Abstract
Systematic defects due to design-process interactions are a dominant component of integrated circuit (IC) yield loss in nano-scaled technologies. Test structures do not adequately represent the product in terms of feature diversity and feature volume, and therefore are unable to identify all the systematic defects that affect the product. This paper describes a method that uses diagnosis to identify layout features that do not yield as expected. Specifically, clustering techniques are applied to layout snippets of diagnosis-implicated regions from (ideally) a statistically-significant number of IC failures for identifying feature commonalities. Experiments involving an industrial chip demonstrate the identification of possible systematic yield loss due to lithographic hotspots.

1. Introduction
The relentless scaling of CMOS devices continues to pose formidable challenges to the IC manufacturing community. As the IC manufacturing process becomes more and more complex, its interactions with design becomes much less predictable [1-2]. As a result, certain layout features are more difficult to manufacture than others and thus have an increased likelihood of failure. Unlike random defects caused by contaminations, defects due to design-process interactions are systematic in nature. In other words, they can lead to repeated IC failures wherever there are similar layout features. If these yield-limiting layout features can be identified and eliminated, then yield can be improved without any improvement in process control. Furthermore, additional test patterns can be generated to target these features to reduce test escape and improve the quality of the product if they instead pose a threat to quality [3]. Given the competitive nature of the semiconductor industry, rapid identification of key yield limiters and high quality control are essential to success.

Traditionally, test structures are used to debug and monitor the manufacturing process. Test structures are simple layout structures that are carefully designed and placed in a wafer to extract vital (defect) statistics of the process. Typical test structures include via chains, comb and serpentinite structures, and densely populated lines. To allow easy measurement of the parameter of interest, the test structure is deliberately kept simple but with the requirement that the structure is sensitive to the parameter of interest while being insensitive to everything else. When yield excursion occurs, test structures can provide timely feedback so that corrective action can be taken promptly. Unfortunately, also because of their simplicity, typical test structures are unsuitable for monitoring the design-process interactions in the presence of complex layout geometries, which are only found in large enough numbers in the product ICs themselves. In fact, in [1], two concrete examples of systematic defects in product ICs are shown to have escaped detection by test structures.

Due to the inherent limitations of test structures, volume diagnosis of test-fail data collected from product ICs (e.g., [4-10]) has gained increasing popularity in recent years as an alternative (and supplementary) process-monitoring and yield-learning vehicle. There are many advantages of this approach. First, diagnosis of the test-fail data consumes only CPU cycles, sacrifices no silicon area, and therefore does not incur substantial cost. Second, the product IC itself contains the diverse geometries and the needed volumes that are too expensive to be fully replicated in test structures that are relegated to small regions of the wafer. Finally, the IC manufacturing process is not static; it changes over time. It is therefore important to monitor these changes on the design in order to maintain high yield and quality. Thus, using on-going diagnosis to detect adverse effects is a cost-effective and non-intrusive way to maintain yield and quality.

The work in this paper is a step further in this direction. By mining volume diagnosis data, our goal is to identify layout features that have an increased likelihood of failure due to design-process interactions. Particularly, layout images of the diagnosis-implicated regions are extracted and clustered to find commonalities in the layout features. When a sufficiently large number of ICs are analyzed and clustered, any yield-limiting layout features can be identified and analyzed (e.g., through simulation, physical failure analysis, etc.) to verify the existence of a systematic defect.

1.1 Prior work
In the last several years, there has been a growing amount of research on identifying systematic defects through volume diagnosis. In [7-8], expected failure rates are computed for layout features that are assumed to be difficult to manufacture. This information is combined with volume diagnosis data to identify outliers (i.e., systematic defects). The authors in [9] use critical-area analysis [11] to compute the expected failure rate for each net and investigate the presence of systematic issues in nets where the empirical failure rate is not as expected. In

* This work is supported by the Semiconductor Research Corporation under contract no. 1644.001.
Our approach analyzes layout differences. First, the work in [15-16] relies on simulation, which is known to be inaccurate and too conservative, i.e., it is likely to identify features that may never cause an IC to fail. Fixing most (if not all) of the hotspots might result in over-correction which can adversely affect area and performance. Relying on simulation alone also means that no systematic defects are identified for failing mechanisms that are not modeled. In contrast, our approach uses diagnosis-implicated regions that are generated by diagnosing actual IC failures, and therefore does not suffer from these limitations. In addition, the work in [15-16] requires the entire design to be simulated, which is impractical especially when many different process corners are considered. Our approach, on the other hand, may apply process simulation (which is not required) to very small portions of the design for validation purposes only. Most importantly, our approach has the capability to provide failure rates for problematic layout features. In contrast, the identification of layout hotspots via process simulation (which likely does not account for RETs (resolution enhancement techniques), dummy fill, etc.) does not provide any information about the likelihood of a hotspot actually leading to failure.

The work in [15-16] also closely resembles the approach described here but again there are several important differences. First, the work in [15-16] relies on simulation, while Section 5 draws conclusions.

In this section, the details of the methodology are described. We begin by giving an overview which is then followed by detailed descriptions of each step in the methodology.

2. Systematic Defect Identification

Our method consists of four steps: (1) volume diagnosis, (2) layout snippet extraction, (3) snippet clustering, and (4) validation. Volume diagnosis is simply applying diagnosis to a sufficiently large number of IC failures. The outcome

Our method consists of four steps: (1) volume diagnosis, (2) layout snippet extraction, (3) snippet clustering, and (4) validation. Volume diagnosis is simply applying diagnosis to a sufficiently large number of IC failures. The outcome
of diagnosis is a set of candidates, where each candidate consists of the suspect net where the defect is believed to reside. A layout region (i.e., a region involving one or more layers in the layout that contains the features of interest, for example, a net and its surrounding nets) is extracted for each suspect net. We then partition the extracted layout region into one or more layout snippets. Each layout snippet is saved as a snippet image. Clustering, such as $K$-means [18], is then applied to group similar snippet images together in order to identify any commonalities. Finally, the identified layout feature is logically localized to a fault. A higher diagnostic resolution leads to an improved diagnostic resolution, 50% on approach that can distinguish logically-equivalent faults, and thereby, a diagnosis outcome consists of more than one candidate location. Diagnostic resolution [26] is a measure of the effectiveness of a diagnosis methodology to logically localize the fault. A higher diagnostic resolution is desired since there is less uncertainty concerning the failure location. Diagnostic resolution is often used as a metric for comparing different techniques and evaluating the merit of a particular diagnostic approach.

2.2 Diagnosis

Our goal is to identify yield-limiting layout features. One possible approach is to grid the entire layout into many small regions (i.e., layout snippets) and then perform clustering on the resulting snippet images. The result of clustering can then be overlaid with volume-diagnosis data to identify any systematic issues. This approach is possible but inefficient since many layout features will be easily manufacturable. In other words, for the aforementioned approach, computational resources are spent on clustering a large number of “healthy” snippets, which is not of interest in the work described here\(^1\). Therefore, to narrow down the scope of investigation and to save computing resources, clustering is limited to the layout regions implicated by diagnosis of failed ICs since these regions are likely to contain any systematic defects that may exist.

A variety of diagnosis methodologies have been proposed (e.g., [20-25]) and the question is which one is the most suitable for the work described here. A diagnosis methodology that can exactly pinpoint the $x$-$y$-$z$ location of a defect would be ideal. However, this requirement is unrealistic because diagnosis is not perfect and, more often than not, a diagnosis outcome consists of more than one candidate location. Diagnostic resolution [26] is a measure of the effectiveness of a diagnosis methodology to logically localize the fault. A higher diagnostic resolution is desired since there is less uncertainty concerning the failure location. Diagnostic resolution is often used as a metric for comparing different techniques and evaluating the merit of a particular diagnostic approach.

The diagnosis methodology adopted in this work is called DIAGNOSIX [21]. The reason for choosing this methodology is that it possesses a novel layout-based approach that can distinguish logically-equivalent faults, leading to an improved diagnostic resolution, 50% on average. High diagnostic resolution in this work means that fewer snippets are involved in clustering, resulting in a higher confidence in the resulting clusters.

It should be emphasized that although DIAGNOSIX [21] is used here, other methodologies (e.g., the critical-area based method [9]) can also be easily used instead.

\(^1\) Identification of easy-to-manufacture features however may be useful for forming a set of allowable patterns for DFM objectives.

---

**Figure 1:** Flow diagram of the layout analysis tool.

2.3 Layout Snippet Creation

An in-house layout analysis tool (LAT) is used to extract layout snippets of diagnosis-implicated regions [27]. LAT makes use of three open-source C++ libraries, namely, the OpenAccess Library (OA) [28], the Computational Geometry Algorithm Library (CGAL) [29], and the CAIRO Library [30]. The OA library performs design-format conversion and provides high-performance database operations. The CGAL library contains geometric algorithms suitable for layout analysis. The CAIRO library is used for image rendering and generation. Figure 1 illustrates the flow implemented by LAT. Each circuit net is retrieved from an annotated layout database to find its layout geometries, which are further processed to create snippet images.

2.3.1 Database Preparation

Since the typical description of design layout is GDS (Graphic Data System) [31], the GDS-to-OA translator provided in the OA package is used to convert GDS to the format of the OA database. In addition, since GDS typically consists of purely geometric information, the resulting OA database will not contain any logical connectivity information, or any net or cell names. Thus, the converted database is further processed to add this information. This is a one-time task whose cost is amortized over repeated use of the annotated database to extract snippet images for each failure of interest.

2.3.2 Net Geometry Processing

The input to LAT is a set of suspect nets that have been identified by diagnosing a failed IC. Since the database has been annotated with connectivity information, the polygons associated with a given net can be easily accessed. Clearly, different nets can have different wire segments that reside on various layers, thus making net comparison non-trivial. Figure 2 illustrates this diversity.
Figure 2a shows a net that is relatively long that spans two layers (highlighted in yellow), while Figure 2b shows a short net that resides in only one layer. If only a single snippet image is extracted for each net, then they would either have a very different scale (with a fixed image dimension) or have a very different image dimension (with a fixed scale). In either case, it is difficult to directly compare these two nets.

2.3.3 Center Line Extraction

The image scale, dimension, and alignment requirements has motivated the use of the straight-skeleton algorithm [32] provided by the CGAL library. A straight skeleton of a polygon is defined as “the union of the pieces of angular bisectors traced out by the polygon vertices” when the polygon is being shrunk [32]. Figure 4a illustrates the straight skeleton (shown in red and built using the CGAL library) of a polygon. A vertex that belongs to both the polygon and the straight skeleton is called a contour vertex, while a vertex that belongs only to the straight skeleton is called a skeleton vertex. For our purposes, however, we are only interested in the skeleton vertices since they alone define the center line of the polygon, as shown in Figure 4b.

2.3.4 Snippet Image Generation

By using the straight-skeleton algorithm, the center line of a polygon can be extracted. Different parts of a polygon can then be centered by splitting the center line into non-overlapping segments and constructing a box of fixed dimension with each split segment at the center. Snippet images are generated from these boxes. This is repeated for all polygons that make up the suspect net. The splitting process is illustrated in Figure 5. Figure 5a shows the suspect net highlighted with a yellow boundary in metal-1 (blue) and metal-2 (red). Figures 5b and 5c show the same net by metal layer, respectively. Figures 5b and 5c also show how each polygon in the suspect net is processed to extract the center line (in white). Each center line is split and used to create a series of boxes (with a green boundary). Each box defines a region for the snippet image. In this example, five snippet images are generated from the diagnosis-implicated region.
The CAIRO library is an open-source graphics library that is widely used for image rendering. The CAIRO library is platform-independent and is capable of producing an image in a variety of formats that include pdf, png, svg, etc. C++ code using the OA Library has also been written to support the GDS format as well. For clustering, the png format is used. Once the yield-limiting layout features are identified, the layout snippet is also saved in the GDS format for further analysis (e.g., lithography simulation).

2.4 Clustering

Clustering, a form of unsupervised learning [33], is a class of data exploratory techniques that partition objects into groups based on their similarity. Similar objects are ideally put in the same cluster while dissimilar objects are placed in different clusters. The goal of clustering is to discover structure in the data by exploring similarity within the data. A typical clustering task involves the following sub-tasks [34]:

1. representation of the object, which can involve feature extraction or selection,
2. definition of a measure of similarity between objects,
3. selection and application of a clustering algorithm.

Object representation is accomplished using the process described in the previous section. More specifically, the objects to be clustered are the center-aligned snippet images of fixed dimension and scale, each of which is represented as a matrix of pixels. There are a number of advantages of using this type of representation. First, it allows direct comparison of snippet images (e.g., two images can be compared on a pixel-by-pixel basis). Second, it is easy to include geometries from different layers in the same image by appropriately setting color transparency values (also called the alpha channel) of the layers involved. This is important when the goal is to identify a systematic defect caused by layout features that span multiple layers (e.g., a bridge induced by a chemical-mechanical polishing (CMP) [35], for example). Third, image processing techniques (such as the distance transform [36]) can be readily applied if desired.

For convenience of computation and simplicity of discussion, the pixel matrix is often converted to a vector, where the rows (or the columns) of the matrix are concatenated. The image can be easily reconstructed from the vector by undoing the concatenation. This convention is adopted in this paper.

Before a clustering technique can be applied, a similarity definition is required (i.e., sub-task 2). Suppose two images are represented as vectors \(X = \{x_1, x_2, \ldots, x_n\}\) and \(Y = \{y_1, y_2, \ldots, y_n\}\) in two-dimensional space, where \(x_i\) and \(y_i\) are component indices. Then, a common metric for comparing images is the cosine distance [37]:

\[
1 - \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}}
\]

Two images are similar if they have a very small cosine distance. Conversely, two images are dissimilar if they have a large cosine distance. It should be noted that cosine distance is chosen because of its popularity. Other distance functions, such as the Mahalanobis distance [38], can also be easily used and should be explored as well.

Having defined the object representation and the similarity metric, the third sub-task is to choose a suitable clustering algorithm. Data clustering is an extensively researched field and there are many clustering algorithms available. Generally, a clustering algorithm can be classified into two categories: 1) hierarchical and 2) partitional approaches [34]. Hierarchical approaches produce a nested series of data partitions while partitional algorithms produce a single, flat partition of the data. These differences are illustrated in Figure 6 [34].

![Figure 6: Illustration of data after (a) partitional clustering and (b) hierarchical clustering [34].](image)

Figure 6a shows data (i.e., A, B, ..., G) in two-dimensional space as well as example output from a partitional clustering algorithm with the requirement that the data be separated into three clusters. Figure 6b shows an example output of a hierarchical algorithm for the same data. The tree-like structure is called a dendrogram [34]. The vertical axis is the distance metric. The height in the dendrogram is proportional to distance metric. For example, B and C are closer in distance than D and E, which is reflected in the heights of the lines corresponding to B and C (before they are joined horizontally) and the heights of the lines corresponding to D and E (before they are joined horizontally). Figure 6b shows that the dendrogram specifies a nested series of partitions. Depending on the distance threshold, it produces different partitioning results. For example, the first threshold produces the partition \{\{A\}, \{B,C\}, \{D,E\}, \{F,G\}\}. The second threshold produces the partition \{\{A,B,C\}, \{D,E\}, \{F,G\}\}, which matches the output of the partitional algorithm in Figure 6a.

The clustering algorithm used in [12] is hierarchical in nature. Hierarchical approaches are more versatile but are more computationally expensive than partitional approaches. For clustering a large data set, a partitional approach has to be used. For our problem, the partitional approach is used because of the size of our data set. It should be noted that our data size is larger than the one in [12] since multiple images are generated for each diagnosis candidate.
The $K$-means algorithm [18] is one of the most popular partitional approaches because of its ease of implementation and high speed. Other partitional approaches include mixture resolving [39] and various graph-theoretic approaches [40-41]. Due to its simplicity, the $K$-means algorithm is used which we describe next.

Let $x_1, x_2, \ldots, x_n$ denote the $N$ data points to be clustered, $K$ be the desired number of clusters, $\mu_1, \mu_2, \ldots, \mu_K$ be the cluster centers, and $r_{ij}$ denote the potential assignment of the $i$-th data point to the $j$-th cluster where

$$r_{ij} = \begin{cases} 
1 & \text{if the } i\text{-th data point is assigned to the } j\text{-th cluster} \\
0 & \text{if otherwise}
\end{cases}$$

The objective of $K$-means is to minimize the following cost function [39]:

$$C = \sum_{i=1}^{N} \sum_{j=1}^{K} \left( r_{ij} \times \text{dist}(x_i, \mu_j) \right)$$

where $\text{dist}(x, \mu)$ is the distance function that measures similarity of data point $x$ and the cluster center $\mu$. Here we use the cosine distance defined earlier. From the expression of the cost function $C$, it is clear that the goal is to choose cluster centers (i.e., $\mu_j, 1 \leq j \leq K$) and data-point assignments (i.e., $r_{ij}$) such that the distance sums (from the data points to their assigned cluster centers) is minimized. This problem is NP-hard [42], so an approximate heuristic is used to find a solution. Since both the cluster centers and assignments are not fixed, the $K$-means algorithm uses a two-step approach to minimize the cost function $C$:

1. Assume cluster centers $\mu$ are fixed, optimize the cluster assignment of data point $x_i$. It is clear that assigning $x_i$ to its closest cluster center will minimize the cost function $C$, i.e., set $r_{ij} = 1$ if assignment of $x_i$ to cluster $j$ minimizes $\text{dist}(x_i, \mu_j)$. Otherwise, set $r_{ij} = 0$.

2. Assume the cluster assignments $r_{ij}$ are fixed, re-calculate the cluster centers as the centroid of its data points, i.e., $\mu_j = (\sum r_{ij} x_i)/\left(\sum r_{ij}\right)$. It can be shown that this choice of $\mu_j$ minimizes the cost function $C$ when $r_{ij}$’s are fixed.

Steps 1 and 2 are repeated until the cost function $C$ does not further reduce.

The algorithm begins by randomly assigning $K$ data points as the initial cluster centers. $K$-means, though efficient, is a heuristic and therefore can easily settle in a local minimum. Particularly, whether the returned result is a local minimum depends strongly on the initial choice of the cluster centers. As a result, in practice, the $K$-means algorithm is often executed several times with random initial cluster centers, with the final result being the one that results in the lowest cost.

The choice for $K$ determines the number of partitions. It is non-trivial to choose an optimal $K$. It should be noted that we cannot simply calculate the optimal cost $C_\text{opt}$ for each $K$ and choose the $K$ that gives the lowest $C_\text{opt}$. The reason is that the optimal $C_\text{opt}$ is zero when $K$ is chosen to be $N$ (i.e., the number of data points) which corresponds to the scenario where each data point is a cluster center. In general, $K$ is chosen empirically or by heuristics that penalize data fragmentation. For our problem, $K$ is chosen using a simple heuristic:

$$\min_K \quad K$$

subject to $r_{ij} \text{dist}(x_i, \mu_j) < T \quad \forall i, j$

In other words, we choose the smallest $K$ such that all the data-point-to-cluster-center distances are smaller than a predefined threshold $T$ (for some $1 > T > 0$). Intuitively, as long as each data point is reasonably close to its cluster center, increasing $K$ is halted to avoid fragmentation of the data. To improve efficiency, binary search is used to search for $K$. A hard limit is also used to restrict $K$ to a reasonably small value. Finding a better and more efficient heuristic for identifying the optimal value of $K$ is the focus of on-going work.

There are several options for employing $K$-means for the problem addressed in this work. The simplest approach is to create snippet images for each net of interest and provide the resulting images directly to the $K$-means algorithm. Unfortunately, this naïve approach unintentionally causes the layout features from long nets to adversely bias the cluster centers, regardless of the presence of the systematic defects. Figure 7 illustrates this situation. Specifically, Figure 7a shows five snippet images extracted from a long net. It is clear that snippet images $a_1$, $a_2$ and $a_3$ are identical. In other words, snippet images derived from the same net tend to be strongly correlated (if not identical) and have a small cosine distance between each other. This is especially the case for a long net. If these images (i.e., $\{a_1-a_5, b_1\}$) are provided as input to the $K$-means algorithm, then images $a_1$, $a_2$ and $a_3$ are very likely to form a cluster center, regardless of whether a systematic defect is present in $a_1$, $a_2$, and $a_3$. 

![Figure 7](image)
A novel approach described here is to (1) perform K-means on the snippet images for each net first, (2) select representative snippet images from each net, and (3) perform K-means again on the representative snippet images from all the nets. This approach identifies unique representative layout features from each net so that repeating/similar features from a long net will not bias the second clustering process. Using again the nets of Figure 7, only snippet images \{a3, a4, a5, b1\} will be used as input to the second pass of the K-means algorithm. In other words, a two-pass K-means algorithm is adopted for this work. The two-pass K-means algorithm is implemented in MATLAB [43].

### 2.5 Simulation Validation

After clustering, each snippet image is assigned a label to indicate its cluster. For example, suppose 100 snippet images are grouped into six clusters, implying that each snippet image will be labeled with an integer from 1 to 6. Snippet images in the same cluster exhibit some degree of similarity and can be manually inspected. However, a more effective approach is to choose a few representatives from each cluster for further analysis. This is achieved by sampling several snippet images closest to the cluster centers. In this paper, lithography simulation of the layout snippets is performed to identify any lithography-induced hotspots. Hotspot investigation is performed since sub-wavelength lithography is ubiquitously deployed in nanoscaled technologies and is a major source of systematic issues. The software tool Optissimo [19] is used to perform lithography simulation.

Our analysis does not have to be limited to lithography however. For example, CMP simulation (using the CMP Analyzer [44] for example) can be used to identify any yield-limiting hotspots induced by CMP. In fact, simulators for any yield-loss mechanism can be used here to investigate the existence of any systematic issues that may explain the large cluster of failures. Of course, each source of failure may require a different radius of influence for constructing the snippet images.

It should be noted that although we adopt a simulation-based approach to validate the existence of the systematic defects, other more comprehensive and conclusive methods can be applied including Physical Failure Analysis (PFA).

### 3. Experiment Results

An experiment was carried out using an industrial test chip from LSI to study the effectiveness of the methodology. The test chip, fabricated using 130nm technology, consists of 384 64-bit ALUs. Each ALU has approximately 3000 gates and is tested within a full-scan environment for ~100% stuck-at fault coverage with approximately 230 tests. In this study, 738 failing ICs are successfully diagnosed using DIAGNOSIX [21] resulting in 2,168 diagnosis candidates. Of the 738 failing ICs, 106 resulted in a single diagnosis candidate while the remaining had two or more diagnosis candidates. We focus on the 106 ICs that result in a single diagnosis candidate since there is less uncertainty in the defect locations for these ICs. In addition, in order to validate the methodology, two additional ICs with pre-existing PFA results are also included in this experiment. Both ICs have however multiple diagnosis candidates. Since the defect locations are known, only the diagnosis candidates that contain the actual defect locations are included. The diagnosis candidates of the PFA’ed failing ICs do not overlap with any of the 106 diagnosis candidates.

LAT (described in section 2.3) is applied to the 108 diagnosis candidates using a 2μm-by-2μm bounding box (Both the width and the height of the bounding box are more than 15x larger than the minimum feature size (0.13μm) and therefore should be sufficient for capturing all optical interactions [45]) for capturing each center line in each candidate. To keep analysis time reasonable, each snippet image is chosen to have resolution of 100-by-100 pixels. A total of 10,819 snippet images are generated for the 108 diagnosis candidates. Column 4 of Table 1 shows the number of extracted snippet images for each layer (column 1).

<table>
<thead>
<tr>
<th>Layer</th>
<th>(T_1)</th>
<th>(T_2)</th>
<th>No. images (original)</th>
<th>No. images (first pass)</th>
<th>No. clusters (K)</th>
<th>Optimal cost (C)</th>
<th>Cluster weight</th>
<th>(\Sigma r_j \times \text{dist}(x_t, \mu_j))</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>polysilicon</td>
<td>0.030</td>
<td>0.050</td>
<td>2057</td>
<td>642</td>
<td>33</td>
<td>9.72</td>
<td>1</td>
<td>81</td>
<td>19.45</td>
</tr>
<tr>
<td>metal-1</td>
<td>0.080</td>
<td>0.100</td>
<td>1645</td>
<td>457</td>
<td>46</td>
<td>13.45</td>
<td>2</td>
<td>34</td>
<td>9.93</td>
</tr>
<tr>
<td>metal-2</td>
<td>0.015</td>
<td>0.035</td>
<td>973</td>
<td>524</td>
<td>14</td>
<td>7.29</td>
<td>9</td>
<td>51</td>
<td>37.43</td>
</tr>
<tr>
<td>metal-3</td>
<td>0.002</td>
<td>0.002</td>
<td>1786</td>
<td>124</td>
<td>3</td>
<td>0.06</td>
<td>5</td>
<td>68</td>
<td>41.33</td>
</tr>
<tr>
<td>metal-4</td>
<td>0.040</td>
<td>0.050</td>
<td>2217</td>
<td>84</td>
<td>6</td>
<td>0.70</td>
<td>3</td>
<td>32</td>
<td>14.00</td>
</tr>
<tr>
<td>metal-5</td>
<td>0.004</td>
<td>0.007</td>
<td>332</td>
<td>79</td>
<td>11</td>
<td>0.18</td>
<td>0</td>
<td>21</td>
<td>7.18</td>
</tr>
<tr>
<td>metal-6</td>
<td>0.001</td>
<td>0.001</td>
<td>168</td>
<td>52</td>
<td>17</td>
<td>0.00</td>
<td>0</td>
<td>12</td>
<td>3.06</td>
</tr>
<tr>
<td>contact</td>
<td>0.003</td>
<td>0.003</td>
<td>569</td>
<td>131</td>
<td>17</td>
<td>0.11</td>
<td>1</td>
<td>29</td>
<td>7.71</td>
</tr>
<tr>
<td>via-12</td>
<td>0.003</td>
<td>0.003</td>
<td>370</td>
<td>111</td>
<td>3</td>
<td>0.06</td>
<td>6</td>
<td>98</td>
<td>37.00</td>
</tr>
<tr>
<td>via-23</td>
<td>0.002</td>
<td>0.002</td>
<td>395</td>
<td>107</td>
<td>1</td>
<td>0.00</td>
<td>107</td>
<td>107</td>
<td>107.00</td>
</tr>
<tr>
<td>via-34</td>
<td>0.002</td>
<td>0.002</td>
<td>213</td>
<td>71</td>
<td>5</td>
<td>0.01</td>
<td>2</td>
<td>60</td>
<td>14.20</td>
</tr>
<tr>
<td>via-45</td>
<td>0.001</td>
<td>0.001</td>
<td>69</td>
<td>35</td>
<td>1</td>
<td>0.00</td>
<td>35</td>
<td>35</td>
<td>35.00</td>
</tr>
<tr>
<td>via-56</td>
<td>0.001</td>
<td>0.001</td>
<td>25</td>
<td>19</td>
<td>3</td>
<td>0.00</td>
<td>1</td>
<td>17</td>
<td>6.33</td>
</tr>
</tbody>
</table>

Table 1: Summary of the two-pass K-means clustering outcome for each layer for 108 IC failures.

\(1 \) The number of tests varies for each ALU.
\(2 \) Our current work is investigating the impact of diagnostic resolution on snippet clustering. We fully expect there will be negligible impact on identifying dominant clusters since incorrect candidates from diagnosis are expected to inject layout snippets across all clusters equally.
The two-pass $K$-means clustering approach described earlier is applied to the snippet images for each layer separately since in this initial work we are investigating lithography issues which is a function of one layer only. $T_1$ (column 2) and $T_2$ (column 3) are the thresholds for choosing $K$ for the first-pass and second-pass $K$-means, respectively. $T_1$ and $T_2$ specify the desired cluster size and are empirically chosen. Column 5 of Table 1 shows the number of snippet images selected by the first-pass of $K$-means. After the execution of the second-pass, the number of clusters formed and the resulting cost $C$ are given in columns 6 and 7, respectively, of Table 1. In addition, cluster weight (i.e., the number of data points assigned to a cluster) is measured for all cluster centers and their minimum, maximum and average values are recorded in columns 8-10, respectively. Similarly, the minimum, maximum and average values for the summation of the data-point-to-cluster-center distances $\Sigma r_{ij} \times dist(x_i, \mu)$ are listed in columns 11-13, respectively. Cluster weight is a measure of the importance of a cluster (e.g., a cluster with 20 data points is not as important as a cluster with 200 data points), while the total data-point-to-cluster-center distances measures the distance size of a cluster. Column 14 shows the clustering run time in each layer computed on a machine with a 2.8GHZ and 16 GB of memory.

Table 1 reveals that the clustering outcome for the via layers consists of a small number of clusters. This is expected because most of the images contain a single via in the center of the image. Occasionally, neighboring vias are included in the image, and sometimes the difference is large enough to result in an extra partition. On the other hand, snippet images in the contact layer are partitioned into many more clusters even when larger values of $T_1$ and $T_2$ are employed. This too is expected since contacts are more densely placed than vias in general and therefore the same 2µm-by-2µm bounding box captures a more diverse set of images in the contact layer. A similar trend is observed for the poly and metal layers. In other words, a more densely routed layer results in more partitions since denser routes result in a more diverse set of layout features. The irregular geometries in the polysilicon layer and the metal-1 layer also contribute to more partitions. As a result, it is worth exploring different bounding-box sizes for each layer.

It is also informative to inspect the images in the same cluster as well as the images that are in different clusters. The clusters for metal-1 are chosen for illustration purposes since metal-1 contains a diverse set of geometries. Figure 8a shows four snippet images from a cluster in metal-1, while Figure 8b shows four snippet images from another metal-1 cluster. Figure 8 shows that geometries in the same cluster resemble each other but are not exactly the same while geometries in different clusters exhibit substantial differences. This example clearly demonstrates that the proposed methodology is able to group images with similar layout features together for further analysis.

![Image](326x645 to 422x740)

**Figure 8:** Illustration of clustered snippet images (a) from one cluster and (b) a second cluster.

The layout snippets that correspond to the snippet images are fed to Optissimo [19] for lithography simulation. Since the LSI test chip for this experiment is implemented in 130nm technology, the lithography parameters are chosen to coincide with that technology, as shown in Table 2. (A more detailed explanation of the parameter values chosen is given in [17].) For this part of the experiment, we again focus on the metal-1 layer only.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelength</td>
<td>248 nm</td>
</tr>
<tr>
<td>Numerical aperture</td>
<td>0.65</td>
</tr>
<tr>
<td>Outer coherence</td>
<td>0.8</td>
</tr>
<tr>
<td>Inner coherence</td>
<td>0.5 (metal), 0.3 (via)</td>
</tr>
<tr>
<td>Magnification</td>
<td>4.0</td>
</tr>
<tr>
<td>Resist threshold</td>
<td>0.3</td>
</tr>
<tr>
<td>Defocus</td>
<td>0</td>
</tr>
<tr>
<td>Layer amplitude</td>
<td>0</td>
</tr>
<tr>
<td>Layer phase</td>
<td>0</td>
</tr>
<tr>
<td>Background amplitude</td>
<td>1</td>
</tr>
<tr>
<td>Background phase</td>
<td>0</td>
</tr>
<tr>
<td>Corner rounding</td>
<td>0.06 µm</td>
</tr>
<tr>
<td>Optical range</td>
<td>1.5 µm</td>
</tr>
<tr>
<td>Long range density</td>
<td>0.8</td>
</tr>
<tr>
<td>Long range coefficient</td>
<td>0.04</td>
</tr>
<tr>
<td>Short range density</td>
<td>0.14</td>
</tr>
<tr>
<td>Short range coefficient</td>
<td>-0.07</td>
</tr>
</tbody>
</table>

**Table 2:** Process parameters used for lithography simulation.

Figure 9 shows an example of a lithography-induced hotspot (i.e., insufficient contact coverage) identified by the Optissimo simulation. The simulation result for the contact layer is also shown so that the hotspot can be revealed. There are altogether 20 hotspots identified in the metal-1 layer. These hotspots are likely to cause systematic defects.

To prevent future occurrence of these hotspots, DFM rules can be formulated. For example, the insufficient coverage of the contact in Figure 9 can be resolved by increasing the metal-1-to-contact enclosure requirement. If it is too expensive to enforce the new rule in the entire design, the new rule can be refined so that it is enforced only in the standard cells that contain the hotspot.
To validate the methodology, we examine the clustering outcome of two ICs that have PFA results. Both ICs have snippet images of their corresponding defect location (as identified by PFA) in large clusters. Specifically, the first PFA result is a bridge between a polysilicon gate and its active region. The snippet image corresponding to this defect belongs to the third largest cluster (with weight 40) in the polysilicon layer (the maximum cluster weight in the polysilicon layer is 81). Figure 10 shows the SEM image of the PFA result and its corresponding layout region.

The second PFA result is a bridge between two nets in metal-2, and its SEM image and layout region are shown in Figure 11. The snippet image corresponding to this defect belongs to a cluster with weight 29 in metal-2 (the maximum cluster weight in metal-2 is 51).

4. Discussion

Once the systematic defects are identified and understood, it is possible to formulate DFM rules that describe the yield-limiting layout features that result in the systematic defects. The discovered DFM rules can be implemented [46] (and enforced) by using geometric operations provided by all rule-checking programs (e.g., diva [47]) or by using the approach described in [48]. The follow-on analysis is an important step to complete the feedback loop in yield learning and is the focus of our current work.

As discussed above, the value of $K$, the number of clusters, is currently chosen using a simple heuristic that can be expensive to compute. We are exploring other techniques for finding a suitable and efficient heuristic for selecting $K$ however. In addition, the similarity metric is obviously problem-dependent. In other words, the cosine distance [18], although commonly used, may not be the best choice for the problem addressed. We intend to experiment with other distance functions to improve clustering results. Similarly, the $K$-means algorithm can be replaced with other unsupervised techniques.

Besides exploring/fine-tuning the clustering process, snippet extraction is another area that requires further investigation. First, the choice of snippet size, as already mentioned, will likely depend on the yield-loss mechanism. For lithography-related issues, a dimension of 2 µm-by-2 µm should be sufficient for capturing all optical interactions [45]. However, other yield-loss mechanisms (e.g., CMP-induced yield loss) might require a different snippet size and is the focus of our ongoing work. Second, layout features at the boundary of the snippets can be missed. We plan to address this issue by extracting overlapping snippets or by repeating the analysis with a different offset to capture the boundary features. Finally, snippet extraction involving multiple layers will also be explored.

Further analysis of the clustering outcome is also possible. This includes normalizing the clustering result with critical area of the extracted snippets as well as with the frequency of occurrence of similar snippets in the entire layout.

Finally, the experiment in this paper only uses 108 failing ICs (including the two PFA results) for validation. We plan to further validate the proposed methodology with more industrial data and with simulation data [49].

5. Conclusions

In this work, we described a comprehensive methodology for identifying yield-limiting layout features. It begins by processing volume diagnosis data to extract snippet images of the diagnosis-implicated layout regions. Clustering snippet images is performed to identify commonalities in the layout features captured. The layout snippets that correspond to representative snippet images from each cluster are also analyzed using lithography simulation. Experiment results have demonstrated that the methodology is effective in grouping snippet images with
similar features. Moreover, several lithographic hotspots have been identified within the dominant clusters. These hotspots are likely to cause systematic defects. Our approach resolves the missing link in many systematic-defect identification methodologies by providing an automatic method of discovering failure-causing layout features. Our approach can be used independently or integrated into existing systematic-defect identification methodologies. Integration into existing approaches can be easily achieved by using the proposed method as a post-processing step to automatically identify and extract layout features that are problematic.

6. Acknowledgements

The authors would like to thank Prof. Marios Savvides of Carnegie Mellon University for his advice on image processing techniques and the LSI corporation for providing design and fail data.

References

[27] www.s2i.org/openedta.s2i.org
[28] www.cgal.org
[29] www.cadgraphics.org