Boundary cost optimization for Alternate Test

Gildas Leger
Instituto de Microlectrónica de Sevilla, Centro Nacional de Microelectrónica
Consejo Superior de Investigaciones Científicas (CSIC) and Universidad de Sevilla
Av. Américo Vespucio s/n, 41092 Sevilla, Spain.
e-mail: leger@imse-cnm.csic.es

***Abstract*—Alternate Test has demonstrated in the last decade that advanced machine-learning tools can leverage the accuracy gap between functional test and indirect, or model-based, test. If a regression approach is taken, a model should be trained for each specification. The advantage is that the results are interpreted just like performance measurements but the drawback is that accuracy is required over the full variation range. On the other hand, a classification approach can be seen as a wiser solution since it locates the pass/fail boundary, which inherently contains all the specification information, in the cheap measurement space. Cost optimization due to imbalance between test escape and yield loss is usually handled by guard-banding on specifications. This is straightforward to translate to regression-based Alternate Test but not for classification-based.**

This paper shows that two different asymmetric approaches consistently outperforms an off-the-shelf symmetric algorithm. The first technique is based on manipulating the decision threshold while the second technique directly builds an optimized pass-fail boundary by considering different costs to penalize test escapes and yield losses.

I. Introduction

The test of Analog, Mixed-Signal and RF circuits (AMS-RF) is considered a bottleneck for the development of complex Systems-on-Chip (SoCs). Whenever it is feasible, the industry relies on functional testing where every single specification on the datasheet is measured using standardized procedures. The goal is to achieve the best possible quality.

The test metrics that characterize test quality are not direct expressions of the measurement accuracy. By the way, they are indirectly related to the specifications: The test escapes represent the circuits that pass the test despite failing one or more specifications. Similarly, yield losses represent the circuits that fulfill all the specifications but fail the test. These two concepts can be expressed relatively to their reference population or not. For instance the test escape ratio is the proportion of bad circuits that fail the test over the total number of bad circuits while the defectivity level is the proportion of bad circuits that fail the test over the total number of fabricated circuits. These two metrics are related by the underlying manufacturing yield (in what follows, the yield Y) which is the ratio of fabricated devices that fulfill all the specifications.

It may seem obvious to design the test to minimize both errors equally. However, as it all boils down to money, the overall cost/benefit relation should be taken into account. This relates to test quality but also includes the probability of occurrence of defects as well as cost imbalance. The unit cost of a customer return is much higher than the unit selling price of the device. Intuitively, we understand that it may be beneficial to reduce the number of test escape even at the cost of an increased yield loss. On the other hand, if the manufacturing yield is high, it is possible to relax the test escape ratio. For a defectivity target of 100ppm, if the yield is 90% the test escape ratio should be 0.1%. However, if the yield is 99% the required test escape ratio is of only 1%.

All of this is a matter of optimization and for traditional functional testing it is handled by guard-banding the specifications [1]. Measurements are affected by noise, or more generally speaking, limited accuracy. By setting test limits a little bit more stringent than the specifications we can force that almost the entire error distribution falls within the failing zone. Guard-banding is thus carried out as a function of the measurements standard deviation.

In the past decade, the concept of Alternate Test [2], [3] has found quite a large audience. It consists in using powerful machine-learning algorithms to build a mapping between simple measurements (or at least a reduced set of measurements) and the specifications. In this way the cost of performing the test can be drastically reduced. If regression tools are used, a different model has to be built for each specification and the same guard-banding procedure as for functional test can be used. The only difference is that the measurement errors are replaced by the model generalization errors.

However, when several specifications are involved, it may be wiser to make use of classifiers instead of regressors [4], [5], [6], [7]. Internally, machine-learning classifiers work with the same principles as regression tools, but they do not focus on the error on the entire range. Instead, they try to locate the best possible pass-fail boundary in the space of the simple measurements. For classifiers, the quality is not measured as a generalization error but as a misclassification errors, just like for production test quality metrics. False negatives are yield losses while false positives as test escapes.

Off-the-Shelf classifiers have usually been built for a generic purpose and there is no a-priori reason to prefer a false negative rather than a false positive. The target of their internal algorithms is thus to minimize the overall classification error (probably subject to some form of regularization). However, in the particular case of IC testing, test escapes and yield loss do not have the same consequences: consumer returns are much more harmful to the image of a company than the simple loss of not selling a device that was good. The location of

the pass-fail boundary should thus take into account this cost imbalance.

In this paper, we will show that it is not the best practice to use off-the-shelf classifiers. Since the yield is usually not 50% and the cost of test escapes and yield loss are not the same, it is better to use an asymmetric classifier, which will take these costs into account for building an otpimum decision boundary. Another possibility to artificially create imbalance in the results would be to alter the decision threshold of a conventional classifier. We will also show that this strategy is a better option than the off-the-shelf classifier though not optimum in most cases.

II. Cost asymmetric classifier

A. A brief description of Adaboost

Adaboost is the acclaimed classifier introduced by Shapire back in 1997 [8]. This classifier is based on boosting: It is a combination of weak learners built in such a way that each weak learner g_m brings an additional bit of information.

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m g_m(x)\right) \tag{1}$$

A weak learner is a low-accuracy classifier, whose classification error is slightly better than a random guess. Typically, Adaboost is implemented with decision trees or even decision bumps (a tree of order 1) but this is not a conceptual restriction. In theory, any classifier can be used – as commented in [9], where Adaboost is considered as an ensemble classifier.

The main reason for its success is that it has been shown to be almost immune to overfitting. In the majority of machine-learning algorithms, when complexity is gradually increased the generalization error usually begins to drop as the model gets closer to the training data structure but after an optimum it usually increases. The reason is that the model starts learning the particular noise pattern of the training set as if it were a characteristic of the data. On the test set the noise pattern is different and the model output is thus worse. Due to the particular characteristics of boosting, this overfitting trend is almost negligible. Notice that this resilience to overfitting is linked to the use of weak classifiers. If more elaborate classifiers are used, this property may be compromised.

The target of the learning algorithm is to minimize the exponential loss:

$$loss = \sum_{i=1}^{N} e^{-G(x_i)y_i} \tag{2}$$

where y_i the true class (either 1 or -1) of sample i.

It can be shown [6] that this minimization can be carried out by a simple algorithm because the value of the weak learner contribution (the α_m in (1)) can be separated from the selection of the weak learner itself. The iterative algorithm works as follows:

- initialize the N sample weights of the training set to a uniform distribution,

$$w_i = 1/N \tag{3}$$

- for $m = 1$ to M (the number of weak learners)
  Select the best possible weak learner g_m, that is to say, the learner that gives the lowest weighted misclassification rate:

$$err_m = \frac{\sum_{i=1}^{N} w_i I\left(y_i \neq g_m(x_i)\right)}{\sum_{i=1}^{N} w_i} \tag{4}$$

  where I is the indicator function.
  Compute the weak learner contribution to the final decision

$$\alpha_m = log\left(\frac{1 - err_m}{err_m}\right) \tag{5}$$

  Update the sample weights as follows

$$w_i \leftarrow w_i \times exp\left(-\alpha_m g_m(x_i)\, y_i\right) \tag{6}$$

- Output the final classifier as a combination of the weak decisions, as defined in (1)

Basically, Adaboost iteratively fits weak learners to the data but updates the weight of the training samples at each step in such a way that the relevance of misclassified samples is increased while that of correctly classified samples is decreased. This can be seen in (6): as α_m is strictly positive, when the output of the weak classifier $g_m(x_i)$ is of the same sign as the true class y_i the exponential is lower than 1, whereas when they are different it can be significantly higher than 1. The higher the error, the higher the value of α_m, and thus the higher the sample weight correction.

B. Decision Thresholding

A straightforward way of causing imbalance in a classifier is to consider a variable threshold on the class likelihood. The class likelihood is the probability that the device is failing (or conversely passing) knowing the measurement results. It is thus the conditional posterior probability of the samples. Many classifiers, and in particular Adaboost, produce an output that is in reality an estimate of the class likelihood. The decision is thus simply taken by comparing this output to the probability of a random draw (50%). It appears quite intuitive that if we tag as good only devices that have a high passing probability (say 90%) we will displace the pass-fail boundary towards the good population and thus decrease the false positive rate. The unavoidable consequence is an increase of the false negative rate. Conversely, if we tag as bad only devices that have a low passing probability we will displace the pass-fail boundary towards the bad population and thus increase the false positive rate.

Thresholding is of particular interest for some applications where the objective is to control the test escape probability rather than explicitly optimizing a cost. In a recent paper [10], an adaptive alternate test procedure was proposed where incremental models were constructed for each newly acquired feature in the test sequence. These models were used to perform early diagnosis but strict guardbanding should be taken to limit test escape penalty. Though the authors applied

the proposal to a regression case, thresholding was suggested as a way to control the guardband for classifiers.

Though thresholding is applicable to any sort of classifier (whenever it gives a probability-like output), we will apply it to Adaboost in order to perform a fair comparison. For Adaboost, we simply have to consider the value of the quantity prior to applying the sign function in (1). If these weights are normalized during the construction of the model, this quantity can be viewed as an affine mapping from the interval $[-1, 1]$ to the probability range $[0, 1]$. In this way, the 50% probability threshold corresponds to 0.

C. Making Adaboost asymmetric

The basic idea of the asymmetric Adaboost proposed in [11] is to introduce a different weight modification to the misclassified devices whether they are false positives or false negatives.

Let us define I_p and I_n, the subsets of good and bad samples in the training set, respectively. The asymetric weight update would be of a from similar to (6):

$$\begin{aligned} w_i &\leftarrow w_i \times exp\left(-C_1 \alpha_m g_m\left(x_i\right)\right) \;\; i \in I_p \\ w_i &\leftarrow w_i \times exp\left(C_2 \alpha_m g_m\left(x_i\right)\right) \;\; i \in I_n \end{aligned} \tag{7}$$

To evolve the Adaboost classifier, we must add the weak classifier g_m, with contribution α_m that minimize the modified exponential loss,

$$\begin{aligned} err &= E_{Y|X}\left(I_p w_p e^{-\alpha C_1 g(x)} + I_n w_n e^{-\alpha C_2 g(x)}\right) \\ &= \left(e^{C_1\alpha} - e^{-C_1\alpha}\right) b + T_p e^{-C_1\alpha} \\ &\quad + \left(e^{C_2\alpha} - e^{-C_2\alpha}\right) b + T_n e^{-C_2\alpha} \end{aligned} \tag{8}$$

where,

$$\begin{aligned} Tp &= \sum_{i \in I_p} w_i \\ Tn &= \sum_{i \in I_n} w_i \\ b &= \sum_{i \in I_p} w_i \times I\left(y_i \neq g_m\left(x - i\right)\right) \\ d &= \sum_{i \in I_n} w_i \times I\left(y_i \neq g_m\left(x - i\right)\right) \end{aligned} \tag{9}$$

The main issue of the cost-sensitive Adaboost is that the search to find the optimum weak learner and its associated coefficient cannot be split into two independent steps to maintain optimality.

The cost-sensitive algorithm is thus as follows:

- We first initialize the sample weights of the two subsets as an uniform distribution,

$$\begin{aligned} w_i &= 0.5/N_p \;\; \forall i \in I_p \\ w_i &= 0.5/N_n \;\; \forall i \in I_n \end{aligned} \tag{10}$$

- for $m = 1$ to M (the number of weak learners)
  Perform a search over the possible weak learners g_k considering the sample weights.
  For each instance of the search, compute the contribution step α_k as the value that makes the derivative of the error equal to zero, which is the solution of:

$$\begin{aligned} &2bC_1 cosh\left(C_1\alpha\right) + 2dC_2 cosh\left(C_2\alpha\right) \\ &= C_1 T_p e^{-C_1\alpha} + C_2 T_n e^{-C_2\alpha} \end{aligned} \tag{11}$$

  Once the step value is known, compute the associated error as,

$$\begin{aligned} err_k = &\left(e^{C_1\alpha_k} - e^{-C_1\alpha_k}\right) b + T_p e^{-C_1\alpha_k} \\ &+ \left(e^{C_2\alpha_k} - e^{-C_2\alpha_k}\right) b + T_n e^{-C_2\alpha_k} \end{aligned} \tag{12}$$

  Finally select the pair of weak learner g_m and associated contribution step α_m that minimizes the error.
- Output the final classifier as a combination of the weak decisions, just like in (1)

Obviously, C_1 is the cost associated to false negative and C_2 the cost associates to false positive. In what follows we will refer to C_{fn} and C_{fp} for the sake of clarity. It appears clearly that the computational cost is higher than for the conventional Adaboost since the solution of (12) requires numerical approximation (and is thus a nested search loop). Since our Matlab codes are not fully optimized, we prefer not to give any value for the impact on efficiency, but we can say that the increase is manageable (not an order of magnitude).

D. Other approaches

Some papers have already dealt the topic of boundary guardband in a somewhat indirect way. In [4], two defect filters – a strict one and a lenient one – are built to assess which devices should be rejected directly, should pass through a regression test or should be submitted to additional tests. These defect filters are in fact one-class classifiers based on a joint-probability density estimate. The devices subjected to the filters are considered to be correct whenever the probability estimate is higher than zero. In that paper, instead of applying thresholding to the probability estimate, the strictness of the filters is controlled by a parameter that defines the size of the kernel associated to each training sample. Since the definition of the kernel is somewhat arbitrary, the boundary can hardly be considered optimal.

In [5], the case of population imbalance is commented: for high yields, the number of good devices is much higher than the number of bad devices in the training set of the classifier. This usually tends to favor test escapes since the faulty circuit appear to be statistically less important. A proposed solution consists in replicating faulty devices to artificially generate a symmetric population. Though the biasing effect is clear, here again, the optimality of the decision cannot be assessed easily.

III. An Alternate Test experiment

A. Cost Model for validation

In order to compare the results obtained by different classification approaches, we need to define an appropriate cost model.

Let us begin by the simplest part. The amount of money earned is the number of sold devices times the unit selling price (U_{price}). And the number of devices sold is the number of devices that pass the test. Among the sold devices, we have those that are actually correct and those that have passed the test but are bad devices: these are the test escapes. Similarly we can consider that the number of good devices that pass the test is the full number of good devices minus the number of good devices that fail the test: these latter are the yield losses. So it comes,

$$\begin{aligned} benefit &= U_{price} \frac{N_{good} - N_{good|fail} + N_{bad|pass}}{N} \\ &= U_{price} \left(Y - YL + TE\right) \end{aligned} \quad (13)$$

Notice that the first term is independent of the test strategy: it is the true number of good devices fabricated, the manufacturing yield. We could argue that in some cases the decision of the specification values may be linked to marketing considerations. The value of Y would thus vary with the chosen specifications and could be introduced into the optimization. However, this is beyond the scope of our study: we consider a fixed specification that will not enter into the cost optimization. We can thus obviate it.

Then we have to model the cost of customer returns. In a first approximation, we may want to consider that all test escapes are customer returns and we would thus have the simple cost function,

$$cost = U_{return} \times TE \quad (14)$$

However, it seems reasonable to think that customer returns are more unlikely if the performance of the device is close to the specification. Indeed, if the device is to be used in a larger system its specification may have been guard-banded. It is possible that a customer may never verify that the part is not performing as in the datatsheet because its performance is good enough. Similarly, some parts may have been purchased as spare parts and never actually be used. We thus need a function to model the probability of customer returns. This is probably application-dependent sensitive information for the industry and this modeling task shall be done in-house. However, without loss of generality, we will consider that the probability of a customer return knowing that the device is actually faulty (and thus a test escape) is of the form

$$P\left(return|TE\right) = \beta \left(1 - \prod_{i=1}^{n_p} e^{-\alpha_i \left| \frac{s_i - s_{pi}}{s_{pi}} \right|}\right) \quad (15)$$

With such a function, we can see that test escapes that are much worse than the specifications will be almost always detected. On the contrary, if the performance is sufficiently close to the specification, the customer will not be able to see the malfunction. This asymptotic rate is controlled by parameter α_i for each specification. The factor β should be lower than 1 and represents the fraction of devices that will actually be tested or used by the consumer.

We can now express the cost of customer returns as,

$$\begin{aligned} cost &= U_{return} \int P\left(return|TE\right) P\left(TE\right) dTE \\ &= \sum_{j \in [escapes]} \beta U_{return} \left(1 - \prod_{i=1}^{n_p} e^{-\alpha_i \left| \frac{s_{i,j} - s_{pi}}{s_{pi}} \right|}\right) \end{aligned} \quad (16)$$

As a result, we can write the total cost balance, which should be minimized, as

$$cost_{balance} = cost - benefit \quad (17)$$

Neglecting the terms that do not depend on test, we get a simpler cost function,

$$\begin{aligned} cost = &\sum_{j \in [escapes]} \beta U_{return} \left(1 - \prod_{i=1}^{n_p} e^{-\alpha_i \left| \frac{s_{i,j} - s_{pi}}{s_{pi}} \right|}\right) \\ &- \sum_{j \in [escapes]} U_{price} + \sum_{j \in [yield\ losses]} U_{price} \end{aligned} \quad (18)$$

It appears that the parameter β, which was introduced to quantify the fraction of the devices that would never be tested by the customer, has the exact same influence as the return cost. In what follows, we will thus set it to one, without loss of generality.

B. LNA Alternate Test

In order to observe the possible impact of cost-sensitive boundaries, it is necessary to consider a case of study. Ideally, we would have chosen a realistic case of an analog circuit with high yield and selected the input features that would give the best possible classifier performance. This would yield to low dppm levels (defective parts per million). Unfortunately, such a situation requires a very large number of MonteCarlo simulations since we want to examine the variation on the test escape and yield loss levels.

On the other hand, cost sensitive pass-fail boundary is a concept that is independent of model accuracy: whatever the absolute misclassification rate, we should be able to modify the relative contributions of test escapes and yield.

So for the sake of statistical validity, we decided to sacrifice the realism of the test situation and consider a very unfavorable case. The Circuit Under Test is an LNA and the signatures are 9 DC probes augmented with 4 simple tests related to the passive devices. These passive tests are carried out on dummy structures for capacitors and resistors, as proposed in [12], but the inductor test relies on another mechanism since replicating the inductor would not be cost-effective. As a matter of fact, a number of additional signatures is available, for instance the input and output of a co-designed envelope detector but also the same signatures obtained at different supply voltages [13], [14]. We deliberately restricted ourselves to using few signatures to observe statistically meaningful variations on the misclassified circuits for the 2000 simulated samples. These samples are split in a training and testing set of equal size.

Similarly, we purposely set the specification values to demanding levels in order to get a relatively low manufacturing yield. Indeed, it is known that the misclassification rate

TABLE I
LNA SPECIFICATIONS

Specification	value	# failing devices
gain (in dB)	11	73
IIP3 (in dB)	-7.5	285
NF	4.2	54

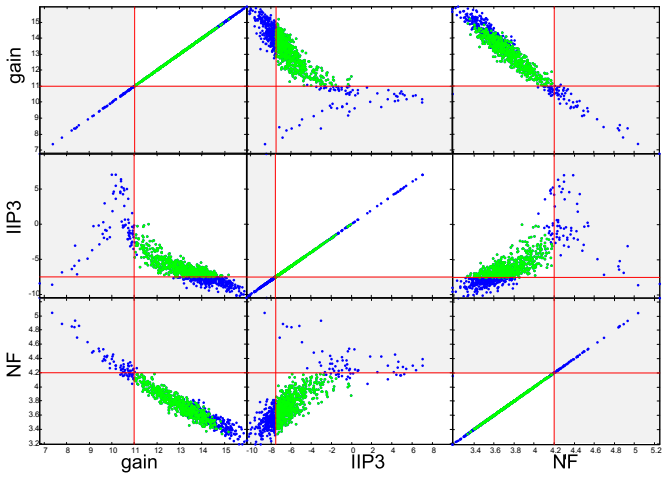

Fig. 1. Correlation of the LNA performance metrics. The green points are the circuits that pass the 3 specifications.

depends on the classifier accuracy but also on the density of probability of the circuit population in the vicinity of the pass-fail boundary. We considered three different performance parameters as are the gain, the noise figure (NF) and the third order Intermodulation Intercept Point (IIP3) and their specifications are summarized in Table I.

In this way we obtain a yield level in the range of 65%. It is also important to notice that we set the value of each specification such that they all impact the pass-fail boudary. Indeed, if a given specification is relaxed, or significantly guard-banded by design, it may happen that it never cause any failure under process variations because the circuits that fail such a specification already fail others. This could have been the case for the gain and the Noise Figure which are highly correlated, as can be observed in Fig.1. It can be observed how in each cloud of points the good devices are limited by the two specifications.

The first experiment consists in verifying the cost sensitivity of the modified Adaboost algorithm. We trained several models varying the cost ratio between false positives and false negatives (C_{fp}/C_{fn}) and evaluated these models over the test set. The obtained results are plotted in Fig.2. First of all, the topmost curve shows the total number of misclassified devices which is seen to remain relatively stable around 100 for moderate cost imbalances. This represents a 10% error but keep in mind that we have selected purposedly a worst case model and yield situation. Thanks to this high error level, we can observe clearly that the number of false negatives and false positives cleary follow the expected trends.The minimum number of test escapes (i.e. false positives) is only 9 for a cost

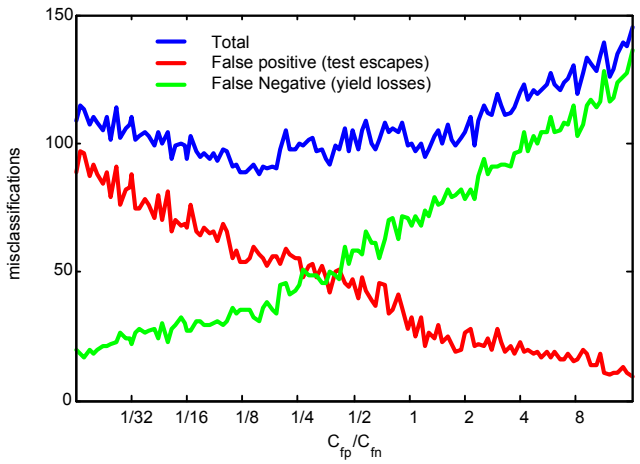

Fig. 2. Mis-classifications versus cost imbalance ratio

imballance of 16. The test escape rate is thus reduced by a factor 5.

It is worth noticing the counterintuitive fact that a balanced cost ratio does not lead to symmetric false positive and false negative rates. Our guess is that the sample density in the feature space influences the results. Roughly speaking, since there are less bad circuits than good circuits (and since the distributions of the two populations are different), the bad circuits should be assigned a slightly higher weight. The very principle of Adaboost training states the link between cost and sample weights through the update rule (6).

We have also trained a conventional Adaboost model in the exact same conditions as in the previous experiment, varying the decision threshold. Since this threshold cannot be compared directly to the cost ratio, Fig.3 presents the false negatives versus false positives obtained in the two experiments. It is clearly seen that the cost-optimized boundary obtains better results, whenever the imbalance is not too extreme. Beyond a certain point the two techniques offer similar results. This is understandable since the asymptotic behavior is the same in both cases: Ultimately, if test escapes are infinitely more expensive than yield losses all the devices would be tagged as bad. Similarly, if the consider a device as good only is the likelihood of being good is 1, all the devices would also be tagged as bad.

To further validate the optimality of the proposed approach, we have to consider the cost function defined in the previous section. In this way, we can make a fair comparison between the proposed approach and the most traditional one. For this experiment, we consider that the cost of a return (U_{return}) is ten times as high as the selling price (U_{price}) in (19). We set the sensitivity parameter of the different specifications to the same value, and vary this value from 0.001 to 1000.

$$\alpha_{gain} = \alpha_{IIP3} = \alpha_{NF} \in \left[10^{-3}; 10^{3}\right] \tag{19}$$

For each value of α, we compute the cost associated to the different thresholds in the case of the thresholding approach

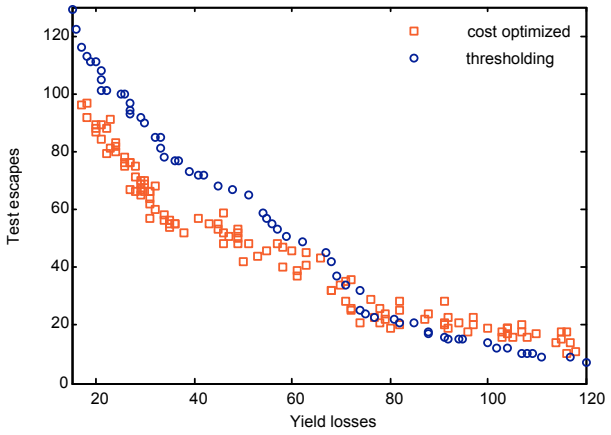

Fig. 3. ROC curves for cost-sensitive Adaboost ans for conventional Adaboost with decision thresholding

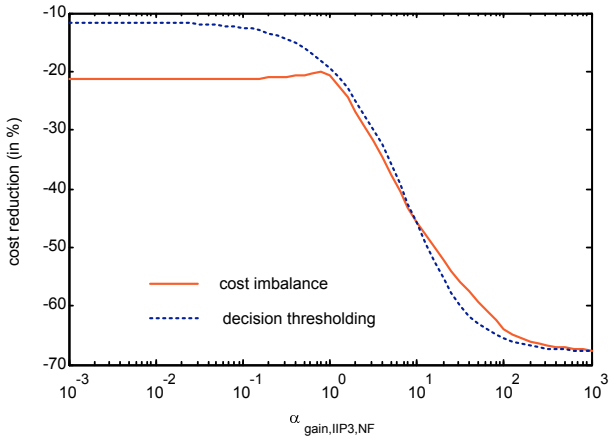

Fig. 4. Minimum achievable cost as a function of the parameter sensitivity, for the conventional thresholding method and for the proposed cost-optimized boundary

and to the different cost ratios in the case of the proposed approach. In both cases, the final selected model for a given α is the one that gives the lowest cost. In other words, we do not consider the cost ratio or the threshold as fixed a-priori but rather perform a search on all the possible values to get the best result.

Fig.4 displays the obtained results. These are presented as a variation with respect to the cost obtained by an off-the-shelf Adaboost classifier. It can be clearly seen how both the proposed methods outperform the conventional classifier, offering a cost reduction between 10% and 70%.

When the sensitivity parameter is low, which gives a high detection probability (and thus customer return probability) to the escapes located farther from the specification than those located closer, the cost-asymmetric Adaboost classifier outperforms the decision thresholding approach. We can thus infer that the escapes that occur with the former are less numerous or are closer to those produced by the latter.

On the other extreme, when the sensitivity parameter is high, all the escapes are almost equivalent and at some point the thresholding method outperforms our approach, though the difference is relatively marginal. As a matter of fact, both methods saturate to an almost similar value.

IV. Conclusion

It has been shown in this paper that go/no-go Alternate Test based on machine learning classifiers may significantly benefit from the introduction of asymmetric cost considerations within the pass/fail boundary learning process. Indeed, our experiments on a LNA showed that the cost-optimized classifier, either through the straightforward technique of decision thresholding or through the use of a built-in test escape/yield loss imbalance, consistently outperforms a similar symmetric classifier in 10% to 70% over a wide range of cost model parameters.

Acknowledgment

This work has been partially funded by the Ministerio de Economia y Competitividad project TEC2011-28302, co-financed by the FEDER program.

References

[1] G. Roberts and S. Aouini, "An overview of mixed-signal production test from a measurement principle perspective," *IEEE Design & Test*, pp. 1–1, 2013.

[2] P. N. Variyam and A. Chatterjee, "Enhancing test effectiveness for analog circuits using synthesized measurements," in *Proc. of IEEE VLSI Test Symp.*, 1998, pp. 132–137.

[3] P. N. Variyam, S. Cherubal, and Chatterjee, "Prediction of analog performance parameters using fast transient testing," *IEEE Trans. on CAD*, vol. 21, no. 3, pp. 349–361, 2002.

[4] H. Stratigopoulos and S. Mir, "Adaptive Alternate Analog Test," *IEEE Design & Test of Computers*, vol. 29, no. 4, pp. 71–79, 2012.

[5] H.-G. Stratigopoulos and Y. Makris, "Nonlinear decision boundaries for testing analog circuits," *IEEE Trans. on CAD*, vol. 24, no. 11, pp. 1760–1773, Nov. 2005.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction.*, 2nd ed. Springer, 2009.

[7] W. M. Lindermeir, H. E. Graeb, and K. J. Antreich, "Analog testing by characteristic observation inference," *IEEE Trans. on CAD*, vol. 18, no. 9, pp. 1353–1368, 1999.

[8] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[9] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[10] G. Leger, "Combining Adaptive Alternate Test and Multi-Site," in *Proc. of Design, Automation and Test in Europe Conference(DATE)*, Mar. 2015, pp. 1–6.

[11] H. Masnadi-Shirazi and N. Vasconcelos, "Cost-sensitive boosting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, pp. 294–309, Feb. 2011.

[12] L. Abdallah, H. G. Stratigopoulos, C. Kelma, and S. Mir, "Sensors for built-in alternate RF test," in *Test Symposium (ETS), 2010 15th IEEE European*, 2010, pp. 49–54.

[13] M. J. Barragan, R. Fiorelli, G. Leger, A. Rueda, and J. L. Huertas, "Alternate test of LNAs through ensemble learning of on-chip digital envelope signatures," *Journal of Electronic Testing*, vol. 27, no. 3, pp. 277–288, Jan. 2011.

[14] M. Barragan, R. Fiorelli, G. Leger, A. Rueda, and J. Huertas, "Improving the accuracy of RF alternate test using multi-VDD conditions: Application to envelope-based test of LNAs," in *IEEE Asian Test Symp.*, Nov. 2011, pp. 359 –364.