An effective approach for identifying defective critical fabrication path

Kamal Taha*

Abstract: Most defect signatures on wafers are caused by faulty tools. If these defects are not captured by in-line inspection tools or sampled during Defect Review-SEM, they are carried over multiple processing steps and discovered at the end of the fabrication procedure. Wafers with different defect signatures discovered at the end of the fabrication procedure are, most often, processed through same faulty tools. The fabrication paths of most defective wafers converge at some point to form a common sub-path before they disperse again. Usually, faulty tools in this common path cause most of the different defect signatures discovered at the end of the fabrication procedure. Process engineers would need to trace in reverse the fabrication paths to identify this common path to repair its tools. We introduce in this paper a defect diagnostic system called IDFP that can identify the common fabrication path that caused most of the defectivities on wafers discovered at the end of the fabrication procedure. We evaluated the quality of IDFP by experimentally comparing it with two systems. Results revealed marked improvement.

Subjects: Production Systems; Manufacturing & Processing; Engineering Productivity

ABOUT THE AUTHOR

Kamal Taha is an Associate Professor in the Department of Electrical and Computer Engineering at Khalifa University, UAE, since 2010. He received his Ph.D. in Computer Science from the University of Texas at Arlington, USA. He has over 80 refereed publications that have appeared in prestigious top ranked journals, conference proceedings, and book chapters. Over 20 of his publications have appeared in IEEE Transactions journals. He was as an Instructor of Computer Science at the University of Texas at Arlington, USA, from August 2008 to August 2010. He worked as Engineering Specialist for Seagate Technology, USA, from 1996 to 2005. His research interests span defect characterization of semiconductor wafers (under which the research reported in this paper falls), bioinformatics, Information Forensics & Security, information retrieval, data mining, and databases, with an emphasis on making data retrieval and exploration in emerging applications more effective, efficient, and robust.

PUBLIC INTEREST STATEMENT

Each wafer undergoes various fabrication equipment before it is fully functional. These wafers are highly vulnerable to defects caused by faulty equipment at the different fabrication processing steps. Each of these equipment is susceptible to malfunction that causes a specific type of defect signature on wafers. Specific malfunction in equipment can cause specific defect signature on wafers. That is, a defect signature on wafer can be caused by a specific type of equipment malfunction at a specific fabrication processing step. Some of these defects may not be captured by in-line inspection tools and are carried over multiple processing steps. They can be discovered at the end of the fabrication procedure, which requires process engineers to trace back the equipment that caused the defects. To overcome this, we introduce in this paper a defect diagnostic system called IDFP that can identify the common fabrication path that caused most of the defectivities on wafers discovered at the end of the fabrication procedure.
1. Introduction

Relying on engineers’ vision to judge defect patterns is unreliable and inconsistent, especially with the increasing sizes of wafers and the shrinking sizes of features. As alternatives, automatic defect detection (ADD) (Chou & Rao, 1997; Chen & Liu, 2000; Liu, Chen, & Lu, 2002; Zhou et al., 2002) and automatic defect classification (ADC) (Adly, Al Hussein et al., 2015; Adly, Yoo et al., 2015; Fenner et al., 2005) methods are gaining importance. The automatic detection is carried out by comparing an image of an inspected die to a reference defect-free die image (Chien, Wang, & Cheng, 2007; Palma, Nicolao, & Miraglia, 2005; Pukite & Berman, 1990; Yuan & Kuo, 2008; Xie & Simske, 2011). Despite the success of most of these methods in identifying defects, most of them have not been successful in associating the source of a defect signature to a specific processing step or tool. This is due, in part, to the fact that some defects may not be captured by in-line inspection tools or may not be sampled during Defect Review-SEM and are carried over multiple processing steps. This makes it difficult for these methods to pinpoint the root cause of the defects.

Most defect signatures on wafers are caused by faulty tools. A defect signature on wafer can be caused by a specific type of tool malfunction at a specific processing step. Some of these defects may not be captured by in-line inspection tools and are carried over multiple processing steps (Iwata et al., 2000). They can be discovered at the end of the fabrication procedure, which requires process engineers to trace back the tools that caused the defects.

Each wafer undergoes various fabrication equipment before it is fully functional. These wafers are highly vulnerable to defects caused by faulty equipment at the different fabrication processing steps. Each of these equipment is susceptible to a malfunction that causes a specific type of defect signature on wafers. Actually, most defect signatures on wafers are caused by faulty equipment. Specific malfunction in equipment can cause specific defect signature on wafers. That is, a defect signature on the wafer can be caused by a specific type of equipment malfunction at a specific fabrication processing step. Some of these defects may not be captured by in-line inspection tools and are carried over multiple processing steps. They can be discovered at the end of the fabrication procedure, which requires process engineers to trace back the equipment that caused the defects.

Associating the source of a defect signature to a specific processing step or equipment using ADD (Chen & Liu, 2000; Liu et al., 2002; Zhou et al., 2002), ADC (Adly, Al Hussein et al., 2015; Adly, Yoo et al., 2015; Taha, 2017), or engineers’ judgments can be ineffective and inconsistent. This is due, in part, to the fact that some defect signatures may not be captured by in-line inspection tools or may not be sampled during Defect Review-SEM and are carried over multiple processing steps, which makes it difficult for these methods to pinpoint the root causes of the defects. To overcome this, we introduce in this paper a defect diagnostic system called **IDFP “Identifying Defective Fabrication Path”** that can identify the common fabrication path that caused most of the defectivities on wafers discovered at the end of the fabrication procedure. It does so by identifying the set of all equipment that is likely caused all distinct set of defect signatures discovered at the end of the fabrication procedure.

Identifying all equipment that caused all distinct defect signatures discovered by a prober equipment at the end of the fabrication procedure is more effective than identifying each equipment caused a defect signature individually. This is due to the following: (1) multiple defect signatures can be caused by the same equipment, and (2) some defect signatures can be caused by a sequence of faulty equipment (i.e., a defect signature caused by some faulty equipment can be altered by a subsequent faulty equipment to a different defect signature).

The contributions of this paper are summarized as follows:

(1) Proposing a novel framework that identifies the common fabrication sub-path (i.e., common equipment) that caused most of the defectivities on wafers discovered at the end of the
fabrication procedure. It can identify this common fabrication sub-path by pinpointing the points of convergence and disperse of the fabrication paths that caused most defective wafers.

2. Literature review

Wafers usually pass through over 300 fabrication processing steps in the semiconductor manufacturing. Each of these steps comprises one or more equipment. That is, each wafer undergoes various fabrication equipment before it is fully functional. Specific malfunction in equipment can cause specific defect signature on wafers. Below are some of the common defects and the faulty equipment causing them:

- Bull eye and circular ring on wafers can be caused by either: (a) non-uniformities of thin film deposition caused by malfunction of the CVD, ECD, or PVD equipment, or (b) uneven temperature distribution caused by malfunction of the rapid thermal annealing equipment (Friedman, Hansen, & Nair, 1997).
- Edge ring defect on wafers can arise from etching problems caused by malfunction of the wet bench equipment (Burkeen et al., 2007).
- Zone defect signature on wafers often caused by malfunction of the thin film deposition equipment (Cunningham & Mackinnon, 1998).

For the sake of early identification of process problems, engineers need to detect the root causes of problems during the fabrication process to reduce the losses caused by excursion (Hansen et al., 1997). Towards this, many methods have been proposed for detecting spatial defect patterns on semiconductor wafers (de Berg et al., 2008). The authors of (Lee, Yu, & Park, 2001a, 2001b) proposed an unsupervised self-organizing map (SOM) algorithm using data sampling method. The proposed algorithm clusters the locations in the spatial chip that have similar defect features. The authors of (Chen & Liu, 2000; Liu et al., 2002) adopted adaptive resonance theory (ART) techniques to identify defect patterns on WBM. The authors of (Hsu & Chien, 2007) proposed a hybrid method to detect defective patterns. The method integrates ART network and spatial statistics. The authors of (Boly & Hajj, 2012) adopted SOM and ART as wafer classifiers using real data sets and extensive simulated and.

A number of methods proposed model-based clustering. The authors of (Chien, Hsu, & Chen, 2013) employed multi-way principal analysis and data mining techniques to monitor and diagnose the semiconductor fabrication process. These techniques are used to derive rules for fault classification and to detect faults. The authors of (Zhou et al., 2002) proposed a method for predicting yield using Critical Area Analysis. First, the method estimates the size of defects based on inline defect data. Then, it partitions yield loss according to the estimated defect sizes. The authors of (Yuan & Kuo, 2008) proposed a Bayesian model-based clustering models for clustering spatial defective pattern on semiconductor wafers. The models can determine ellipsoidal, curvilinear, and non-uniform global defect patterns. However, these algorithms cannot differentiate between the different types of defects.

The authors of (Taha, Salah, & Yoo., 2017) proposed a method for clustering the patterns of defective chips on wafers according to the spatial dependence of these defects across all wafer maps. The proposed method can identify the dominant defect patterns and clusters chip defects based on how dominant are their spatial patterns across all wafer maps. The authors of (Taha, 2017) proposed a method for detecting defect signatures on semiconductor wafers and allowing process engineers to trace back the source of a final defectivity identified at the end of the
fabrication procedure and associate it to specific defect signatures carried over multiple process zones. The author of (Taha, 2018) proposed a method based on the rules of inferences for diagnosing and identifying the root causes for semiconductor fabrication process. The authors of (Adly et al., 2015) proposed a method for discovering defect patterns using simplified subspaced and randomized general regressions. The authors cluster and partition defects using Voronoi-based data partitioning.

3. Methods

3.1. Outline of the approach

Let \( M \) be a non-empty set whose elements are themselves sets. Let each element of \( M \) be a set \( p \) of tools used in the fabrication of a defective wafer, whose defects were discovered at the end of the fabrication procedure. We observe that the set of defective tools that caused a distinct set of all defect signatures discovered at the end of the fabrication procedure is likely lie in the intersection of all the sets \( p \) of \( M \). That is, this set of defective tools is likely to be within a set \( S \), were \( S = \cap_{p \in M} p \). We call the fabrication path that passes through the set \( S \) the Critical Fabrication Path (CFP). This set is critical for identifying the root causes of most defect signatures discovered at the end of the fabrication procedure. IDFP employs an effective approach for identifying the CFP.

Below is an overview of the sequential computational steps taken by IDFP to identify the CFP:

1. **Representing the fabrication paths of defective wafers**: In the framework of IDFP, the fabrication paths of defective wafers are represented as a graph. We call such a graph the Fabrication Path Graph (FPG). The leaf nodes of FPG represent the end-of-line prober tools. These nodes are labelled with the defect signatures identified from the electrical failures captured by the prober tools. In the framework of IDFP, Wafer Defect Maps (WDMs) are created from either defects captured by in-line inspections stations or by reverse-engineering failure analysis techniques. We advocate the use of process engineers’ discretion in deciding the number of samples that justify performing failure analysis. Figure 1 shows an illustrative FPG.

2. **Converting a FPG into an equivalent tree**: IDFP transforms a FPG into an equivalent tree called the Fabrication Path Tree (FPT). We observe that the defective tools that caused a distinct set of all the defect signatures identified by failure analysis in response to electrical failures captured by end-of-line probers are likely to be located in the path that starts at the root of FPT and ends at the Lowest Common Ancestor (LCA) of the set \( N \) of leaf nodes labelled with the distinct set of defect signatures. This process is better illustrated in section 3.

3. **Identifying the CFP**: In the framework of IDFP, the LCA of the set \( N \) is identified using the concept of existence dependency. If the existence of the set \( N \) in FPT is dependent on the existence of a LCA \( lca \), the CFP starts from the root node and ends at the \( lca \). This process is better illustrated in section 4.

4. **Efficiently computing the CFP**: To efficiently compute the LCA of the set \( N \), IDFP employs a combination of stack-based sort-merge algorithm and the concept of existence dependency. This process is better illustrated in section 5.

3.2. Running example

We present a running example throughout the paper using the FPG shown in Figure 1. Each node in the figure represents a tool. Each node is identified by an alphabetical letter and a colour. All nodes representing tools of the same processing type (i.e., perform the same fabrication process) are assigned the same letter and colour. For example, all plating tools will be assigned the same letter and colour.
Figure 1. An illustrative FPG with few paths (in real-world, the number of paths is much larger). Each node represents a tool. Each node is identified by an alphabetical letter and a colour. All nodes representing tools of the same processing type (i.e., perform the same fabrication process) are assigned the same letter and colour. For example, there are two nodes with the same red colour and letter “e” to signify that they are of the same processing type.

Table 1. Abbreviations of the concepts proposed in the paper

| Abbreviation | Description                        | Abbreviation | Description                        |
|--------------|------------------------------------|--------------|------------------------------------|
| CFP          | Critical Fabrication Path          | DC           | Defect Context                     |
| FPG          | Fabrication Path Graph             | CDC          | Critical Defect Context            |
| FPT          | Fabrication Path Tree              | TYP(x)       | The fabrication processing type of equipment x |
| LCA          | Lowest Common Ancestor             | Dewey ID     | Dewey number-like label            |

Taha, Cogent Engineering (2019), 6: 1575636
https://doi.org/10.1080/23311916.2019.1575636
3.3. Converting a FPG into an equivalent tree

In the IDFP implementation, Dewey IDs are assigned to nodes for the efficient computation of: (1) the existence dependencies between nodes, and (2) the LCA of nodes. A Dewey ID is a Dewey number-like and it provides a straightforward solution to locating the LCA of nodes. Since Dewey numbers/IDs work on trees rather than graphs, IDFP converts a FPG into an equivalent tree. We call the resulting tree the FPT. A FPG and a FPT are equivalent, if each two nodes connected by an edge in the FPG are also connected by an edge in the FPT. Let \( P_w \) be the fabrication path of a defective wafer \( w \) in FPT. \( P_w \) starts from the root node and ends at a leaf node \( n \). The leaf node \( n \) will be labelled with the defect signature discovered on \( w \) by failure analysis in response to electrical failure captured by prober tools at the end of the fabrication procedure. Definition 1 formalizes the concept of FPT.

**Definition 1, FPT**

A FPT is a tree that is equivalent to a FPG. A FPT \( t \) is a tuple: \( t = (N, E, r, \lambda_t) \) where \( N \) is the set of nodes representing tools, \( E \subseteq N \times N \) is the set of edges representing the sequences of tools in the fabrication process, \( r \) is the root node of \( t \), and \( \lambda_t: n' \to \Sigma \) is a node-labelling function, where \( n' \) is a leaf node representing an end-of-line prober and \( \Sigma \) is an alphabet for labelling \( n' \) with the defect signature mapped to an electrical failure captured by the prober. Each node in \( t \) is labelled with a Dewey number-like label called a Dewey ID. If \( n' \) is labelled with a defect signature \( d \), the Dewey ID of \( n' \) reveals the fabrication path of the wafers with defect \( d \) (i.e., the sequence of the ancestor tools of \( n' \), which includes the tools that caused the defect \( d \)).

A Dewey ID of node \( n \) is a sequence of components, each having the form \( L_x \), where \( L \) is an alphabetical letter denotes a processing type, and the subscript digit \( x \) denotes the number of times that a tool of processing type \( L \) was used in the path from the root to \( n \) using DFS. The Dewey ID of a leaf node labelled with a defect \( d \) reveals the path that caused \( d \).

![Figure 2. The FPT equivalent to the FPG in Figure 1.](image)
Example 1: Figure 2 shows the FPT that is equivalent to the FPG in Figure 1. Each tool is identified by a Dewey component and a Dewey ID. The Dewey component uniquely identifies the tools. The Dewey ID uniquely identifies the sequential order of the tools in the fabrication procedure. The Dewey ID of a leaf node labelled with a defect signature d reveals the path that caused d. For example, the Dewey IDs of the leaf nodes $a_0.b_0.c_0.e_0.f_0.b_1.e_1$ and $a_0.c_1.d_1.e_3.f_1.g_3.h_1$ reveal two paths that caused the defect OE.

- The component $e_2$ reveals that tool $a_0.b_0.c_0.e_0.f_0.b_2.e_2$ is of processing type “e” and the subscript digit 2 reveals that two tools of processing type “e” were used prior to $a_0.b_0.c_0.e_0.f_0.b_2.e_2$ in the path from the root using Depth First Search.
- The Dewey ID of the parent of $a_0.b_0.c_0.e_0.f_0.b_2.e_2$ is $a_0.b_0.c_0.e_0.f_0.b_2$. The component $b_2$ reveals that the parent is of processing type “b” and that two tools of processing type “b” were used prior to the parent in the path from the root node using Depth First Search (DFS).

3.4. Identifying the critical fabrication path

Notation 1, Defect Context (DC)

A DC is a leaf node in FPT. It represents an end-of-line prober tool. It is labelled with a defect signature $d$ mapped to an electrical failure captured by the prober on a defective wafer $w$. The Dewey ID of the DC reveals the sequence of tools used in the fabrication of $w$, which includes the tools that may have caused the defect $d$.

Let $D$ be a set of all distinct defect signatures discovered on wafers at the end of the fabrication procedure during a specified fabrication period. Let $l_{ca}$ be the LCA of the DCs in FPT labelled with all the set $D$. A $CFP$ is the path in FPT that starts from the root node and ends at $l_{ca}$. We observe that the CFP is a common fabrication path of the wafers with all the defect signatures $D$. That is, a CFP is a sequence of tools used in the fabrication of the wafers with all the defect signatures $D$. We observe that it is highly likely that the faulty tools that caused all the defect signatures $D$ are located in the CFP. Thus, investigating the CFP is crucial for identifying the root causes of defect signatures discovered at the end of the fabrication procedure. To identify the CFP, IDFP computes the LCA of each set of DCs labelled with all the set $D$. We call each of these LCAs a Critical Defect Context (CDC). Thus, a CDC is a LCA of a set of DCs labelled with all distinct defect signatures discovered on wafers at the end of the fabrication procedure during a specified fabrication period.

Investigating all tools to identify the root causes of defect signature is time-consuming. It can waste the time of process engineers and distract them from investigating the influential tools in the fabrication process that are likely to have contributed to defect signatures discovered at the end of the fabrication procedure. IDFP overcomes this problem by adopting the concepts of CDC and CFP. These concepts allow IDFP to refine the set of tools used in the fabrication of defective wafers to investigate only the influential ones in the deficiencies discovered at the end of the fabrication procedure. That is, the CDC and CFP concepts allow process engineers to pay more attention to ongoing tools malfunctions that are likely to have contributed to defect signatures discovered at the end of the fabrication procedure. Most of these tools are usually confined between the root tools and the CDC, inclusive.

IDFP employs the concept of existence dependency to identify the CDC of a given set of DCs labelled with all distinct defect signatures discovered at the end of the fabrication procedure. The concept of existence dependency was first proposed for Entity-Relationship modelling (Elmasri & Navathe, 2007). An object $x$ is existence-dependent on an object $y$ if the existence of $x$ is dependent on the existence of $y$ (Widjaya, Taniar, & Rohayu, 2003). Intuitively, in order for a CDC to be valid, the existence of the DCs in FPT should be dependent on its existence. According to the existence dependency concept, for two nodes $n$ and $m$ to have an existence dependency relationship, $m$ and $n$ should belong to different types (Snoek & Dedene, 1998).
Snoeck et al. (Snoeck & Dedene, 1998) confirmed that two objects of the same type can never be existence-dependent on each other. The authors demonstrated that an object type in a graph is never existence-dependent on itself. They transformed an OO schema into a graph consisting of the object types found in the schema and their relations. The object types in the graph are related only through associations that express existence dependency. The authors demonstrated that if the two objects $O_i$ and $O_j$ belong to the same type, $O_i$ cannot be dependent on $O_j$ and vice versa.

IDFP applies the above-mentioned observations to determine whether a CDC of a set of DCs is valid. Recall that a CFP starts from the root node and ends at a valid CDC. Based on the above observations, in order for a CDC to be valid, the existence of the DCs in FPT should be dependent on the existence of the CDC. And, in order for the existence of the DCs to be dependent on the existence of the CDC, the processing type of the CDC should be different than those of the DCs. Otherwise, the CDC is invalid and cannot be used for identifying the CFP.

We describe below the scenarios when a CDC and a DC can be of the same processing type. During wafer fabrication procedure, some fabrication operations of the same line are repeated several times in intervening processing steps. For example, the fabrication operations of depositing different thin layers of metals on the wafer are performed several times in intervening processing steps. These fabrication operations of the same line are performed by tools of the same processing type. For example, the fabrication operations of depositing different thin layers of metals can be performed by tools of the same processing type, such as CVD, ECD, and/or PVD. Therefore, if a CDC and a DC happen to be performing fabrication operations of the same line, the two tools are of the same processing type.

We now demonstrate via an example that a DC cannot be existence-dependent on a CDC, if the DC is of the same processing type as the CDC. Consider the FPT in Figure 2. We are going to demonstrate that tool $a_0.b_0.c_0.e_0.f_0.b_1.e_1$ cannot be existence-dependent on tool $a_0.b_0.c_0.e_0$, because both are of the same processing type. Since $a_0.b_0.c_0.e_0.f_0.b_1.e_1$ and $a_0.b_0.c_0.e_0$ are of the same processing type, if $a_0.b_0.c_0.e_0$ does not exist, $a_0.b_0.c_0.e_0.f_0.b_1.e_1$ can perform the fabrication operation of $a_0.b_0.c_0.e_0$. Therefore, the fabrication path from $a_0.b_0.c_0.e_0$ to $a_0.b_0.c_0.e_0.f_0.b_1.e_1$ will not be impacted by the removal of $a_0.b_0.c_0.e_0$. On the other hand, we are going to demonstrate that tool $a_0.b_0.c_0.e_0.d_0.g_0.h_0$ is existence-dependent on tool $a_0.b_0.c_0.e_0$, because the two are of different processing types. If $a_0.b_0.c_0.e_0$ does not exist, $a_0.b_0.c_0.e_0.d_0.g_0.h_0$ will not be able to carry out the fabrication operation it supposed to perform, because: (1) there is no other tools in the path from $a_0.b_0.c_0.e_0$ to $a_0.b_0.c_0.e_0.d_0.g_0.h_0$ that can perform the same fabrication operation of $a_0.b_0.c_0.e_0$, and (2) the fabrication operation performed by $a_0.b_0.c_0.e_0.d_0.g_0.h_0$ cannot be done before the fabrication operation performed by $a_0.b_0.c_0.e_0$ is done. We formalize the concept of CDC in Definition 2.

Notation 2, \( \text{TYP}(x) \)
\( \text{TYP}(x) \) denotes the fabrication processing type of tool \( x \) (e.g., etching, plating, etc.).

Definition 2, CDC properties
CDC properties are the constraints that permit a LCA \( \text{lca} \) to be a CDC. These constraints are as follows: (1) \( \text{lca} \) is the LCA of a set \( S \) of DCs, \( S = \{ \text{DC} \mid \text{TYP}(\text{DC}) \neq \text{TYP}(\text{CDC}) \} \), and (2) \( |S| \) equals all distinct set of defect signatures discovered on wafers at the end of the fabrication procedure during a specified period.

Example 2: Consider the FPT in Figure 2. The distinct set of defect signatures is (OE, LRA, DM). Let us identify the CFP after determining which of the following two LCAs is a CDC by applying the CDC properties:

- \( a_0.b_0.c_0.e_0 \): It is a LCA for the following set of DCs, which are labelled with all the distinct set of defect signatures \( (a_0.b_0.c_0.e_0.f_0.b_1.e_1, a_0.b_0.c_0.e_0.f_0.b_2.e_2, a_0.b_0.c_0.e_0.d_0.g_0.h_0) \). However, this LCA...
does not satisfy the CDC properties, because: $\text{TYP}(a_0,b_0,c_0,e_0,f_0,b_1,e_1) = \text{TYP}(a_0,b_0,c_0,e_0)$ and $\text{TYP}(a_0,b_0,c_0,e_0,f_0,b_2,e_2) = \text{TYP}(a_0,b_0,c_0,e_0)$. Therefore, the LCA $a_0,b_0,c_0,e_0$ is not a CDC.

- $a_0,c_1,d_1,e_3$: It is a LCA for the following set of DCs, which are labelled with all the distinct set of defect signatures $(a_0,c_1,d_1,e_3,f_1,g_1,h_1, a_0,c_1,d_1,e_3,f_2,b_4,h_2, a_0,c_1,d_1,e_3,f_2,g_2,h_3)$. This LCA satisfies the CDC properties, because: $\text{TYP}(a_0,c_1,d_1,e_3,f_1,g_1,h_1) = \text{TYP}(a_0,c_1,d_1,e_3,f_1,g_1,h_1)$ $\neq \text{TYP}(a_0,c_1,d_1,e_3,f_2,b_4,h_2)$ $\neq \text{TYP}(a_0,c_1,d_1,e_3,f_2,g_2,h_3) \neq \text{TYP}(a_0,c_1,d_1,e_3)$. Therefore, $a_0,c_1,d_1,e_3$ is a CDC. Thus, the following is the CFP:

$$a_0 \rightarrow a_0,c_1 \rightarrow a_0,c_1,d_1 \rightarrow a_0,c_1,d_1,e_3$$

### 3.5. Efficiently computing the critical fabrication path

We constructed an algorithm called FindCDC (see Figure 3) that can efficiently compute the CFP by efficiently identifying the CDC of a given set of DCs labelled with all distinct defect signatures discovered on wafers at the end of the fabrication procedure. To identify the CDC, algorithm FindCDC employs a combination of stack-based sort-merge approach and the concept of existence dependency described in section 4. The algorithm employs the concept of existence dependency by applying the CDC properties described in section 4. Thus, the algorithm employs the combination of stack-based sort-merge approach and the CDC properties. The input to the algorithm is two arrays, called contexts and defects. Array contexts contains the Dewey IDs of the DCs. Array defects contains all distinct defect signatures discovered on wafers at the end of the fabrication procedure during a specified period of time. Thus, array defects is of length $m$, where $m$ is the number of distinct discovered defect signatures.

Each iteration of the algorithm produces a new stack state. The entry to the stack is the pair (Dewey component and defects[i]). Let $c_s, c_p, ..., c_k$ be the Dewey components in the stack from the bottom entry to the stack entry. The stack entry represents a node in FPT, whose Dewey ID is $c_s,c_p,...,c_k$. The bottom entry represents the root node in FPT, whose Dewey ID is $c_0$. Consider the FPT in Figure 2 and the stack in Figure 8-a. The stack entry represents node $a_0,b_0,c_0,e_0,f_0,b_1,e_1$ in the FPT, and the bottom entry represents the root node $a_0$.

Let $d_1, d_2, ..., d_m$ be the distinct discovered defect signatures. defects[i] represents defect signature $d_i$. If a DC $d_j$ is labelled with defect signature $d_i$, this is represented in the stack by storing the last Dewey component of $d_j$ in array defects[i]. For example, consider the FPT in Figure 2 and the stack in Figure 8-b, which represents the DC $a_0,b_0,c_0,e_0,f_0,b_2,e_2$. This DC is labelled with the defect signature “LRA”. This defect signature is represented in the stack by storing the last Dewey component of $a_0,b_0,c_0,e_0,f_0,b_2,e_2$ (i.e., the component $e_2$) in the second field of array defects (i.e., defects(Chien et al., 2013)).

In each iteration of the algorithm, the current DC being processed is called Current Context (CC) (line 2), and the prior DC processed is called Prior Context (PC). Line 3 computes the number of common Dewey components prefixes “q” in the CC and PC. For example, the CC in Figure 8-b is $a_0,b_0,c_0,e_0,f_0,b_2,e_2$ and the PC in Figure 8-a is $a_0,b_0,c_0,e_0,f_0,b_1,e_1$; therefore, $q = 5$. stack[q] entry represents the Lowest Common Ancestor (LCA) of the CC and PC. If $q$ equals the stack size (line 4),

---

**Figure 3. Algorithm FindCDC.**

```
FindCDC {
1. for (current = 1 to contexts.length) {
2. CC = contexts[current]
3. find the largest q such that stack[i] = CC[i], 1 ≤ i ≤ q
4. if (stack.length = q) //then the CC is a descendant of the PC
5. PushNode(CC, q, stack)
6. else ApplyCDCproperty(CC, q, stack)
7. } //end for
8. PopNode(q, stack)
} //end the algorithm
```
the CC is a descendant of the PC, and subroutine PushNode is called (line 5); otherwise, subroutine ApplyCDCproperty is called (line 6). Let $h$ be the maximum depth of FPT. The algorithm’s time complexity is $O(h |\text{DC}|)$.

Subroutine PushNode (Figure 4) pushes into the stack the top components of the CC that do not match the top components of the PC. By applying the CDC properties described in section 4, subroutine ApplyCDCproperty (Figure 5) first finds in FPT the closest descendant node $n$ of the LCA of the CC and PC, where $n$ and LCA are tools of the same processing type. It then pops from the stack the entry representing $n$ and all entries above it. It also pops these entries’ defect information (to satisfy the CFP properties). Subroutine popNode (Figures 6 and 7) pops the remaining entries above stack[q], which represent tools of different processing types than the LCA. Lines 7–9 pass the defect information of the popped entries to the current top entry. Lines 3 and 4: if all the fields of array defects of a popped entry are occupied, the entry represents the CDC.

Example 3: Using the FPT in Figure 2, we describe how algorithm FindCDC identifies the CDC for each combination of CDs labelled with the three distinct defect signatures OE, LRA, and DM. The following are the Dewey IDs of the seven DCs shown in Figure 2, which will be stored in array contexts:

- $[a_0,b_0,c_0,e_0,f_0,b_1,e_1]$,
- $[a_0,b_0,c_0,e_0,f_0,b_2,e_2]$,
- $[a_0,b_0,c_0,e_0,d_0,g_0,h_0]$,
- $[a_0,c_1,d_1,e_3,f_1,b_3,e_4]$,
- $[a_0,c_1,d_1,e_3,f_1,g_1,h_1]$,
- $[a_0,c_1,d_1,e_3,f_2,b_4,h_2]$,
- $[a_0,c_1,d_1,e_3,f_2,g_2,h_3]$. The stack is initially empty. Below are descriptions of the stack states shown in Figure 8.
Figure 8. The states of the stack of example 3, which show how algorithm \textit{FindCDC} identifies the CDC for each combination of CDs labelled with the three distinct defect signatures OE, LRA, and DM in the FPT shown in Figure 2.

\begin{itemize}
  \item \textbf{First stack state (Figure 8-a)}
  Line 2 of \textit{FindCDC}: the CC is $a_0,b_0,c_0,e_0,f_0,b_1,e_1$. The CC is labelled with the defect “OE”, which is represented in the stack by storing the last Dewey component of CC (i.e., component $e_1$) in “stack[7].defects[1]”

  \item \textbf{Second stack state (Figure 8-b)}
  Line 2 of \textit{FindCDC}: the CC is $a_0,b_0,c_0,e_0,f_0,b_2,e_2$. Line 3: \(q = 5\). Line 6 calls Subroutine \textit{ApplyCDCproperty}, which in turn calls subroutine \textit{PopNode} (line 5), which proceeds as follows. Line 2 pops the two entries above stack[$q = 5$] shown in Figure 8-a and lines 7-9 pass the defect information of stack[7] (i.e., the component $e_1$) to stack[$q = 5$].defects[1]. Line 6 of \textit{ApplyCDCproperty} calls subroutine \textit{PushNode}. As shown in Figure 8-b, line 2 of \textit{PushNode} pushes the non-matching components of the CC (i.e., the components $b_2$ and $e_2$) into the stack. Since the
CC is labelled with the defect “LRA”, line 3 of PushNode represents this in the stack by storing the last Dewey component of the CC (i.e., the component e₃) in “stack[7].defects[2]”.

Third stack state (Figure 8-c)
The CC is a₀.b₀.c₀.e₀.d₀.g₀.h₀. Line 3: q = 4. Line 6 calls ApplyCDCproperty, which proceeds as follows. Line 2: since TYP(stack[7] = TYP(stack[4]) (see Figure 8-b), line 4 pops stack[7] and stack [7].defects[2]. Line 5 calls PopNode, which proceeds as follows. Line 2 pops stack[6] and stack[5] (see Figure 8-b). Line 8: since TYP(stackEntry[5].defects[1]) = TYP(stack[5] = 4), the defect component e₁ will not be passed to stack[q = 4].defects[1]. Line 6 of ApplyCDCproperty calls PushNode, which proceeds as follows. Line 2: the non-matching components h₀, g₀, and d₀ of CC are pushed into the stack. Line 3: the CC is labelled with the defect signature “DM”, which is represented in the stack by storing the last Dewey component of the CC (i.e., the Dewey component h₀) in “stack [7].defects[3]”.

Fourth stack state (Figure 8-d)
The CC is a₀.c₁.d₁.e₁.f₁.b₁.e₄. Line 3: q = 1. Line 6 of FindCDC calls ApplyCDCproperty, which calls PopNode (line 5), which proceeds as follows. Line 2 pops the entries above stack[1] shown in Figure 8-c and lines 7–9 pass the defect component h₀ to stack[q = 1].defects[3]. Line 6 of ApplyCDCproperty calls PushNode, which pushes the non-matching components of the CC into the stack and represents the defect “DM” of the CC by storing the last component of CC (i.e., component e₄) in “stack[7].defects[3]”.

Fifth stack state (Figure 8-e)
The CC is a₀.c₁.d₁.e₁.f₁.g₁.h₁. Line 3: q = 5. Line 6 calls ApplyCDCproperty, which calls PopNode (line 5), which proceeds as follows. Line 2 pops the entries above stack[q = 5] shown in Figure 8-d and lines 7–9 pass the defect component e₁ to stack[q = 5].defects[3]. Line 6 of ApplyCDCproperty calls PushNode, which pushes the non-matching components of the CC (i.e., h₁ and g₁) into the stack and represents the defect “OE” of the CC by storing the last component of the CC (i.e., h₁) in “stack[7].defects[1]”.

Sixth stack state (Figure 8-f)
The CC is a₀.c₁.d₁.e₁.f₂.b₂.h₂. Line 3: q = 4. Line 6 calls ApplyCDCproperty, which calls PopNode (line 5), which proceeds as follows. Line 2 pops the entries above stack[q = 4] and lines 7–9 pass the defect component h₁ to stack[q = 4].defects[1]. Line 8: since TYP(stackEntry[5].defects[3]) = TYP(stack[q = 4]), the defect component e₃ will not be passed to stack[q = 4].defects[3]. Line 6 of ApplyCDCproperty calls PushNode, which pushes the non-matching components of CC (i.e., h₂, b₄, f₂) into the stack and represents the defect “DM” of the CC by storing the last component of the CC (i.e., h₂) in “stack[7].defects[3]”.

Seventh stack state (Figure 8-g)
The CC a₀.c₁.d₁.e₁.f₂.g₂.h₃. Line 3: q = 5. Line 6 calls ApplyCDCproperty, which calls PopNode (line 5), which proceeds as follows. Line 2 pops the entries above stack[q = 5] and lines 7–9 pass the defect component h₂ to stack[q = 5].defects[3]. Line 6 of ApplyCDCproperty calls PushNode, which pushes the non-matching components of the CC (i.e., h₃ and g₂) into the stack and represents the defect “LRA” of the CC by storing the last component of the CC (i.e., h₃) in “stack[7].defects[2]”.

Eights stack state (Figure 8-h)
Line 8 of FindCDC calls PopNode, which proceeds as follows. Line 2 pops the entries above stack[4] and lines 7–9 pass the defect components h₃ and h₂ to stack[4].defects[2] and stack[4].defects[3] respectively. Line 3 of PopNode will return true, since all the fields of array defects at stack[4] are occupied. Line 4 will output the stack entry a₀.c₁.d₁.e₃ as the CDC.
4. Experimental results

We implemented IDFP in Java, run on Intel(R) Core(TM) i7 processor, with a CPU of 2.70 GHz and 16 GB of RAM, under Windows 10. We evaluated IDFP by comparing it experimentally with the following two approaches:

- Chien et al. (2007): This approach employs decision tree analysis to identify faulty tools and process stages. The approach assigns each tool and process stage a P-value. It applies K-W tests to identify the tools with significant performance variation. Tools and process stages with low P-values are used as input for the decision tree analysis. The P-value threshold proposed in (Chien et al., 2007) is 0.3. The approach employs ANOVA F-test as the splitting criterion that identifies the tools with low yields. In the evaluations, we considered these low yields tools as the root causes of defects. We refer to this approach by Chien for easy reference.

- SDCA (Taha, 2017): We proposed previously in (Taha, 2017) a system called SDCA that identifies the tools that caused defect signatures on wafers. SDCA assigns a severity score for each defect signature at each processing step. The severity score of a defect signature \(d_j\) at a processing step \(S_i\) is computed based on the maximum Contiguity Ratio (Adly et al., 2015) of \(d_j\) at \(S_i\). Then, SDCA assigns a severity score for each processing step and tool based on the severity scores of defect signatures. The score of a tool reflects the relative severities of all defect signatures that caused by this tool.

The dataset used in the experiments is real-world wafer maps from Samsung Electronics in Korea. We used the same dataset in our previous work (Adly et al., 2015). The dataset includes the following: (1) fabrication records of 26 lots, (2) 843 wafer maps captured during the fabrication of the 26 lots at various processing steps, (3) information about the tools processed the 843 wafers during the fabrication procedure, and (4) the types of defect signatures captured on the wafers by in-line inspection tools during the fabrication procedure. We employed the software SAS, STATISTICS, and Scenario for data analysis and transformation and graph visualization. We constructed a FPG that represents the dataset by marking the fabrication paths (i.e., a sequence of tools) of the 843 wafers. IDFP transformed the FPG into an equivalent FPT. Table 2 shows the notations of the defect types in the evaluation dataset, which we used in the legends of the figures that plot the experimental results.

### Table 2. Notations of the defect types in the evaluation dataset, which we used in the legends of the figures that plot the results

| Defect type                  | Abbreviation | Defect type                  | Abbreviation |
|------------------------------|--------------|------------------------------|--------------|
| Overetch                     | OE           | Bridging                     | BR           |
| Underetch                    | UE           | Overexposure                 | OEX          |
| Corrosive residue            | CR           | Poor etching uniformity      | PEU          |
| Heavy metal contamination    | HMC          | Misalignment                 | MA           |
| Nonuniform etching           | NE           | Delamination                 | DM           |
| Loss of resist adhesion      | LRA          | Excessive etching            | EE           |
| Particulate contamination    | PC           | Photoresist pattern          | PPD          |
|                              |              | discontinuity                |              |
| Stringers                    | ST           | Missing pattern              | MP           |
| Wrong etch rate              | WER          |                              |              |

4.1. Evaluating the prediction accuracy in terms of precision

We divided the fabrication period of the 26 lots into 10 equal periods. For each period, we measured the precision of each model for identifying the tools that caused defect signatures during this period. Figure 9 shows the results. We computed the Precision using the standard formula shown in Equation (1).
4.2. Evaluating the prediction accuracy in terms of the percentage of hit rate

We evaluated the effectiveness of the three models for identifying the faulty tools that contributed to the defect signatures on the 843 testing wafers in terms of percentage of hit. For each defect signature $d_j$ on the testing wafers, we computed the percentage of hit achieved by each model for identifying the faulty tools that caused $d_j$. Figure 10 shows the percentages of hit.

4.3. Evaluating the prediction accuracy in terms of the closeness between the predicted and actual faulty tools in their steps in the fabrication sequence

Each wafer undergoes various fabrication processing steps in the fabrication sequence. Each of these processing steps is performed by one or more tools. Thus, each wafer undergoes various tools in a sequential order. Let $d$ be a defect signature on some of the testing wafers. Let $e_i(d)$ be the actual tool that caused the defect $d$. Consider that a model $m$ predicted tool $e_j(d)$ as the cause of the defect. The precision of each model for identifying the tools that caused defect signatures during each of the 10 periods of fabricating the 26 lots is shown in Figure 9.
In this section, we evaluate the prediction accuracy of model \( m \) by measuring the distance in the fabrication sequence between the steps of \( e_i(d) \) and \( e_j(d) \). The distance between \( e_i(d) \) and \( e_j(d) \) is the number of processing steps between the steps performed by \( e_i(d) \) and \( e_j(d) \) in the fabrication sequence. Intuitively, the smaller the distance, the more accurate the prediction is. A zero distance represents an exact match (i.e., a hit). For all defect signatures of type \( d \), let \( dis_m(d) \) be the average distance of model \( m \) with regard to \( d \), which is the average distance between the steps performed by \( e_i(d) \) and \( e_j(d) \) in the fabrication sequence. We use Equation (2) to measure \( dis_m(d) \).

\[
dis_m(d) = \frac{\sum_{i=1}^{d} |e_i(d) - e_j(d)|}{|d|}
\]

5. Discussion

As Figure 10 shows, IDFP outperformed the other two models in terms of percentage of hit in identifying the exact faulty equipment that caused defect signatures. The overall average percentages of hits achieved by the models were 53%, 44%, and 34% for IDFP, SDCA (Taha, 2017), and Chien (Chien et al., 2007) respectively. Based on our observation of the experimental results, we attribute the outperformance of IDFP over SDCA and Chien’s methods to the following: (1) the strengths of IDFP predictive techniques (i.e., its good predictive capabilities in associating defect signatures to faulty equipment), and (2) the limitations of Chien and SDCA’s methods. In subsection 5-1 we outline the strengths of IDFP’s method that have been observed from the experimental results. In subsections 5.2 and 5.3, we outline the limitations of Chien and SDCA’s methods, respectively, observed from the experimental results and contributed to the outperformance of IDFP over them. In subsection 5-4 we outline the limitations of IDFP’s method that have been observed from the experimental results.

5.1. Strengths of IDFP predictive techniques

In 36% of the test wafer maps, IDFP was able to achieve exact hits. That is, IDFP predicted faulty tools that are exact match to the actual faulty tools caused the defect signatures on 36% of the test wafer maps. As Figure 10 shows, IDFP achieved hits greater than 50% in identifying the faulty tools that caused more than 50% of the defect signatures. This is an indicative of the good predictive capabilities of IDFP, which would allow process engineers to prioritize their investigation of faulty tools and to pay more attention to the ones that caused most defect signatures.

As Figure 11 shows, IDFP achieved smaller distances than the other two methods in most of the defect signatures. Intuitively, the smaller the distance, the more accurate is the prediction of...
a method. Thus, a zero distance represents an exact prediction match. IDFP achieved zero distances in heavy metal contamination and poor etching uniformity defect signatures. In general, and as Figure 11 shows, IDFP achieved small distances under all the defect signatures. It achieved distances less than 1 in 47% of the defect signatures. This confirmed the practical viability of IDFP’s method in accurately identifying the faulty tools that cause defect signatures.

Let $e_i(d)$ be the actual equipment that caused the defect $d$. The experimental results revealed that IDFP could identify either $e_i(d)$ or an equipment $e_j(d)$ close to $e_i(d)$ in the fabrication sequential order as the root cause of $d$. Actually, in real-world scenarios, $e_j(d)$ can contribute to the defect $d$ along with $e_i(d)$. That is, equipment that are close to one another in the fabrication sequential order can all contribute to a specific defect. Therefore, not all IDFP’s incorrectly predicted faulty equipment are considered false, since most of them are close to the actual faulty equipment. Moreover, these missed predictions can make it easier for process engineers to pinpoint the exact faulty equipment, since there are close to the actual faulty ones in the fabrication sequence.

5.2. Limitations of Chien method

Based on our observation of the experimental results, we outline the limitations of Chien’s method as follows:

(a) The method achieved poor prediction accuracy in each fabrication period (recall section 4.1) that do not have large test wafer maps. It also achieved poor prediction accuracy for defect signatures (recall section 4.2) that did not occur on a large number of test wafer maps. That is, the method requires a large sample size for its decision tree analysis. If the sample is not large enough, it terminates the decision tree immaturely, due to small observations in leaf nodes.

(b) The method was biased toward the predictors in the decision tree paths that have more levels.

(c) The method generated significantly different decision trees for the fabrication periods (recall section 4.1) that have small variations in test wafer maps. That is, the method produces very large changes in the decision trees even though the changes in wafer maps are minimal.

5.3. Limitations of SDCA method

Based on our observation of the experimental results, we outline the limitations of the SDCA’s method to the following:

(a) Similar to the Chien’s method, the SDCA’s method showed very sensitivity to small test wafer maps. It also achieved poor prediction accuracy in the fabrication periods (recall section 4.1) and the defect signatures (recall section 4.2) that do not have large test wafer maps. This is because the contingency table analysis employed by SDCA cannot obtain a significant association, if the sample size is small. As a result, the findings obtained by a contingency table may not be significant even though they are statistically significant.

(b) The method produced erroneous conclusions, when the expected frequency in a contingency table is small (e.g., less than 5). That is, it is very sensitive to small frequencies.

(c) The method did not permit a participant measured to fit in more than one category. That is, the method requires all measured individual participants to be independent.

5.4. Limitations of the IDFP method

The experimental results showed that, if a defect was caused by several faulty tools, IDFP may not identify all of them. However, it can identify at least one of them. For example, while IDFP achieved acceptable hit rates in identifying the faulty tools that contributed to aluminum bridging, it achieved relatively low hit rates in identifying the faulty tools that contributed to copper bridging. This is because the aluminum bridging in the evaluation dataset was due to under-etch caused by only wet etch tool, while copper bridging was due to several different tools, namely plating tools (which caused copper corrosion), litho-etch tools (which caused copper under-etch), and copper
CMP tools (which caused faulty smoothing). Even in such a scenario, the IDFP’s concept of CFP can narrow down the list of suspect tools, which saves the time of process engineers to identify the faulty one.

As shown in Figure 10, IDFP achieved relatively lower hit rates in identifying the faulty equipment that contributed to corrosive residue and misalignment defect signatures. This is because there are relatively larger number of fabrication equipment that caused these two defects (e.g., all photolithography steppers can cause misalignment and all CVD, ECD, and PVD equipment can cause corrosive residue). Therefore, the CFP concept may include multiple equipments that can cause these defects, which makes it harder for IDFP to pinpoint the exact faulty one.

6. Conclusions
We introduced in this paper a defect diagnostic system called IDFP that can identify the common fabrication path (i.e., the CFP) that caused most of the defectivities on wafers discovered at the end of the fabrication procedure. This path includes the set of all tools that is likely caused all the distinct set of defect signatures discovered by failure analysis in response to electrical failures captured by prober tools at the end of the fabrication procedure. Identifying all tools that caused all distinct defects is more effective than identifying individual faulty tools.

We evaluated IDFP by comparing it experimentally with Chien et al. (Chien et al., 2007) and SDCA (Taha, 2017), in terms of: (1) the precision of identifying the tools that caused defect signatures during different fabrication periods, (2) the percentage of hit, and (3) the closeness between the predicted and actual faulty tools in their steps in the fabrication sequence. We summarize below the major findings of the experimental results:

(a) IDFP achieved very good hit rates. It achieved hits greater than 50% in identifying the faulty tools that caused more than 50% of the defect signatures.

(b) IDFP achieved small distances between its predicted faulty tools and the corresponding actual ones in the fabrication sequence that caused defect signatures. Actually, it achieved small distances in all defect signatures. It achieved distances less than 1 in 47% of the defect signatures.

The above confirmed the practical viability of IDFP’s method to accurately identify the faulty tools that cause defect signatures. This allows process engineers to prioritize their investigation of faulty tools and to pay more attention to the ones that cause most defect signatures. In general, the experimental results revealed clear indicative of the good predictive capabilities of IDFP.

If a defect was caused by several faulty tools, IDFP may not identify all of them. Specifically, IDFP achieved relatively lower hit rates in identifying the root causes of defect signatures that caused by a larger number of fabrication equipment. Therefore, we need to investigate further approaches in a future work that can improve IDFP’s performance by overcoming the above limitation.

Funding
The author received no direct funding for this research.

Author details
Kamal Taha
E-mail: kamal.taha@ku.ac.ae
ORCID ID: http://orcid.org/0000-0002-6674-4614
Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, UAE.

Citation information
Cite this article as: An effective approach for identifying defective critical fabrication path, Kamal Taha, Cogent Engineering (2019), 6: 1575636.

References
Adly, F., Al Hussein, O., Yoo, P., Al-Hammadi, Y., Taha, K., Muhaidat, S., ... Ismail, M. (2015). Simplified subspaced regression network for identification of defect patterns in semiconductor wafer maps. IEEE Transactions on Industrial Informatics, 11(6), 1267–1276. doi:10.1109/TII.2015.2481719
Adly, F., Yoo, P., Muhaidat, S., Al-Hammadi, Y., Lee, U., & Ismail, M. (2015). Randomized general regression network for identification of defect patterns in semiconductor wafer maps. IEEE Transactions on Semiconductor Manufacturing, 28(2), 145–152. doi:10.1109/TSM.2015.2405252
Boly, R., & Haji, H. (2012). Wafer classification using support vector machines. IEEE Transactions on Semiconductor Manufacturing, 25(3), 373–383. doi:10.1109/TSM.2012.2196058

Burkene, F., Vedula, S., & Meeks, S. (2007). Visualizing the wafer’s edge. KLA-Tencor yield management solutions magazine (pp. 18–20). Spring. doi:10.1016/j.jsb.2006.07.007

Chen, F. L., & Liu, S. F. (2000). A neural-network approach to recognize defect spatial pattern in semiconductor fabrication. IEEE Transaction Semiconductor Manufacturing, 13(3), 366–373. doi:10.1109/66.857947

Chen, C.-F., Hsu, C.-Y., & Chen, P.-N. (2013). Semiconductor fault detection and classification for yield enhancement and manufacturing intelligence. Flexible Services and Manufacturing Journal, 25(3), 367–388. doi:10.1007/s10696-012-9161-4

Chien, C. F., Wang, W. C., & Cheng, J. C. (2007). Data mining for yield enhancement in semiconductor manufacturing and an empirical study. Expert Systems with Applications, 33(1), 192–198. doi:10.1016/j.eswa.2006.04.014

Chou, P. B., Ras, A. R., Stuvenbecker, M. C., Wu, F. Y., & Brecher, V. H. (1997). Automatic defect classification for semiconductor manufacturing. Machine Vision and Applications, 9, 201–214. doi:10.1007/s001380050061

Cunningham, S. P., & MacKinnan, S. (1998). Statistical methods for visual defect metrology, semiconductor manufacturing. IEEE Transactions, 11(1), 48–53.

de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. Computational Geometry: Algorithms and Applications (3rd ed.). Springer-Verlag.

Elmasri, R., & Navathe, S. (2007). Fundamentals of database systems. Addison-Wesley Computing.

Fenner, J. S., Jeong, M. K., & Lu, J. C. (2005). Feature extraction for defect detection in semiconductor wafers using model-based clustering. LNCS 2877, 662–673.

Friedman, D., Hansen, M., & James, D. (1997). Model-free estimation of defect clustering in integrated circuit fabrication. IEEE Transactions on Semiconductor Manufacturing, 10(3), 346–359. doi:10.1109/66.618208

Hansen, M., Friedman, D., & Nair, V. J. (1997). Monitoring wafer map data from integrated circuit fabrication process for spatially clustered defects. Technometrics, 39(3), 241–253. doi:10.1080/0040170.1997.10485116

Hsu, S. C., & Chien, C. F. (2007). Hybrid data mining approach for pattern extraction from wafer bin map to improve yield in semiconductor manufacturing. Journal of Production Economics, 25(107).

Iwata, H., Ono, M., Konishi, J., Isogai, S., & Furutani, T. (2000). In-line wafer inspection data warehouse for automated defect limited yield analysis. In Advanced Semiconductor Manufacturing Conference and Workshop (pp. 124–129). Boston, MA: IEEE/SEMI.

Lee, J. H., Yu, S. J., & Park, S. C. (2001a). Design of intelligent data sampling methodology based on data mining. IEEE Transactions on Robotics and Automation, 5(17), 637–649. doi:10.1109/70.964664

Lee, J. H., Yu, S. J., & Park, S. C. (2001b). A new intelligent SOFM-based sampling plan for advanced process control. Expert Systems with Applications, 20, 133–151. doi:10.1016/S0957-4174(00)00054-3

Liu, S. F., Chen, F. L., & Lu, W. B. (2000). Wafer bin map recognition using a neural network approach. International Journal of Production Research, 40, 2207–2223. doi:10.1080/002075402102122275

Palma, D., Nicolao, D., & Miraglia, G. (2005). Unsupervised spatial pattern classification of electrical-wafer sorting maps in semiconductor manufacturing. Pattern Recognition Letters, 26, 1857–1865. doi:10.1016/j.patrec.2005.03.007

Pukite, P. R., & Berman, C. L. (1990, Aug). Defect cluster analysis for wafer-scale integration. IEEE Transactions on Semiconductor Manufacturing, 3, 128–135. doi:10.1109/66.56569

Snoeck, M., & Dedene, G. (1998). Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types. IEEE Transactions on Software Engineering, 24(24), 233–251. doi:10.1109/32.677182

Taha, K. (2017). An effective approach for associating the sources of defect signatures to process zones. IEEE Transactions on Semiconductor Manufacturing, 30(2), 176–184. doi:10.1109/TSM.2017.2679714

Taha, K. (2018). CDID: A system for identifying the root cause of a defect in semiconductor wafer fabrication. IEEE Transactions on Semiconductor Manufacturing, 31(2), 221–231. doi:10.1109/TSM.2018.2808703

Taha, K., Salah, K., & Yoo., P. (2017). Clustering the dominant defective patterns in semiconductor wafer maps. IEEE Transactions on Semiconductor Manufacturing, 31(1), 156–165. doi:10.1109/TSM.2017.2768323

Widjaya, N., Taniar, D., & Rahayu, W. (2003). Aggregation transformation of XML schema to object-relational databases. Innovative Internet Community Systems, LNCS 2877 (pp. 251–262).

Xie, L., Li, D., & Simske, S. J. (2011). Feature dimensionality reduction for example-based image super-resolution. Journal of Pattern Recognition Research, 6(2), 130–139. doi:10.13176/11.323

Yuan, T., & Kuo, W. (2008). Spatial defect recognition on semiconductor wafers using model-based clustering and Bayesian inference. European Journal of Operational Research, 190, 1. doi:10.1016/j.ejor.2007.06.007

Zhou, C., Ross, R., Vickery, C., Mettee, B., Gross, S., & Verret, D. (2002). Yield prediction using critical area analysis with inline defect data. In Advanced Semiconductor Manufacturing 2002 IEEE/SEMI Conference and Workshop (pp. 82–86). Boston, MA.
