Selection of Tests for Outlier Detection

Harm C.M. Bossers, Johann L. Hurink, Gerard J.M. Smit
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands
{h.c.m.bossers, j.l.hurink, g.j.m.smit}@utwente.nl

***Abstract*—Integrated circuits are tested thoroughly in order to meet the high demands on quality. As an additional step, outlier detection is used to detect potential unreliable chips such that quality can be improved further. However, it is often unclear to which tests outlier detection should be applied and how the parameters must be set, such that outliers are detected and yield loss remains limited. In this paper we introduce a mathematical framework, that given a set of target devices, can select tests for outlier detection and set the parameters for each outlier detection method. We provide results on real world data and analyze the resulting yield loss and missed targets.**

I. Introduction

In the semiconductor test industry, outlier detection methods are used more and more in order to achieve (close to) zero dppm (defective parts per million). An outlier is defined as a measurement which differs significantly from an expected pattern of behavior, even if the measurement is within its specification limits [1]. The rationale for outlier detection is that deviating behavior is probably caused by some latent defect and entails reliability risks. This is also supported by empirical evidence [2]. Therefore, a lot of outlier detection methods have been developed in recent years. However, applying outlier detection immediately raises some problems:

- Which outlier detection method should be used?
- On which tests should outlier detection be applied?
- How to set the parameters of the outlier detection method?
- What is the impact on yield loss?

Hereby, the last question is of crucial importance. In the past decade, a wide range of different outlier detection methods have been developed, all with their strengths and weaknesses. So, choosing one or a few outlier detection methods is not a trivial task. And after the choice of method(s) is made, the second and third problem pop up: on which tests to apply outlier detection and how to set the parameters of the method. If just blindly all tests are selected with strict tolerance levels, probably dppm will go down to zero, but at the cost of a very high yield loss. In our experience, manufacturers aiming at near-zero dppm are willing to accept an additional yield loss caused by outlier detection of at most 1%.

One way to tackle these problems is to develop new multivariate outlier detection methods that cover all test parameters at once. For example, [3] proposes a multivariate kernel density approach and [4] describes a support vector machine approach. Because estimation errors grow rapidly with increasing dimension, these methods also must select the most important tests on which outlier detection is based. A problem with these advanced methods is that it is much more difficult to understand what is going on and why certain devices are rejected. Therefore, our objective is to provide a framework that can help answering the posed questions and in which each user can still use his favorite outlier detection method(s).

In this framework we use a defect-oriented approach; we want to select tests for outlier detection such that the same failures are detected as if all tests are executed with specification limits. This results in a set-cover-type problem as for example is discussed in [5]. This works because a lot of tests are related and a failure in one test can often be predicted by deviating behavior in other tests as is shown in for example [6]. However, at the same time we enforce that yield loss (caused by outlier detection) remains limited to some small percentage. Finally, the resulting yield loss has apparently similar behavior as failing devices which could be an indication for lower reliability.

The rest of this paper is organized as follows. First, we present a general framework in Section 2. In Section 3, we show an example of an application of our framework. Experimental results of this application are presented in Section 4, after which Section 5 concludes the paper.

II. General Framework

In this section we introduce a general framework for the outlier test selection problem. This model can be used to select tests for outlier detection, and set parameters for the outlier detection methods. The model might also be used for Test Time Reduction by selecting a combination of parametric tests and outlier detection tests, such that other parametric tests can be skipped. Note that we aim to develop a general framework, such that it can be used for different purposes. This also means that some parts of the model may not be necessary in specific cases, but are included for generality.

Key part of the model are the so called *outlier screens*. We define an outlier screen as a combination of an outlier detection method, the test(s) on which it is applied and the corresponding parameter settings. This means that if one outlier detection method is applied on different tests, these are considered as different outlier screens in our model. So, especially with multivariate outlier detection methods, the number of outlier screens in our model can grow very fast if we consider all possible outlier screens. Hence, it is essential to use (heuristic) solution approaches which are able to deal

978-1-4673-5543-8/13/$31.00 ©2013 IEEE

with this large number of outlier screens. In our model we only consider the result of the outlier detection method, this can either be a score indicating the outlyingness or a binary variable indicating acceptance or rejection of each device. Hence, our model can handle any outlier detection method, so each user can use his own favorite method(s).

Another important element of the model is the *target set*. The target set is a subset of devices, which must be detected with outlier detection. This can for example be the set of devices failing specification limits, or devices that are failing during a later test stage (for example burn-in). This target set is necessary for steering the test selection process. Without such a set, it is from mathematical perspective very difficult to determine which tests are good candidates for outlier detection.

Finally, we also introduce a *bonus set*. This set can be used, if there are devices which are not directly targets, but do not need to be included in yield loss computation. For example, in case we only want to consider a specific type of failures.

In the following subsections, we first describe the required input and the objectives and constraints. Then in the third subsection we present our mathematical framework.

A. Input

As input for the model we use parametric test data of a large number of devices. This test data should be continue-on-fail test data, meaning that testing of a failing device is continued when a device fails in one test. Normally, testing is stopped as soon as one test fails in order to save test time. Continue-on-fail data is required because otherwise the tests early in the sequence are almost always selected, because failures in these tests cannot be found with tests later in the sequence.

Furthermore, we need a target set of devices. This target set contains the devices that should be detected. For example, the devices that fail the hard specification limits. However, this target set can also consist of other devices; how the target set is determined does not matter for the test selection model.

Finally, the model requires a set of outlier detection methods that can be performed on the parametric tests. From an abstract perspective, an outlier detection method is just a function of one or more parametric tests, which gives scores on which thresholds can be set to detect outliers. The function can be influenced with some parameters, depending on the outlier detection method. From these outlier detection methods, the outlier screens are computed as a pre-processing step. In this pre-processing, we can control the number of outlier screens, and during the optimization, we also can add additional outlier screens and remove others if necessary.

B. Objectives and Constraints

Based on the described input, the model has to choose outlier screens and corresponding parameters such that the number of detected target devices with the selected tests and methods is maximized while satisfying some constraints. The constraints are:

- Yield loss is limited (detected devices not belonging to the target or bonus set). We implement this constraint per wafer (or lot or batch), in order to obtain a stable solution over different wafers.
- A parametric test is not used in too many outlier screens. Otherwise the influence of a single test may be too high.
- The total number of used parametric tests is limited, this constraint is especially relevant in case the aim is to reduce test time. Another reason is to reduce complexity of the solution; the user can easily check if indeed technically relevant tests are selected.
- The total number of used outlier screens is limited, in order to restrict total computation time and complexity. This constraint can also be generalized, such that total required computation time for all the outlier screens is limited.

C. Mathematical Framework

This section contains all the mathematical notations and definitions, which are used in our framework. The required input consists of the sets and the input parameters. In the framework we have decision variables and state variables. The decision variables are the real variables on which we can decide, state variables are completely determined by the decision variables.

Sets

- T: $\{t_1, \ldots, t_m\}$: set of parametric tests
- W: $\{w_1, \ldots, w_s\}$: set of different wafers (or lots of batches)
- D_a: set of devices of wafer $w_a \in W$
- $TD_a \subset D_a$: set of target devices of wafer $w_a \in W$
- $BD_a \subset D_a$: set of bonus devices of wafer $w_a \in W$
- $D = \bigcap_{w_a \in W} D_a$: set of all devices
- $TD = \bigcap_{w_a \in W} TD_a$, set of all targets
- $BD = \bigcap_{w_a \in W} BD_a$: set of all bonus devices
- M: $\{m_1, \ldots, m_r\}$: Set of outlier screens
- for each $m_k \in M$:
  - P_k: $\{p_1, \ldots, p_{n_k}\}$: Set of parameter vectors ($p_l \in P_k$ can consist of multiple parameters)
  - $T_k \subset T$: the parametric tests on which the outlier screen m_k is based

Input Parameters

- v_{ij}: measurement value of device $d_i \in D$ on test $t_j \in T$
- $s_{ik}(p_l)$: score of device $d_i \in D$ on outlier screen $m_k \in M$ with parameters $p_l \in P_k$
- $b_{ik}(p_l)$: binary score of device $d_i \in D$ on outlier screen $m_k \in M$ with parameters $p_l \in P_k$ indicating if device d_i is rejected
- c_k: computation time of using outlier screen $m_k \in M$
- u_{jk}: binary parameter indicating if parametric test t_j is used in outlier screen m_k. ($u_{jk} = 1$ if $t_j \in T_k$, 0 else)

Decision Variables

- $x_k(p_l)$: binary variable, $x_k(p_l) = 1$ indicates that outlier screen $m_k \in M$ is used with parameter vector $p_l \in P_k$

State Variables (variables determined by decision variables)

- O_a: Set of detected outlier devices in wafer $w_a \in W$ ($O_a \subset D_a$), by the selected outlier screens and parameters. There are different options how this set is determined; for the moment we add all the the devices that are once detected as outlier, but this can easily be extended to more sophisticated rules
- $O = \bigcap_{w_a \in W} O_a$: set of all detected outliers
- $YL_a = |O_a \setminus (TD_a \cup BD_a)|$
  yield loss in wafer $w_a \in W$, number of outliers that are not target or bonus device
- $HT = |TD \cap O|$: number of hit targets
- $HB = |BD \cap O|$: number of hit bonus devices
- $CT = \sum_{m_k \in M} c_k \sum_{p_l \in P_k} x_k(p_l)$
  total computation time for all selected outlier screens
- $UOT = \sum_{m_k \in M} \sum_{p_l \in P_k} x_k(p_l)$
  total number of used outlier screens
- $U_j = \sum_{m_k \in M} u_{jk} \sum_{p_l \in P_k} x_k(p_l)$
  number of times test j is used

Constraints

- $YL_a \leq \alpha * |D_a|$
  Yield loss per wafer must be limited to some small percentage of total number of devices, so usually $0 < \alpha \leq 0.01$
- $CT \leq \beta$
  Total computation time must be limited
- $\sum_{p_l \in P_k} x_k(p_l) \leq 1 \quad \forall m_k \in M$
  Each outlier screen can only be used with one parameter setting
- $UOT \leq \gamma$
  A limited number of outlier screens can be selected
- $U_j \leq \delta \quad \forall t_j \in T$
  Each test can only be used a limited number of times, to avoid that a single test can have too much impact

Objective: $\max (w_H HT - w_Y YL + w_B HB - w_U CT)$

The first part of the objective is to maximize the number of target devices in the set of outlier devices. As this is the primary objective, this part should have a high weight w_H. The other three components are side objectives, which can be added to the objective with smaller weights. For example, if we have maximized the number of hit targets, then afterwards we can look how to minimize yield loss while still attaining the same number of hit targets. We might also include an incentive to find bonus devices with the third part of the objective. Note that w_B should be smaller than w_H and w_Y, otherwise it does not make sense to distinguish between bonus and target devices. Finally, the total computational effort is expressed by the fourth part of the objective. The weights (w_H, w_Y, w_B and w_U) must be set, such that the objectives are optimized in the desired order.

This problem is a variant of (the training phase of) the classical classification problem. In our case we set a constraint on the number of false positives (yield loss) and minimize the number of false negatives (missed targets). Furthermore, we restrict ourselves to certain outlier detection methods as classification rules, since the classification rules must make sense in practical applications. In this way, we also aim to avoid the overfitting risk. Another important observation is that the problem is expected to be very sparse in the sense that the number of detected outliers per test is usually very low, otherwise we would not consider the test as a good candidate for outlier detection.

Furthermore, note that the outcome of the model on the given data set is in practice not the most important outcome; the performance of the resulting selection on other data sets is crucial. So robustness of the outcome is more important than achieving optimal solutions. Finally, we want to remark that the selected tests are a proposal to the user, and the user may modify the given solution.

III. Application

In this section we give an application of the general framework presented in the previous section. In this application we make some modeling choices in order to simplify the problem and do not use all aspects of the more general framework. Furthermore, we consider one specific outlier detection method, which is described in [7]. This method is an online method which updates a robust mean and standard deviation in order to set limits based on deviation from the mean measured in the number of standard deviations. This method has several parameters, but we only let the model choose the tolerance parameter among three options, for the other parameters we use the default values proposed in [7]. Note that the specific choice of outlier detection method is not the scope of this paper and that any method could have been used. The reason for our choice is just that we have an implementation available of that method.

We implemented our model as an Integer Linear Program (ILP) and used the solver CPLEX 12.4 to solve the problem. For the moment we do not consider the use of heuristics for solving the model, since we are mainly interested in the results. Afterwards, we may look at heuristics in order to avoid the use of advanced commercial solvers. The ILP formulation is presented below. In order to simplify the notation we consider each outlier screen with different parameter settings as a different outlier screen. In this model, x_j is the decision variable indicating that outlier screen j is used. As a result of the values for x_j, the variable $count_i$ counts how often a device is detected and the binary variable det_i indicates if device i is detected by the selected outlier screens. The parameter b_{ij} indicates if device i is detected by outlier screen j.

$$\max \; w_H \sum_{i \in TD} det_i - w_Y \sum_{i \notin TD \cup BD} det_i + w_B \sum_{i \in BD} det_i - w_U \sum_{j \in M} x_j \quad (1)$$

subject to:

$$\sum_{j \in M} x_j \leq \gamma \tag{2}$$

$$\sum_{i \in D_k \setminus (TD_k \cup BD_k)} det_i \leq \alpha |D_k| \qquad \forall k \in L \tag{3}$$

$$\sum_{j \in M} b_{ij} x_j = count_i \qquad \forall i \in D \tag{4}$$

$$count_i - det_i \geq 0 \qquad \forall i \in D \tag{5}$$

$$count_i - det_i \gamma \leq 0 \qquad \forall i \in D \tag{6}$$

$$x_j, det_i \in \{0,1\} \qquad \forall j \in M, \forall i \in D \tag{7}$$

The objective function (1) consists of four parts with their corresponding weights: the number of hit targets, the amount of yield loss, the number of hit bonus devices and the number of used screens. Constraint (2) limits the number of selected outlier screens, and Constraint (3) is used to set limits on yield loss per production lot. Constraint (4) is used to compute the state variable $count_i$, which counts the number of times each device is detected. Constraints (5) and (6) are used to ensure that the binary variable det_i is set correctly:

- If $count_i = 0$, device i is not detected by the selected methods. Then constraint (5) ensures that $det_i = 0$ and constraint (6) is always satisfied.
- If $count_i > 0$, device i is detected by at least one selected outlier screen. Then, constraint (5) is always satisfied, but we have to ensure that $det_i = 1$. Note that $count_i$ can attain at most γ, the maximum number of used screens. So, if $det_i = 1$, then $det_i \gamma \geq count_i$ and hence constraint (6) is only satisfied if $det_i = 1$.

In all our experiments, we set $\gamma = 100$ and we set the weights such that we are sure that the different objectives are optimized in the desired order: $w_H = 10^7$, $w_Y = 10^4$, $w_B = 100$ and $w_U = 1$.

IV. Results

In this section we describe some results we obtained with the model presented in the previous section. We applied our model on a real-world continue-on-fail dataset of 10 wafers of in total approximately 60,000 devices. We split this data into a training set of five wafers and an evaluation set of the remaining five wafers. The training set is used to select the outlier screens, after which we apply this selection on the evaluation set in order to check if these results are repeatable.

A. Pre-processing

The dataset contains data of 644 tests. We removed 29 tests because these are not suitable for outlier detection with our method due to their distribution, for example distributions with long tails or discrete values. We also removed the devices failing only in these unsuitable tests from the target set and add these to the bonus set, because we cannot expect to find all those type of failures with outlier detection on the other tests.

Next, we compute outlier detection results on all remaining tests, each with three possible settings for the tolerance level (4, 5 and 6). This leads to 1815 outlier screens, from which we remove all outlier screens which do not detect targets or only detect targets at the cost of high yield loss. With the remaining outlier screens it turned out that seven target devices are not detectable, i.e. these are not detected by any of the outlier screens. However, a closer examination of these seven devices shows that they are so called marginal failures, that is, devices just outside the limits and close to devices passing the test. In Figure 1 an example of a test with such marginal failures is shown. This graph shows test results on a particular test with the horizontal dashed lines indicating the specification limit. Such marginal failures are not likely to be detected with outlier detection (or at high yield loss), but these are most likely not the problematic devices, since the difference between a marginal failure just above the limit and a good device just below the limit seems very arbitrary. In a re-test the outcomes can be completely opposite due to measurement variations. Therefore, we also remove those undetectable devices from the target set.

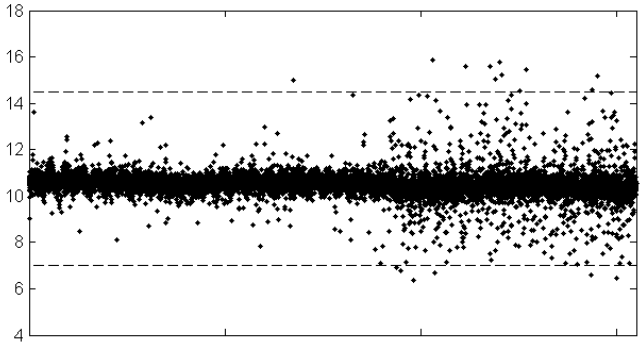

Figure 1. Example marginal failures

B. Yield loss vs. hit targets

In this subsection we analyze the effect of allowed yield loss on the number of hit targets. We start without restriction on yield loss resulting in detection of all targets. Then we restrict yield loss stepwise until 0%. The results of these experiments are shown in Figure 2, where we plot the percentage hit targets and percentage of hit bonus devices versus yield loss percentage (caused by outlier detection). This graph shows that with a yield loss of 1.86% we can detect all targets. If we restrict yield loss to a more acceptable range of something between 0.5% and 1%, the percentage of hit targets is still above 90%. Also, 84% of the targets can still be detected if we do not allow yield loss. We also see the percentage of detected bonus devices is slightly increasing if we increase yield loss. This is because we also maximize the number of detected bonus devices as a side objective (after hit targets is maximized and yield loss is minimized). In Figure 3 a similar graph is shown. In this graph we use the selected screens of Figure 2 and apply these on the evaluation dataset. So the selected tests of each datapoint in Figure 2 correspond with the datapoints in Figure 3. It is surprising that the percentage hit targets remains quite constant between 84 and 89 % for

most datapoints. However, the percentage of detected bonus devices increases if yield loss increases.

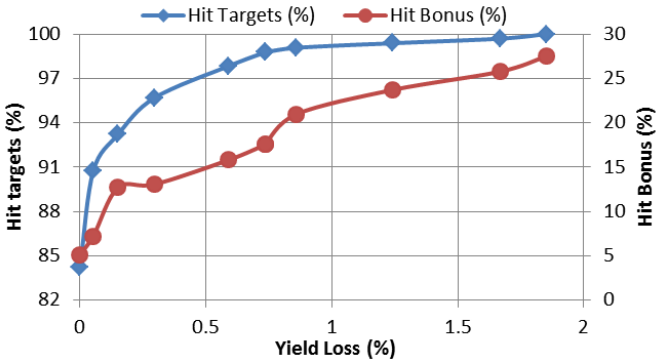

Figure 2. Results Training

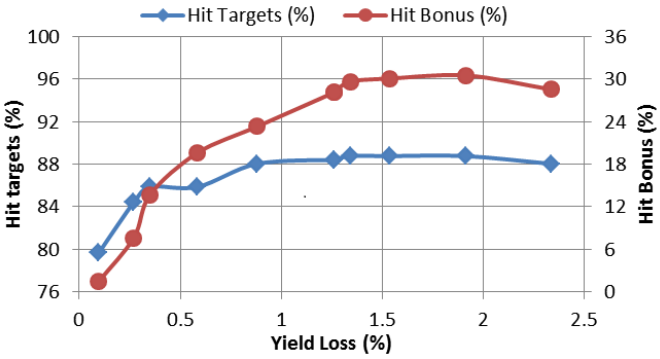

Figure 3. Results Evaluation

C. Missed devices

In this subsection we analyze the missed target devices in more detail. We use the results of our method in which we restrict yield loss to 0.5% per wafer. Then, with a selection of 29 screens, we miss 14 targets in the training set. It turns out that 11 of these 14 missed devices are marginal failures, i.e devices that are just outside the limits, whereas there are several pass devices just on the other side of the limits. Detection of the remaining three devices costs too much yield loss with the allowed tolerance settings, because the distribution of the test in which these devices fail has a quite long tail. These devices could be detected with less yield loss if we would allow a higher tolerance setting in this test.

In the evaluation set the number of missed targets rises to 39. However, if we look at these in more detail, 16 of them are again marginal failures, 2 are missed on the same test as the 3 missed targets in the training set and could be easily detected if a higher tolerance was used. Furthermore, 10 missed targets seem to fail on some tests due to a touchdown problem, rather than they are real failures, since those 10 missed targets are all on three (quad-site) touchdowns. Summarizing, we have 11 clearly failing missed devices left, which should have been detected.

D. Yield loss

In this subsection we analyze the yield loss in more detail. Note that those yield loss devices are most important from outlier detection perspective. Although our approach can be used for test time reduction, our main objective was to select tests for outlier detection. Hence, in practice all specification tests are still executed and all target devices are still rejected. In addition, outlier detection is applied on the selected tests, which will cause some additional yield loss. We use the same 29 selected outlier screens as in the previous subsection, in which we restrict yield loss per wafer to 0.5%. This selection results in a yield loss of 90 (0.3%) devices in the training set and 180 (0.6%) in the evaluation set. Note that the total yield loss in the training set is somewhat lower than 0.5%, because the yield loss restriction is applied per wafer. In Table I we show the percentage of the yield loss of the four outlier screens that cause most yield loss (96% in total) in the training set. Surprisingly, those four screens also cause most yield loss in the evaluation set, although the percentage is decreased to 80%.

Outlier screen 26 is responsible for around 50% of the yield loss, both in the training as in the evaluation dataset. A closer examination shows that this large yield loss is caused by a distribution with a rather long tail in which approximately 40% of those tail devices is either a bonus or target device. Therefore, this test is selected since it is able to detect a large number of target and bonus devices at the cost of a yield loss that remains within the total yield loss constraint. It is interesting to investigate if this deviating behavior of the devices in the tails of the distribution is an indication for lower reliability. The same pattern we see in outlier screen 7, which also causes 20% respectively 14% of the total yield loss in training and evaluation. Outlier screen 6 causes around 10% yield loss in training and evaluation, in the training set all those yield loss devices turned out to be devices in the tail of the distribution, i.e. not really deviating from the other devices, but only at the borders. So, probably the tolerance of this outlier screen should be increased. Indeed, inspection on the evaluation set shows that again mostly tail devices are detected, but also 5 clearly outlying devices which also would have been found with an increased tolerance. Finally, outlier screen 9 causes 9% of the yield loss in training and evaluation. However, this outlier screen is applied on a test with very tight limits. It is probably better to execute this test just with specification limits such that all devices failing this tests are still found. Note that all the above described behavior already could be found by quickly analyzing the results on the training dataset. So, this also shows that with a quick analysis of the results on the training dataset, we can apply some manual changes if necessary, in order to decrease yield loss in the evaluation dataset.

Furthermore, it is surprising that in the evaluation dataset more yield loss devices turned out to be clear outliers on visual inspection (around 10%). In Figure 4, an example of this is shown. Here, we plot the results of one test for the

Outlier screen	%YL training	%YL evaluation
26	59%	47%
7	20%	14%
6	9%	10%
9	9%	9%
rest	3%	20%

Table I
PERCENTAGE OF TOTAL YIELD LOSS

evaluation set, where the horizontal black lines indicate the specification limits. All detected outliers are marked with a circle; the red datapoints are the failing devices; green points are the clearly outlying pass devices and the blue points are the more disputable tail devices. In the training data all yield loss devices are in the tails, but no clearly outlying pass devices are found. Apparently, the test selection in the training set is really fitted to find only failures and limit yield loss as much as possible. However, the selected tests contain useful information to find outliers in the evaluation set. An open question remains if we have found the most important outliers, which will be topic of further research. Furthermore, the fact that only 10% of the yield loss are clearly outlying devices, whereas the other 90% are the more disputable devices in the tails, is also an indication to perform some additional analysis in order to filter only the real outliers.

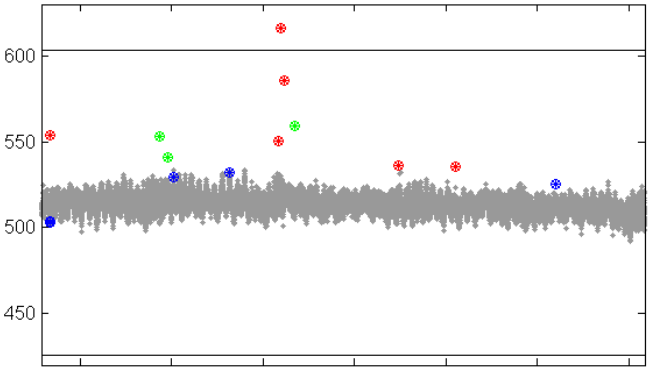

Figure 4. Example Outliers

V. CONCLUSION

In this paper we have presented a framework which can be used for selecting tests for outlier detection. We also applied this framework in a specific case and presented some results. In the evaluation dataset, our method was able to detect 89% of the target devices, with only 29 outlier screens, at the cost of 0.6% additional yield loss. Furthermore, most of the missed devices turned out to be marginal failures. Among the yield loss devices, around 10% are clearly outlying devices, the other 90% are tail devices. Also, the yield loss in the evaluation dataset can be controlled by analyzing the results in the training set. Note that potential use of our method is not limited to test selection for outlier detection. Our method can also be used to gain quick insight in which tests have high prediction power in explaining failures, which then can be used as a way to apply test time reduction.

In the previous section we already touched upon a few issues, which could be improved in future research. First, we adjusted the definition of our target set; devices which we could not find with outlier detection we removed with justification that those devices are marginal failures. Also most missed devices are marginal failures. So, for better validation of our method we need ways to quantify the distinction between marginal failures and real outlying failures. The same holds for the devices passing all specification limits. Some of them are clearly outlying on some tests, those are the devices we would like to detect as much as possible. At the moment, our approach does not include an incentive to find these, which we also saw in the results in the training set. This might also help in answering the question if all (or most) outliers are found by the selected outlier screens. If data is available, it might be very interesting to include burn-in failures or customer returns in the target set and investigate if these could be found with the presented approach.

A second problem might be the potentially large number of outlier screens, especially if multiple outlier detection methods are considered with several parameters to set. The ILP approach we used, is certainly not able to handle a large number of outlier screens, but also for other solution approaches it might help to limit the number of outlier screens.

Finally, as we already mentioned, it also may be worthwhile to implement a two-stage outlier detection approach: First make a rough classification of suspect and good devices, than analyze the suspect devices in more detail to find the real outliers. In that way, we can allow more yield loss in the first step, meaning that it is more likely to find all outliers.

VI. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support for this research of AgentschapNL and the province Overijssel (Pieken in de Delta). Furthermore, we would like to thank Stefan Eichenberger from NXP and Patrick Zwegers and Kees Visser from Salland Engineering for providing data and giving valuable comments.

REFERENCES

[1] P. O'Neill, "Statistical Test: A New Paradigm to Improve Test Effectiveness and Efficiency," in *ITC*, pp. 1–10, 2007.
[2] R. Madge, M. Rehani, and K. Cota, "Statistical Post-Processing at Wafersort - An Alternative to Burn-in and a Manufacturable Solution to Test Limit Setting for Sub-micron Technologies," in *VTS*, pp. 69–74, IEEE, 2002.
[3] E. Yilmaz, S. Ozev, and K. Butler, "Adaptive Test Flow for Mixed-Signal/RF Circuits Using Learned Information from Device Under Test," in *ITC*, pp. 1–10, IEEE, 2010.
[4] D. Dramanac, N. Sumikawa, L. Winemberg, L. Wang, and M. Abadir, "Multidimensional Parametric Test Set Optimization of Wafer Probe Data for Predicting in Field Failures and Setting Tighter Test Limits," in *DATE*, IEEE, 2011.
[5] P. Drineas and Y. Makris, "Independent Test Sequence Compaction through Integer Programming," in *ICCD*, IEEE, 2003.
[6] B. Kruseman, B. Tasic, C. Hora, J. Dohmen, H. Hashempour, M. van Beurden, and Y. Xing, "Defect Oriented Testing for Analog/Mixed-Signal Devices," in *ITC*, pp. 1–10, IEEE, 2011.
[7] H. C. M. Bossers, J. L. Hurink, and G. J. M. Smit, "Online Univariate Outlier Detection in Final Test: A Robust Rolling Horizon Approach," in *ETS*, IEEE, 2011.