Outlier Detection for Large Scale Manufacturing Processes

Abhinav Jauhri
Carnegie Mellon University
Pittsburgh, PA
Email: ajauhri@cmu.edu

Bradley McDanel
Harvard University
Cambridge, MA
Email: mcdanel@fas.harvard.edu

Chris Connor
Intel Corporation
Hillsboro, OR
Email: chris.connor@intel.com

***Abstract*—Integrated circuit manufacturing consists of tests at various stages to ensure functionality and performance using numerous test metrics for each system on chip (SoC) captured as part of assessment. At a later stage, functional units are evaluated in terms of multiple performance characteristics. In this paper, we propose a system that uses test metrics as features for machine learning models to predict the performance characteristics of each SoC. We show that these models are robust against erroneous or noisy signal in test metrics and provide accurate prediction. Given accurate models, we build a system that automatically detects systematic changes in the manufacturing process from week to week and identifies wafers, a grouping of patterned dies in the fabrication process, which have significantly higher than average prediction error and label them as outliers. These outliers are analyzed in order to determine the cause of the discrepancy and to assess potential problems in the manufacturing process. The system has been proven applicable across multiple products and process technologies.**

I. Introduction

The manufacturing process for a SoC stretches across many weeks during which time it undergoes a series of tests at each stage. Figure 1 depicts a simple manufacturing flow. The metrics captured during these tests are numerical values corresponding to specific components (e.g. graphics) on the SoC. These metrics help in the detection of operational anomalies within a SoC. Only the units (SoCs) which have acceptable test metrics are allowed to proceed to the next stage of the manufacturing process. Microprocessors with insufficient test metrics are dropped from the manufacturing pipeline. A later stage of the manufacturing process measures performance metrics such as *yield*, the percentage of good die in a wafer, and *parametric*, the performance of a component (e.g. core or graphics) on the SoC at a set frequency. While the test metrics are able to identify outright defects, they do not provide a high level view of potential systematic problems in the testing process overall. In other words, these metrics do not summarize large scale systematic changes in the manufacturing process that may occur from one week to the next. Additionally, these metrics are at an individual SoC or wafer level, and do not directly measure the health of a wafer when compared to the population.

This work focuses on applying machine learning techniques to predict performance metrics (*yield* and *parametric*)

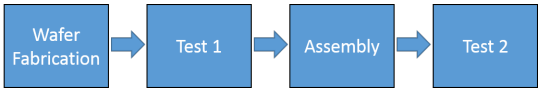

Figure 1. An overview of the manufacturing flow. The test metrics measured at Test 1 are used to predict the performance metrics measured at Test 2.

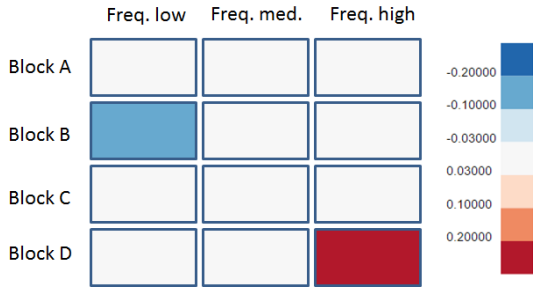

Figure 2. Diagram of the interface discussed in section VI, showing the prediction residuals for the *parametric* of various SoC block components assessed across measurement frequency. The blue and red tones indicate under- and over-prediction respectively.

using metrics from an earlier test as features for the machine learning models. These models, which we treat as a *gold standard*, are used to identify systematic changes in the manufacturing process and identify wafers with substantial prediction error as outliers. Figure 2 shows an example of the user interface discussed in section VI for *parametric* predictions of a SoC.

Additionally, these models provide insight into which test metrics are significantly different for the outlier as compared to the entire population. This gives engineers a starting point when trying to diagnose the potential issue, since each test metric is associated with a specific part of the SoC. During high volume manufacturing of millions of SoCs, machine learning provides a very agile platform for instant assessment and action. The system has been applied to several of the latest generation processor models at Intel and providing weekly feedback to engineers on the health of the manufacturing process.

II. Related Work

Frameworks developed by [1] for monitoring health of equipments used in manufacturing processes apply statis-

tical techniques like mean drift and variance inflation on equipment parameters recorded when a wafer passes through them. In our work, manufacturing processes use performance metrics of SoCs in form of feedback to take corrective measures. Manufacturing processes have been modeled using Fuzzy Regression [2] for instances where there is a scarcity of data. To our knowledge, this is the first study on developing an automated framework for diagnosing potential issues in a manufacturing process using machine learning models which are generated from millions of SoCs.

III. Problem Definition

Detailed information about a single unit is captured at each stage of the manufacturing process including metric values and a completion timestamp. For the purpose of prediction, units are grouped by the time frame (week) that they go through the test 2 (performance) stage from Figure 1. We use units processed in the previous weeks to predict units that have not yet been processed through the performance stage and will be processed in the following weeks. Let test metrics for any week i be represented by $X^i \in \mathbb{R}^{n \times m}$ where n is the number of units being processed and m is the number of test metrics associated with each unit. For yield calculation we classify each unit as good or bad and aggregate the number of good units in a wafer to get the *predicted yield*. Therefore, let the label associated for n units from all wafers which went through the performance stage in week i be denoted by $y^i \in \{bad, good\}^n$. Let the *parametric* for a specific component (e.g. Block A) associated with n units for week i be denoted by $v^i \in \mathbb{R}^n$. Training data is taken from the previous weeks using sliding window of size s where $1 \leq s \leq 5$. For instance to make predictions for week w, the model will be trained using metrics from s weeks: $(X^{w-s}, v^{w-s}), ..., (X^{w-2}, v^{w-2}), (X^{w-1}, v^{w-1})$. The bounds on s were decided from trial and error experiments such that s captures the dynamics of the manufacturing process.

IV. Model Selection

In this section, we discuss the choice of models for predicting *yield* and *parametric*. Due to different dimensionality of the test metrics and domain of the response variables (*yield* or *parametric*), we use different techniques to model each problem. The test metrics used as features are problem specific and were picked by experts using domain knowledge.

A. Yield Prediction

Yield prediction is a binary classification problem with the labels $\{bad, good\}$. Accuracy of a model is measured by the total number of units classified correctly for a week:

$$acc_y = \frac{\sum_{i=1}^{n} y_i^w == \widehat{y}_i^w}{n} \quad (1)$$

$\widehat{y}_i^w$ is predicted label of ith unit which went through the performance stage in week w. Only two test metrics are used as features for this prediction problem. The features used to classify a die are illustrative of its neighbor's functionality and the die's parametric performance. We use a nearest neighbor approach (k-NN) to model this problem. Acceptable accuracy was attained with $k <= 10$. For $k > 10$, we observed that the accuracy began to deteriorate. To improve the accuracy further, we applied boosting using weighted k-NN. Formally, a booster is provided with a set of labeled training examples $(x_1, y_1), \ldots, (x_n, y_n)$, where y_i is the label associated with instance x_i. On each round $k = 1, \ldots, K$ (with $K <= 10$), the booster devises a distribution D_k over the set of examples, and requests a weak hypothesis (k-NN in our case). D_k specifies the relative importance of each example for the current round. After K rounds, the booster combines weak hypotheses into a single prediction rule. To speed up training of models, and weekly prediction results, we used Fast Library for Approximate Nearest Neighbors [3].

B. Parametric Prediction

Parametric prediction involves a larger set of test metrics (≈ 1000). We chose Random Forest regression for this problem [4]. For this task, one of the most important factors in the selection of a model is the ability to rank the importance of the test metrics used by the model. This has a practical impact as the models serve as a tool for the engineers to understand potential issue in the manufacturing process. Additionally, we can use the metric ranking to understand the major difference between an outlier as compared to the general population. Random Forests provide a very natural ranking of metric importance by determining how much each feature contributes to reducing out of sample measurement error. In practice, we can show only the top ≈ 10 metrics of a model as a way of explaining what are the important metrics. Additionally, Random Forests are able to handle the large volume of data and can be quickly updated from week to week.

C. Model Update Criteria

Due to the dynamics of the manufacturing process, it is important to be able to adjust the models to account for new data each week. In general, however, the manufacturing process for a given product is very stable from one week to a next. Updating the model each week when there are only small changes to the data will not have a large impact on prediction accuracy. Additionally, updating the model every week makes the system more complex for the engineers to understand, as they need to investigate what caused the change in the model from the previous week to the current week. Therefore, for each problem, we use a criteria to determine if a model needs to be updated given the newly observed data for the week. The generation of a new model

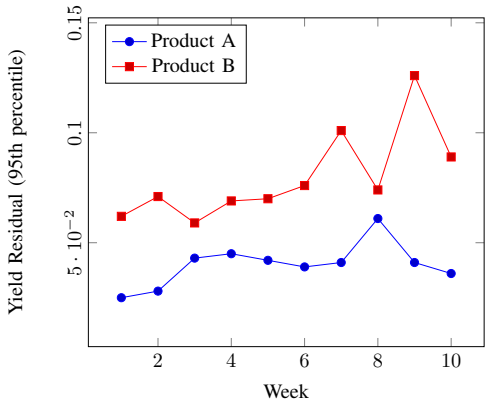

Figure 3. Yield residual results using boosted k-NN with $k \in \{1, 3, 5, 7, 9\}$ for weak hypothesis. The y-axis is the 95th percentile of the absolute difference between predicted and actual yield (residual) of wafers.

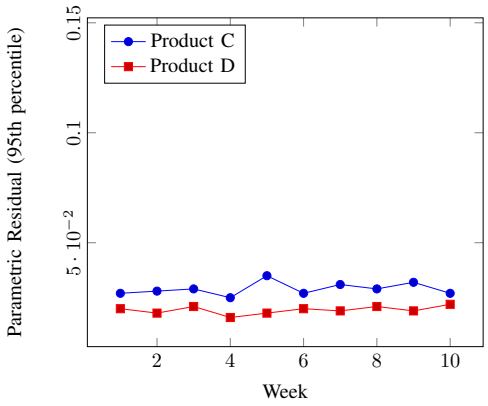

Figure 4. Parametric residual results using Random Forests. The y-axis is the 95th percentile of the absolute difference between predicted and actual voltage of each SoC.

signifies a reasonably large change in the test metrics as compared to the previous weeks. For *yield* prediction, we use the previously trained model to predict the new samples for the week. If the average residual i.e. the absolute difference between predicted and actual yield, for the new week is equal to or smaller than the average residual for the previous weeks, then we do not update the model. Otherwise, a new model is trained and used to predict the new week. For *parametric* prediction using Random Forests, we start by always training a model for the new week. Then, we measure the cosine similarity between the importance metric vectors of the new model and the previous model and chose the new model if they are less than 90% similar. Models that are more than 90% similar imply that there is no significant change, so the old model is kept. In practice, we observe that a model can be used for many weeks without the need for change and still make accurate predictions.

V. Model Validation

As stated earlier, we treat the machine learning models as a *gold standard* used to inform the manufacturing process, and therefore it is essential for the models to make accurate predictions for a majority of units. Figure 3 shows the 95^{th} percentile of residual for *yield*. For product A, the residuals are ≈ 0.05. For product B, which is a newer product than A, the residuals are slightly higher at ≈ 0.1. Generally, we observe that newer product have larger residuals due to more changes on a weekly basis in the manufacturing process. Figure 4 shows the 95^{th} percentile of residuals for *parametric* prediction. For both products, the residual is ≈ 0.02. The *parametric* predictions are considered quite accurate which is due to the larger number of test metrics used for the model. We see that the residuals for both the *yield* and *parametric* models are very stable across multiple product and over many weeks.

VI. Outlier Detection

The main benefit of an outlier detection system is that it dramatically reduces the number of wafers that must be analyzed manually. Wafers are labeled as outliers based on the model residuals for *yield* or *parametric*. Since *yield* is measured for an entire wafer, the residual is the absolute percentage difference between the model's predicted number of good units in the wafer and actual number of good units in the wafer. *Parametric* is measured for each SoC on a wafer individually. The model predicts the *parametric* of a component for each unit and then summarizes the error to the wafer level. To aggregate these residuals to a wafer level, we average the residuals for each unit on a wafer. Figure 5 shows the average residual for all the wafers for Product D at week 4. In general, the average residual for the wafers are similar to the average residual at the die level as shown in Figure 4. However, we observe that there are several wafers that have much higher than average residual, which would be labeled as outliers.

There are an abundant number of outlier detection techniques [5]. We use Density-based spatial clustering of applications with noise (DBSCAN) as it naturally marks samples in low density regions as outliers [6]. There are two tunable parameters for DBSCAN which must be set for the application. Eps is the minimum distance between two points for them to be considered in the same neighborhood (cluster). MinPts is the minimum number of points required in a neighborhood for it to be considered a cluster. Neighborhoods with fewer than MinPts will be marked as outliers. For instance, if we set Eps to 0.005 and MinPts to 5, then for Figure 4, DBSCAN would mark the two wafers beyond 0.45 average residual as outliers. The main benefit of using DBSCAN, as opposed to setting an outlier threshold (e.g. mark wafers with residuals > 0.40 as outliers), is that it is able to adjust to differences in the distribution of residuals over both products and weeks. For instance, setting

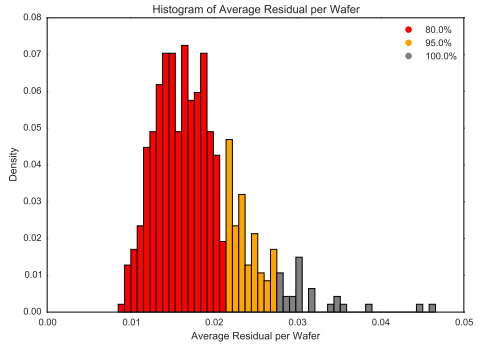

Figure 5. Histogram of average residual per wafer for all product D wafers at week 4. Red, orange and grey represent 80.0%, 95.0% and 100.0% of the population cumulatively.

a threshold of 0.40 may work well for product D, but would label many normal wafers as outliers for product C. Using DBSCAN allows for a more robust outlier detection system that works well over different products.

A. Outlier Diagnosis

Apart from labeling wafers as outliers each week, meta-information about the model and test metrics are provided to aid the engineers in the diagnose process. The meta-information includes the relevant test metrics (which are ranked by the model) and an indication of if the model was updated that week. For *yield* the small number of parameters means that they are all relevant each week. For *parametric*, the relevant parameters depend significantly on which component the model is being trained for. For instance, if a model is predicting the *parametric* of a graphical component on the unit, most of the important test metrics will be related to graphics.

For ease in diagnosis, we provide all trends and meta-information via a graphical user interface. Additionally, we provide in-depth views for each wafer so that the outliers can be examined at a more granular level. Figure 6 is a screenshot from the interface showing an arbitrary die grid. This is an illustrative example of an outlier group and does not contain any real data. While this is a mock example, it generally follows similar patterns of the real observed outliers for the various products. The colors represent the residuals and show if the model over- or under-predicted the actual value. To determine which test metrics are causing the large residual, a model is trained using just the units on the outlier wafer. For *parametric*, we take the difference of the important metric vectors of the model trained on the entire population and the model trained on just the outlier. After ordering the differences by magnitude, we show which test metrics have a much larger or smaller impact for the outlier as compared to the total population.

The wafers labeled as outliers due to large *yield* residual need domain experts to investigate further. This is due to the limited number of test metrics used to predict the *yield*. Other test metrics which were not used in the model are analyzed by experts to determine the root cause for the large residual.

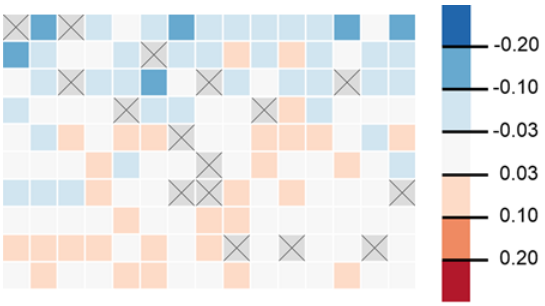

Figure 6. An arbitrary grid of SoC die. Blue tones indicate that the model under-predicted the actual value and red tones indicate over-prediction. The gray boxes with black Xs indicates units that did not have sufficient test metrics and were dropped before the performance metric stage.

VII. Conclusion

In this paper, we have applied advanced machine learning techniques to large scale manufacturing process at Intel to automatically identify systematic changes and detect outlier wafers. We show that machine learning can effectively model this manufacturing process and provide accurate predictions, leading to a significant reduction in the amount of data that must be inspected manually. Meta-information about each outlier is provided to aid in expediting the inspection process. A user interface provides a high level summary of the state of the manufacturing process which is updated on a weekly basis for each product. Ultimately, this automation facilitates faster detection and decision making with application to any type of manufacturing process.

References

[1] H.-C. Yu, K.-Y. Lin, and C.-F. Chien, "Hierarchical indices to detect equipment condition changes with high dimensional data for semiconductor manufacturing," *Journal of Intelligent Manufacturing*, vol. 25, no. 5, pp. 933–943, 2014.

[2] K. Y. Chan, C. Kwong, and T. C. Fogarty, "Modeling manufacturing processes using a genetic programming-based fuzzy regression with detection of outliers," *Information Sciences*, vol. 180, no. 4, pp. 506–518, 2010.

[3] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[4] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.