows two clusters mainly comprising failed test cases (in red). For each row, The scatter plot (A) highlights the respective cluster on the initial global projection based on symptoms (first phase). The second scatter plot (B) visualizes the reprojection of the respective data selection to help understand potential intra-cluster patterns. Column (C) shows a ranking of interesting attributes based on the Kolmogorov–Smirnov test. The histogram views (D) show how the distribution of values of a specified attribute differs between a reference cluster (top histogram in each row) and the respective clusters (bottom histogram), which helps delineate different subproblems.

is based on the nonparametric Kolmogorov–Smirnov test, which evaluates the distance between two distributions by computing the maximum distance between two (discretized) cumulative density functions at any bin. High values in purple indicate pronounced differences between the two distributions, guiding users to potentially interesting symptom attributes that could help differentiate different selections and, thus, subproblems. When we know which symptom attributes differ the most between two selections, odds are high that they can help to characterize a given problem.

To the right, the user interface displays two stacked histograms that both correspond to one selected attribute but on different subsets of the dataset, which allows users to explore in greater detail how certain symptom attributes differ between selections. The first histogram at the top visualizes the distribution of values in a user-defined reference selection; the one underneath is based on all data items of the respective selection in the current row. To simplify their comparison, we align them vertically in our row-based layout of potential subproblems. The option to reorder rows interactively lets users compare problems more easily. The minimum, maximum, and mean of the values represented in the histogram are displayed on the right side to further support the difference assessment of the attributes. The best case scenario is to find a distribution in one candidate problem that rarely overlaps with another candidate problem, since the respective attribute and identified value ranges could then be used to delineate this problem from other problems in the dataset.

Once analysts have identified an attribute of interest that helps delineate or understand a given set of problematic cases, they can bookmark it for subsequent analysis steps. This list is then stored together with the respective selection and serves as a compact problem characterization in terms of involved symptom attributes.

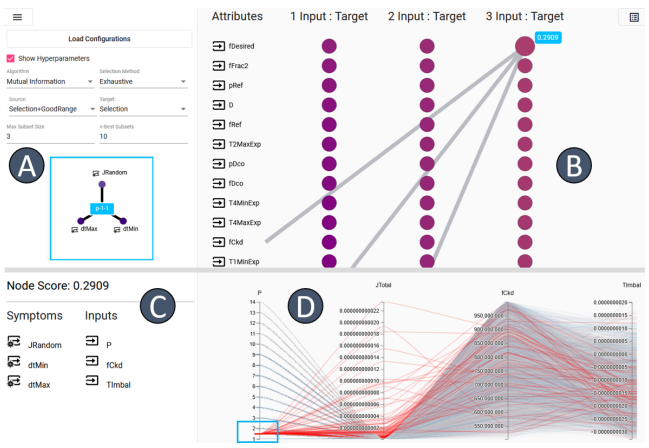
4.5 Problem Analysis

The final step of the workflow supports validation engineers in developing an understanding of the cause of identified errors by visualizing which input attribute value combinations relate to a problem, i.e., for a given subproblem we identified and delineated in phases two and three, we want to find out which inputs and input ranges correlate with the occurrence of that subproblem. The rationale behind this idea is that if one attribute or a combination of several attributes is correlated to failed test cases, it is likely that these inputs are causing the problem. Should several input attributes in combination cause a problem, their joint correlation with the target attribute should be higher than the individual correlations of single inputs and the target.

To achieve this, we first add for each user-identified subproblem a binary attribute to the dataset that depicts if the respective item is part of the identified problem and filter the dataset to all items in the selection plus all items of a ‘good’ cluster mainly containing passing test cases. We then compute and visualize the mutual information between (sets of) inputs and the corresponding binary attribute to get an idea which input attributes correlate with our subproblem the most. The advantage of computing the mutual information is that we can also detect nonlinear correlations, since it is an information-theoretic test of dependence.

Fig. 7 shows an example. The left side lists all previously created selections of subproblems and the respective symptom attributes that have been bookmarked (Fig. 7 A). If the user picks one of these selected problems, the system visualizes a ranking of possibly correlated attributes and sets of attributes in a grid-like layout (Fig. 7 B). The first column of dots indicates the mutual information scores (from purple to orange) between the respective input attribute to the left and our target binary attribute. The second column of dots visualizes the mutual information between attribute pairs and our target, the third column between sets of three attributes and our target, and so on. The parallel coordinates plot (Fig. 7 D) visualizes the data items in the currently selected problem. Users can hover over a dot, which will update the axes of the parallel coordinates plot at the bottom with the respective input attributes. The mutual information grid helps users to select interesting subspaces and to quickly assess how the selected input attributes influence the subproblem of interest, even if the actual number of attributes in the dataset is high.

An exhaustive computation of the mutual information of all possible attribute combinations and the target would be prohibitively expensive. As of now, we compute the mutual information up until sets of three input attributes, but we plan to implement an iterative process in the future to also support subspaces of four and more input attributes by pruning low-scoring sets after each step and only extending the remaining sets for higher-order correlations.


Load Configurations

Show Hyperparameters

Algorithm: Mutual Information

Selection Method: Exhaustive

Source: Selection+GoodRange

Target: Selection

Max Subset Size: 3

n-Best Subsets: 10

Node Score: 0.2909

Symptoms	Inputs
JRandom	P
dtMin	fCkd
dtMax	TImbal

Figure 7: *Problem Analysis* view to investigate which input attributes and ranges correlate to one of the identified subproblems. After selecting one of the previously created selections of subproblems (A), the system visualizes a ranking of possibly correlated attributes and sets of attributes in a grid-like layout (B). The first column of dots indicates the mutual information from purple (low) to orange (high) between the input attribute to the left and our target, which is a binary attribute indicating the membership to the respective problem selection. The second column of dots visualizes the mutual information between attribute pairs and our target, and so on. Users can hover a dot, which will update the result view (C) showing the score and attributes being analyzed, as well as the axes of the parallel coordinates plot at the bottom accordingly (D), visualizing the data items in the selection.

5 Use Case

As stated in Sect. 3, validation engineers want to understand in which situations the first chip prototypes fail to behave as expected and which input attributes and value ranges lead to such erroneous behavior. We describe the following use case based on a proprietary artificial benchmark dataset, which is based on realistic scenarios and was created by domain experts. We assume here that we have a clocking chip that should not exhibit too high jitter, that is, the timing of the output signal should not deviate too much from the ideal values. The dataset consists of 10,000 data items and 64 attributes (59 after preprocessing). The chosen target is the total jitter *JTotal*.

Based on visual clusters of red points in the projection (see Fig. 1), the validation engineer creates three selections of candidate problems *cp1*,*cp2*, and *cp3*, as well as a fourth selection comprising all data items that passed the test (*goodRef*) for comparison. In the *Refine* tab, the engineer sees that the first selection differs greatly from the other two problem selections, but the second and third selection show similar characteristics in the respective top four symptom attributes. This is an indicator that, even though the clusters were further away in the projection, they still might be caused by the same problem. Now, the validation engineer takes a closer look at the histograms and sees how well the distributions match. The candidate problems *cp3* and *cp2* show a different distribution of the symptom attribute *fErr* as depicted in Fig. 6 D.

The validation engineer now bookmarks relevant symptom attributes for both selections of problem cases. They bookmark *JRandom*, the random jitter, and *dtMin* and *dtMax*, depicting the limits of control deviations (Fig. 7 A).

In the final stage, the mutual information for the identified subproblems is computed. Fig. 7 B depicts the ranking of possible attributes and sets of attributes in descending order (top to bottom) of *cp2*. Hovering the topmost node having the highest rank for three input attributes, the engineer realizes that *P*, *fCkd* and *TImbal* seem to correlate with problem cases in *cp2*. This updates both the small overview (Fig. 7 C) and the PCP (Fig. 7 D) that shows the correlation between the input attributes and the chosen target attribute. One interesting finding is that, for this subproblem, problematic test cases only occur when attribute *P* has a value of 1.5. This is consistent with the known ground truth of this benchmark dataset.

6 Discussion

A strength of our proposed approach and the associated workflow is that they were developed iteratively in close collaboration with PSV experts, taking into account broad and longstanding experiences gained on previously used, non-integrated procedures. In contrast to previous work on the visual analysis of multivariate data, our system is specifically tailored to the needs and requirements of validation engineers that analyze post-silicon validation data. The choice of visualization components and interaction methods are based on intensive discussions after reviewing alternatives. Another strength of this work is that we had access to a real test data set for the development. It certainly helped in making the approach more robust and informed many design decisions, especially regarding the close integration of attribute types (symptoms versus targets) and regarding data preprocessing.

At the same time, however, it is clear that focusing on a specific dataset also has drawbacks. There is a risk of overfitting the approach to the errors it contains and their causes. Unfortunately, chip test data sets are rarely (if at all) publicly available, which is not least due to the fact that this data is usually strictly confidential. Another challenge is that not all possible or even typical problems will be present in one test data set. To counteract this problem, we used a tool for generating artificial test data sets. This tool allows us to generate fault situations with different levels of complexity, and at the same time the ground truth is available for evaluating the visual analytics approach.

Our first experiments show that it is possible with the presented approach to detect and characterize errors by means of interactive analyses in addition to the meaningful analysis of a real data set. However, these analyses have so far only been applied by ourselves to a manageable number of error situations. A more comprehensive, independent, and long-term evaluation of the approach with external users is still pending. Nevertheless, based on our experience and carried out analyses, we are convinced that our workflow-driven approach and design can be effectively applied for debugging PSV data. We consider the definition of the iterative workflow and the choices made regarding the interactive visual components an important contribution that can be extended to cover additional problem cases in the future.

7 Conclusion and Future Work

With this work, we contribute a workflow that visually aids validation engineers in debugging complex problems during post-silicon validation. Based on this workflow, we developed a first prototype implementation that enabled us to analyze a real-world validation dataset and several benchmark datasets successfully, showing the suitability of our proposed approach. In the future, we want to get additional feedback from validation engineers who are utilizing our system as part of their daily analyses. This will most probably lead to an improvement of the approach's usability and further revisions of design decisions. In addition, we plan to incorporate additional automatic analysis steps with matching visualization and interaction techniques that can be integrated into a holistic workflow for analyzing post-silicon validation data.

Acknowledgments

This research was supported by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart.

References

[1] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. Scatterplot Matrix Techniques for Large N. *Journal of the American Statistical Association*, 82(398):424, 1987. doi: 10.2307/2289444

[2] A. Chatzimparmpas, R. M. Martins, and A. Kerren. T-viSNE: Interactive Assessment and Interpretation of t-SNE Projections. *IEEE Transactions on Visualization and Computer Graphics*, 26(8):2696–2714, 2020. doi: 10.1109/TVCG.2020.2986996

[3] A. DeOrio, Q. Li, M. Burgess, and V. Bertacco. Machine learning-based anomaly detection for post-silicon bug diagnosis. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 491–496, 2013. doi: 10.7873/DATE.2013.112

[4] C. Eichner, H. Schumann, and C. Tominski. Making Parameter Dependencies of Time-Series Segmentation Visually Understandable. *Computer Graphics Forum*, 39(1), 2020. doi: 10.1111/cgf.13894

[5] T. Fujiwara, O. H. Kwon, and K. L. Ma. Supporting Analysis of Dimensionality Reduction Results with Contrastive Learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):45–55, 2020. doi: 10.1109/TVCG.2019.2934251

[6] M. Gleicher. Explainers: Expert explorations with crafted projections. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2042–2051, 2013. doi: 10.1109/TVCG.2013.157

[7] M. Hund, D. Böhm, W. Sturm, M. Sedlmair, T. Schreck, T. Ullrich, D. A. Keim, L. Majnaric, and A. Holzinger. Visual analytics for concept exploration in subspaces of patient groups: Making sense of complex datasets with the Doctor-in-the-loop. *Brain Informatics*, 3(4):233–247, 2016. doi: 10.1007/s40708-016-0043-5

[8] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(4):69–91, 1985. doi: 10.1007/BF01898350

[9] J. Knittel, A. Lalama, S. Koch, and T. Ertl. Visual Neural Decomposition to Explain Multivariate Data Sets. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1374–1384, 2021. doi: 10.1109/TVCG.2020.3030420

[10] J. Krause, A. Dasgupta, J. D. Fekete, and E. Bertini. SeekAView: An intelligent dimensionality reduction strategy for navigating high-dimensional data spaces. In *Proceedings of the 2016 IEEE Symposium on Large Data Analysis and Visualization, LDAV 2016*, pp. 11–19, 2017. doi: 10.1109/LDAV.2016.7874305

[11] B. Kumar, K. Basu, and V. Singh. A Technique for Electrical Error Localization with Learning Methods During Post-silicon Debugging. In *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*, pp. 1–8, 2018. doi: 10.1109/IGCC.2018.8752141

[12] A. Malik, R. Maciejewski, N. Elmqvist, Y. Jang, D. S. Ebert, and W. Huang. A correlative analysis process in a visual analytics environment. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology, VAST 2012*, pp. 33–42, 2012. doi: 10.1109/VAST.2012.6400491

[13] L. McInnes, J. Healy, N. Saul, and L. Großberger. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861

[14] T. Mühlbacher and H. Piringer. A partition-based framework for building and validating regression models. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1962–1971, 2013. doi: 10.1109/TVCG.2013.125

[15] H. Piringer, W. Berger, and H. Hauser. Quantifying and comparing features in high-dimensional datasets. In *Proceedings of the International Conference on Information Visualisation*, pp. 240–245, 2008. doi: 10.1109/IV.2008.17

[16] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012. doi: 10.1109/TVCG.2012.213

[17] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnor, and D. Keim. Combining automated analysis and visualization techniques for effective exploration of high-dimensional data. In *Proceedings of the 2009 IEEE Symposium on Visual Analytics Science and Technology, VAST 2009*, pp. 59–66, 2009. doi: 10.1109/VAST.2009.5332628

[18] A. Tatu, F. Maaß, I. Färber, E. Bertini, T. Schreck, T. Seidl, and D. Keim. Subspace search and visualization to make sense of alternative clusterings in high-dimensional data. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology, VAST 2012*, pp. 63–72, 2012. doi: 10.1109/VAST.2012.6400488

[19] C. Turkay, A. Lundervold, A. J. Lundervold, and H. Hauser. Representative factor generation for the interactive visual analysis of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2621–2630, 2012. doi: 10.1109/TVCG.2012.256

[20] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. MacKinlay, B. Howe, and J. Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 2648–2659, 2017. doi: 10.1145/3025453.3025768

[21] Z. Zhang, K. T. McDonnell, E. Zadok, and K. Mueller. Visual correlation analysis of numerical and categorical data on the correlation map. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):289–303, 2015. doi: 10.1109/TVCG.2014.2350494