Defect Cluster Recognition System for Fabricated Semiconductor Wafers

Melanie Po-Leen Ooi[1, a,*], Hong Kuan Sok[1, b], Ye Chow Kuang[1, c],
Serge Demidenko[2, d], Chris Chan[3, e]

[1]Monash University, Sunway Campus, Jalan Lagoon Selatan, Selangor, Malaysia
[2]RMIT International University, 702 Nguyen Van Linh, Ho Chi Minh City, Vietnam
[3]Freescale Semiconductor, Sungeiway Free Industrial Zone, Selangor, Malaysia
[a]melanie.ooi@monash.edu, [b]sok.hong.kuan@ monash.edu, [c]kuang.ye.chow@monash.edu,
[d]serge.demidenko@rmit.edu.vn, [e]chrischan@freescale.com
*Corresponding author, Tel: + 603-55146238 and Fax: +603-55146001

Keywords: semiconductor wafer fabrication, defect cluster classification, recognition, feature extraction

Abstract

The *International Technology Roadmap for Semiconductors* (ITRS) identifies production test data as an essential element in improving design and technology in the manufacturing process feedback loop. One of the observations made from the high-volume production test data is that dies that fail due to a systematic failure have a tendency to form certain unique patterns that manifest as defect clusters at the wafer level. Identifying and categorising such clusters is a crucial step towards manufacturing yield improvement and implementation of real-time statistical process control. Addressing the semiconductor industry's needs, this research proposes an automatic defect cluster recognition system for semiconductor wafers that achieves up to 95% accuracy (depending on the product type).

1. Introduction

Advances in semiconductor technology and design have been the driving forces behind the successful progress of the electronics industrial sector. Over the past few decades, the semiconductor industry has evolved into one of the critical foundations and contributors of the world economy, recording a staggering 100% growth from the year 1996 to 2011 (Semiconductor Industry Association (SIA), 2011). Today, the semiconductor industry is generating revenues of over USD $25.5 billion and is expected to continue ascending by 10.5% annually between 2011 and 2013. Such rapid growth is going on hand-in-hand with a rapid increase of design and manufacturing complexity pushing the physical limits of semiconductor production techniques (International Technology Roadmap for Semiconductors, 2009). Reaching the nanoscale half-pitch dimensions alters the type and distribution of defects. This causes defects that were previously benign for microscale technology to manifest as killer defects.

The *International Technology Roadmap for Semiconductor* (ITRS), which is the world's authority on the semiconductor industry, has identified the detection of systematic failures as one of the top challenges in the next generation of semiconductor products (International Technology Roadmap for Semiconductors, 2009). Production data, in particular, was singled out as one of the crucial elements in aiding the process feedback loop.

With rapid developments in computer technology and with the availability of low-cost storage devices, data logging has become a commercially feasible task and is currently a standard procedure in almost all industries. The data logging ensures that virtually all industrial process-related data are always ready for extraction and analysis. Yet there is still much knowledge buried in the huge data collection, which is waiting to be discovered. Furthermore, the knowledge banks are continually enriched with new data being included and compiled each day. Thus, it is envisage that by applying new appropriate analysis tools further improvements in the semiconductor manufacturing could be achieved.

The production of modern microelectronic devices has an important feature that makes it significantly different from other manufacturing processes. The semiconductor wafer fabrication results in large number of *Integrated Circuits* (ICs) produced simultaneously in a multitude of sequential fabrication steps on a single piece of a silicon substrate (Fig. 1). Every fabricated wafer contains hundreds to thousands of devices, which are then tested, singulated (i.e., individually removed from the matrix of the products on the processed wafer) and packaged into individual protection casings thus making so-called IC *chips*. This unique characteristic whereby each individual device is produced together with other devices on a single piece of wafer allows production dataset to be physically and meaningfully interpreted as a two-dimensional "image".

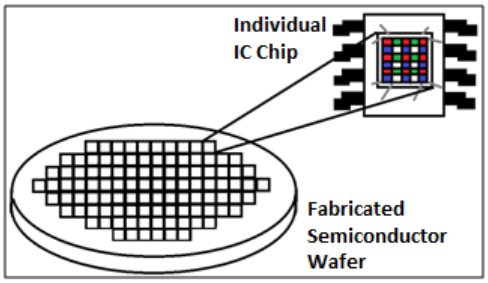
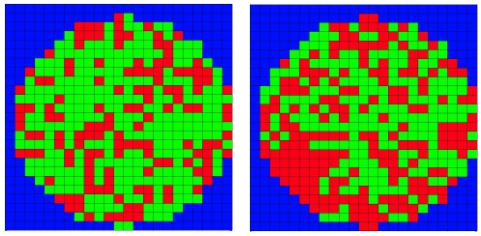


Fig. 1 Devices on a semiconductor wafer are fabricated simultaneously. Each die is singulated and packaged as individual *Integrated Circuit* (IC) product during IC *Assembly* process

Devices that fail due to an identifiable root-cause during the wafer fabrication process have a tendency to form unique and systematic patterns on the fabricated wafers (International Technology Roadmap for Semiconductors, 2009) (Zhao & Cui, 2008) (Ooi M. P.-L., et al., 2011). These are known as defect clusters. Fig. 2 shows two examples of raw production test data, whereby Fig. 2(a) shows random device failure (no identifiable defect clusters) and Fig. 2(b) shows a defect cluster (a group of failing devices clustered together) on the bottom-left edge of the wafer. Defect clusters are normally located around a specific location and are process-related. There are six local defect patterns identified and examined in this research, which are Bull's-Eye, Blob, Edge, Ring, Line and Hat (Fig. 3). Their formal description is provided in (Ooi M. P.-L., et al., 2011). Thus, the problem of detecting systematic failures can be simplified into detecting and identifying these defect clusters from the production test data.



(a) (b)

Fig. 2 Raw manufacturing wafer maps whereby red, green and blue squares represent failing, passing and invalid device locations respectively for: (a) Random failure; (b) Defect cluster at the bottom-left edge of the wafer

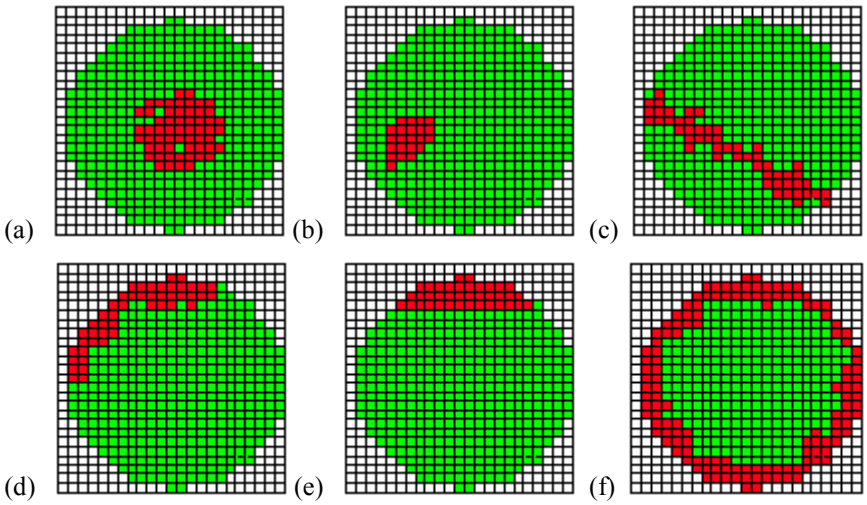


Fig. 3 Some examples of defect cluster patterns found on semiconductor wafers: a) Bull-Eye, b) Blob, c) Line, d) Edge, e) Hat, f) Ring

2. Intelligent defect cluster recognition system for fabricated semiconductor wafers

Developing a robust and accurate defect cluster recognition system for semiconductor wafers based on production test data is a non-trivial task. Unfortunately standard pattern recognition techniques do not work well for the semiconductor wafer based "images" due to several object-specific issues. These issues are discussed below Section 2.1. Section 2.2 outlines the aim of this research and the main contributions of the paper.

2.1 Challenges in developing an intelligent system for defect cluster recognition

. The main factors that must be overcome in order to develop a robust-yet-accurate defect cluster recognition system for semiconductor wafer datasets are listed below (they will be further elaborated below in individual subsections):

- Variations in defect cluster size, shape, location and orientation on the wafer for the same class of defects
- Non-symmetrical geometry
- Low signal-to-noise ratio

- Insufficient quality historical data for training
- *A-priori* unknown best feature sets and/or best classifiers

2.1.1 Variations of defect cluster size, shape, location and orientation for the same class of defects

It is difficult to select versatile feature vectors with strong distinguishing capabilities that describe the different classes of defects accurately. Fig. 4 shows two examples of defects, whereby Fig 4(a), 4(b), 4(c) and 4(d) are examples of Blob defects, while 4(e), 4(f), 4(g) and 4(h) are Line defects. The feature vector must be sufficiently flexible to account for the variations in defect sizes, shapes and locations on the wafer in the Blob defect examples, as well as the orientation in the Line defect examples. Thus, direct application of *principle component analysis* or *template matching* (e.g. techniques discussed in (Jain, Duin, & Mao, 2001)) without some form of transformation is clearly insufficient for this application.

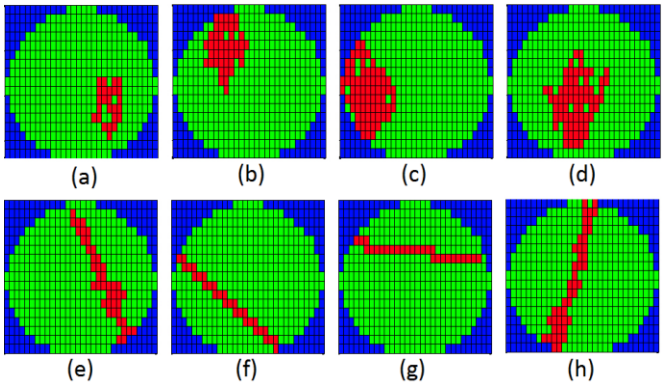


Fig. 4 Examples of different defect sizes, shapes, locations and orientations for Blob (a)-(d) and Line (e)-(f) defect types

2.1.2 Non -symmetrical geometry

Although the semiconductor wafer dataset can be viewed in a 2-dimensional space, it is still important to remember that each data point is an actual device being produced. Thus traditional normalization that graphically rotates the wafer map does not give meaningful results. Additionally, the location of each die on the wafer does not form a square dataset, but rather a circular pattern (Fig. 5 (a)). The wafer perimeter (valid die locations at the wafer edges) varies for different devices and different processes; the data points usually do not form a perfectly symmetrical pattern (see Fig. 5(b)). Thus, many classical algorithms would fail if applied to this subject area.

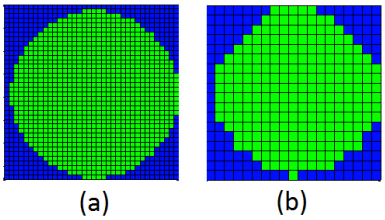


Fig. 5 Two examples of the semiconductor wafer dataset for actual devices in production whereby the blue and green squares represent the invalid and valid die locations respectively

2.1.3 Low signal-to-noise ratio

In most of the previous examples, only "clean" defect patterns were shown. In actuality, the real semiconductor production test dataset is more similar to that shown in Fig. 6. Fabricated semiconductor wafers have a relatively small number of points in the

dataset (ranging from 100 to 10,000 dies/wafer). Additionally, a high proportion of these small datasets (10%-50%) are random failures, which are interpreted as a "noise". The "noise" level differs depending on the manufacturing yield, which in turn depends on the die and wafer sizes, employed fabrication technology and other factors. Thus the failure distribution varies between different types of devices, different implementation technologies, etc.

It is important to note that clustering algorithms in pattern recognition and image processing have been developed for large numbers of pixel count (or points). With a small data count and high noise level, it is difficult to fine-tune most classical algorithms such that the *Type I/II errors* (false alarm and false rejection rates) would both be within acceptable limits. Furthermore, selecting an optimum threshold to differentiate between cluster and non-cluster regions (normally required in most clustering algorithms) is a non-trivial task when each device type has different numbers of dies per wafer, different configurations on a 2-dimensional space and completely different overall failure rates. A preliminary investigative study performed in (Ooi, Sim, Kuang, Demidenko, Kleeman, & Chan, 2011) shows that many clustering algorithms such as *k-means*, *k-medoid*, *mean-shift, Balanced Iterative Reducing and Clustering using Hierarchies* (BIRCH) and *Clustering Using Representatives* (CURE) do not provide meaningful results when applied to the research problem under consideration.

Fig. 6 shows an example of the Ring defect type for the same device type with three different manufacturing yields (40%, 70% and 90%). It can be seen that the noise level at 40% yield is very high, yet any applied intelligent recognition system must be capable of recognising the obvious defect at the wafer edges. Conversely, the noise level is low at 90% yield, but this brings a new problem: the data points that form the defect cluster are quite sparse. Yet, the defect recognition system must still be capable of recognising this defect pattern once it occurs.

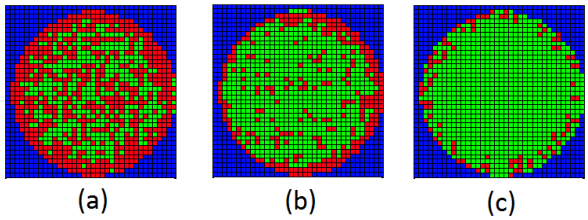


Fig. 6 Examples of the Ring defect type for manufacturing yield of (a) 40%, (b) 70% and (c) 90% whereby red, green and blue squares represent failing, passing and invalid device locations respectively

2.1.4 Insufficient quality historical data

The accuracy of any classifier depends greatly on the completeness of the training set. For the application under discussion, this translates to getting sufficient amount of high quality historical production data for training a classifier to account for the issues highlighted above such as variations in defect cluster size, shape, orientation and location with different manufacturing yield and noise levels. Unfortunately, even though semiconductor devices are manufactured in high volumes, it was observed that there is normally a lack of sufficient number or appropriate selections of good quality training samples available in the historical production test data-logs to obtain a stable statistical inference for a chosen classifier (Kameyama & Kosugi, 1999).

Semiconductor manufacturers produce thousands of different product types annually to cope up with the increasing market demands. At the same time new innovations in design and fabrication are routinely achieved and implemented. This results in

process changes being made on a monthly, if not weekly, basis (Hsieh, et al., 1999). There are also devices that are being produced in small volumes for niche applications. Thus, production life-cycles could be rather short for some semiconductor products leading to new more advanced device types being put into production. Furthermore, some defect clusters may not have yet been encountered during production (thus there will be no historical log on such cases), though this does not mean that they would not occur in future! The combined effect of these factors complicates the developmental efforts for any intelligent system (Hsieh, et al., 1999).

In general, it can be observed that approaches attempting to recognise the cluster type directly from the raw semiconductor production data encounter huge problems during implementation due to the above mentioned factors (Ooi M. P.-L., et al., 2011). To overcome this problem, a defect cluster simulator can be used to simulate the training set for the classifier. However, there are two important decisions are to be considered when using such an approach:

1. Should the training data contain "noise" to emulate the actual raw data? And if so, how much noise?
2. Should the training data contain only pure clusters, with the raw data filtered to extract a "noiseless" defect cluster? And if so, what filtering technique should be applied to the raw data?

2.1.5 Best feature set or most suitable classifier is not known a-priori

A good feature set is an optimised set of attributes that best distinguishes between different classes (Kononenko & Kukar, 2007). A good classifier is one that uses the feature set to obtain the most accurate grouping of data. A good classifier is generally required to test the efficacy of the features. Yet a good feature set is required to generate a good classifier (Kononenko & Kukar, 2007). This is a "chicken-and-egg" problem for most classification algorithms, since neither the best feature set nor the best classifier is known *a-priori*.

When the salient features are unknown, an extremely large set of features is used to train the classifier in a hope of obtaining accurate classification results. Yet, the reverse is often desirable, whereby efficient and optimal machine learning and recognition are achieved through careful selection of an optimal subset of salient features. This is because the memory requirements and computational time are expected to be kept low without compromising the accuracy of the classifier (Kononenko & Kukar, 2007; Jain, et al., 2001; Xu & Wunsch II, 2009). Thus, the chosen features should ideally be robust with respect to noise, easy to obtain and simple to interpret (Xu & Wunsch II, 2009).

There is still no known and accepted feature set or classifier that best suit this application. This leads to an open question in the developmental work on how to best begin developing the defect recognition system: feature first or classifier first?

2.2 Research aims and contributions

The ultimate aim of this research is to develop an optimal and accurate defect cluster recognition system for application on the semiconductor wafer production data. In order to achieve this aim, while basing on the earlier results in defect clustering (Ooi M. P.-L., et al., 2011), an experimental study must be performed to address the needs for classification. Such a study will determine the following:

1. The optimal feature set for chosen classifier (as discussed above in Section 2.1.1)
2. The best feature extraction method and type of classifier (as outlined in Sections 2.1.1, 2.1.2 and 2.1.5)

3. The classifier training method while taking into account the lack of historical data and high level of noise in the relatively small count(as presented in Sections 2.1.3 and 2.1.4)

Within the reported research, the outlined experimental study was conducted on large-scale industrial data of several million units of ICs from five types of semiconductor products that are in the mainstream high-volume production of one of the world's leading semiconductor device manufacturers. The experimental results are presented and discussed in Section 4. In order to follow this study, Section 3 provides the necessary theoretical foundation on suitable feature extraction methods and classifiers employed in the research.

3. Introduction into feature extraction, selection and classification

Feature extraction, selection and *classification* are inter-dependent tasks. *Feature selection* refers to algorithms that choose a small and optimal subset from a larger input set of features. It is normally performed during training of the classifier. *Feature extraction*, on the other hand, refers to algorithms that generate new and hopefully more useful features from the original set by applying a transformation (Xu & Wunsch II, 2009) (Jain, Duin, & Mao, 2001). Extraction of features may provide more discriminative ability compared to the best of the original features. However, transformation normally presents a more abstract representation of the data, thus the new feature may no longer be physically interpretable. Feature extraction normally precedes the feature selection stage. Finally, *classification* is a process of identifying a group (sub-population) to which particular observations belong to.

3.1 Feature Extraction and Selection

In the field of pattern recognition, dimension reduction techniques such as the *principal component analysis*, *factor analysis*, *linear discriminant analysis* and *projection pursuit* are highly popular feature extraction methods (Jain, Duin, & Mao, 2001). These techniques work well only if the defects in a same class are geometrically well aligned. Rotation of the wafer map to align defect cluster is not feasible due to non-symmetrical geometry and relatively large die size. Besides, such a rotation would cause significant number of dies to be misaligned.

Since semiconductor wafers are approximately circular in shape this characteristic can be considered as invariant in its rotational properties. Thus, potentially suitable methods to extract this particular characteristic are the *Polar Fourier Transform* (PFT) and *Rotational Moment Invariant* (RMI), which are discussed below in Sections 3.1.1 and 3.1.2 respectively. In addition, Section 3.1.3 provides the list of geometrical features to account for the variations in shape, size and defect location.

3.1.1 Polar Fourier Transform

The Polar Fourier Transform is similar to the traditional *Fourier Transform*, except that it considers the frequency spectrum in *polar coordinates* (Averbuch, Coifman, Donoho, Elad, & Isreali, 2006). Thus, it is invariant towards rotational properties. The polar grid of frequencies inside the concentric circle is defined as $\xi_{p,q} = \{\xi_\chi[p,q], \xi_y[p,q]\}$. Equation (1) below describes the discrete PFT operation, which includes a set of the samples $F(\xi_{p,q})$ with sample points governed by Equation (2) (Averbuch, Coifman, Donoho, Elad, & Isreali, 2006).

$$F\left(\xi_{p,q}\right)=\sum_{i_1=0}^{N-1}\sum_{i_2=0}^{N-1} f\left[i_1,i_2\right]e^{-i\left(i_1\xi_\chi[p,q]-i_2\xi_y[p,q]\right)} \tag{1}$$

$$\left\{\begin{array}{ll}\xi_\chi[p,q]=\dfrac{\pi p}{N}\cos\dfrac{\pi q}{2N} & \\ \xi_y[p,q]=\dfrac{\pi p}{N}\sin\dfrac{\pi q}{2N} & for(-N\le p\le N-1),(0\le q\le 2N-1)\end{array}\right\} \tag{2}$$

Fig. 7 shows an example of *Discrete Polar Fourier Transform* (DPFT) application on a line defect pattern whereby (a) and (b) show the before and after effect respectively. Comparing wafers A, B and C, it can be seen that DPFT maps any rotational variances as a shift along the *x*-axis. Thus to eliminate any rotational variances between different line defect patterns, the peak of the frequency is centered on the *x*-axis. In this manner, a similar signature is obtained for the same defect pattern, as shown in Fig. 7 (c). Fig. 8 shows the results of applying DPFT on the six different types of defect clusters after the peak centering.

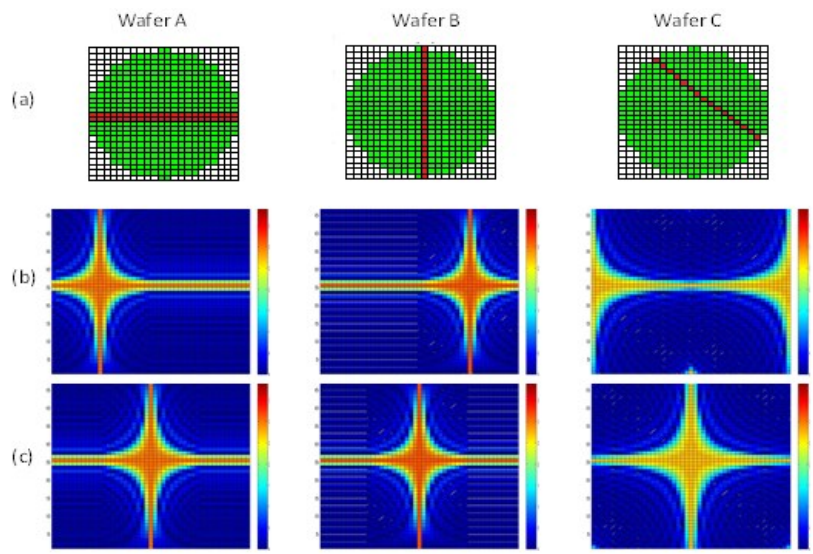
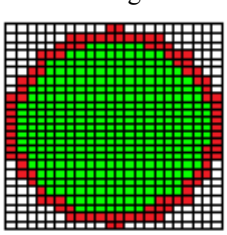
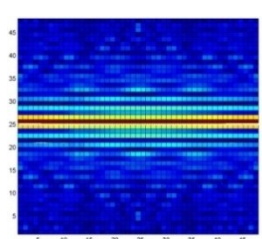
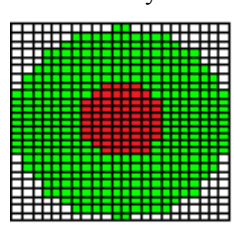
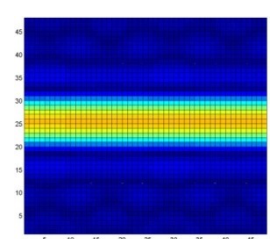


Fig. 7 Effects of DPFT on a rotated Line defect cluster: (a) wafer in X-Y coordinates; (b) wafer in frequency domain after applying DPFT; and (c) wafer in frequency domain after peak-centering

Wafer Yield Map	DPFT Domain	Wafer Yield Map	DPFT Domain
Ring 		Bull Eye 	
Hat		Line/Scratch	

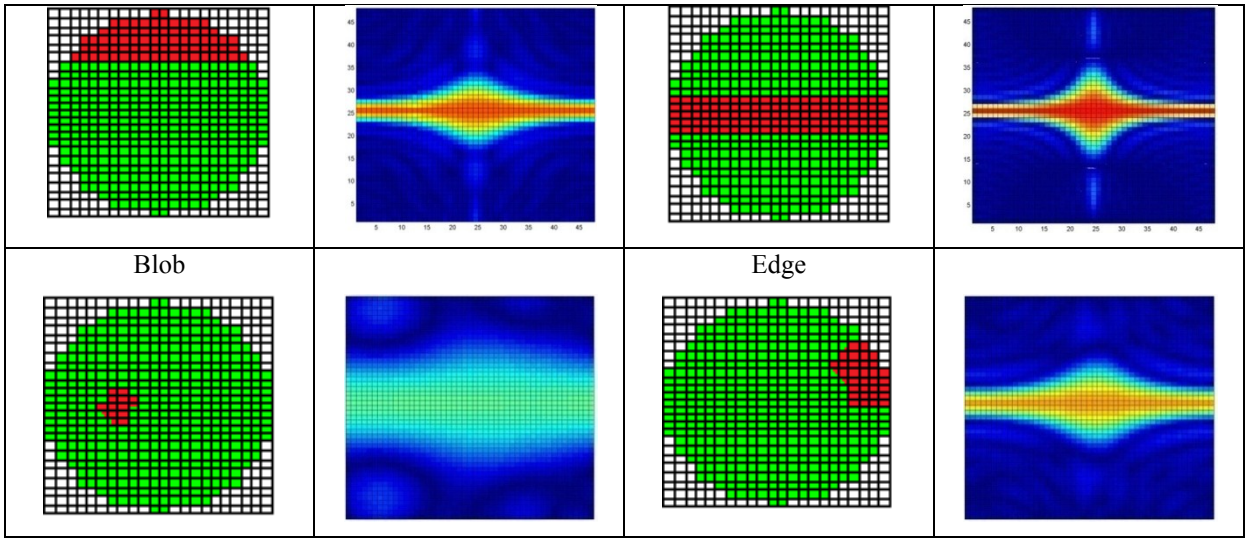


Fig. 8 Applying DPFT with peak-centering on different defect cluster types

Since each defect cluster type has a unique and identifiable frequency domain signature, DPFT can be considered as a possible feature extraction method for classification of the semiconductor defect clusters. However, it is important to note that the DPFT does not consider any translational property of the local defect cluster patterns. Therefore, additional geometric features should be considered for inclusion into the feature set to increase the classification accuracy. These are discussed below in Section 3.1.3.

The frequency signature for each defect cluster is obtained by calculating the eigenvectors using *Principle Component Analysis* - PCA (Jolliffe, 2002). It is important to note that the conventional PCA calculation is a lengthy process because the DPFT application leads to a high number of feature dimensions after transformation. Thus, a two-directional approximation called *(2D)² PCA* (Zhang & Zhou, 2005) is used instead, as it offers a higher computational speed without significant loss in a calculation accuracy.

3.1.2 Rotational Moment Invariants

Moment functions were first introduced in 1962 by Hu , who employed the results of the theory of algebraic and geometric moments to derive the seven famous invariants to the rotation of 2-dimensional objects which he used for automatic character recognition (Flusser, 2000), (Flusser & Suk, 2006). These are simple properties, which are found *via* image moments. They include the *area* or *total intensity*, its *centroid* and *orientation* information. Since then, moment invariants have become a classical tool in pattern recognition and object classification (Mukundan & Ramakrishnan, 1998).

A moment function Φ_{pq} of the order $(p+q)$ for a general 2-dimensional function $f(x,y)$ can be given as equation (3), where ζ denotes the image region of the x-y plane, which is the domain of the function $f(x,y)$ (Mukundan & Ramakrishnan, 1998); $\Psi_{pq}(x,y)$ is the *moment weighting kernel* or the *basis set* and is a continuous function of (x,y) in ζ, the indices p and q usually denote the degrees of the coordinates x and y respectively. Hu's invariants are listed in (Flusser, 2000).

$$\Phi_{p,q} = \iint_{\zeta} \Psi_{pq}(x,y) f(x,y) dxdy \tag{3}$$

Moment invariants with respect to image rotation can be easily derived from the complex moments. In particular, *Rotational Moment Invariant* (RMI) is a good tool for classifying defect clusters on semiconductor wafers because it extracts the same features for the same type of cluster regardless of the rotational differences. However, availability of Hu's invariant is very restrictive for a given order. Research paper (Flusser, 2000) proposes to use complex moments to generate more general rotational invariants, which include Hu's invariant as special cases. The complex moment $c_{p,q}$ of the order $(p+q)$ is an image moment function $f(x,y)$ as shown in equation (4) where i is an imaginary term. Each complex moment can be expressed in terms of geometric moments $m_{p\,q}$ in Cartesian or Polar coordinates (Flusser, 2000):

$$c_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x+iy)^p (x-iy)^q f(x,y)\,dxdy \tag{4}$$

$$c_{pq} = \sum_{k=0}^{p}\sum_{j=0}^{q} \binom{p}{k}\binom{q}{j} (-1)^{q-j}\, i^{p+q-k-j}\, m_{k+j,p+q-k-j} \tag{5}$$

$$c_{pq} = \int_{0}^{\infty}\int_{0}^{2\pi} r^{p+q+1} e^{i(p-q)\theta} f(r,\theta)\,drd\theta \tag{6}$$

According to (Flusser, 2000), if $n \geq 1$ and k_i, p_i and q_i; $i = 1, ..., n$, are non-negative numbers such that $\sum_{i=1}^{n} k_i (p_i - q_i) = 0$, then I in equation (7) is invariant to rotation.

$$I = \prod_{i=1}^{n} c_{p_i q_i}^{k_i} \tag{7}$$

Application of equations (4) and (7) generates a complex number which is the RMI feature. There is a large number of possible combinations of p_1, p_2, q_1, q_2, k_1, k_2 that generates rotational invariance property. However not all combinations are equally useful. Discretisation of the integration operation in Equation (4) negatively impacts the theoretical invariance property and causes perturbation in the resultant RMI feature. Certain combinations may be more sensitive to the discretisation errors. In this study $p_1,p_2,q_1,q_2 \in \{0.67,0.79,0.93,1.00,1.10,1.30,1.54,1.82,2.00,2.15,2.54,3.00\}$ were used because they are almost equally spaced numbers between the range of 0.67 (lower limit) and 3 (upper limit). These limits have been empirically selected to limit the dynamic range of the RMI value to prevent numerical instability. If no limits are enforced, the RMI value may be too large for representation even when using double precision numbers. Indeed, there are 20,736 possible combinations using the set of p_1,p_2,q_1,q_2 given above. Further restrictions are placed on k_1 and k_2 such that $k_1 = -\frac{p_2 - q_2}{p_1 - q_1 - p_2 + q_2}$ and $k_2 = -\frac{p_1 - q_1}{p_1 - q_1 - p_2 + q_2}$ to prevent numerical singularity during calculation. Apart from discretisation error, the geometrical characteristics of the wafer (imperfect circle) and target defect class also affect the robustness of RMI features. Table 1 shows several typical defect clusters and corresponding rotated version. It is clear that the cluster rotation is not exact due to the non-ideal wafer geometry (as mentioned earlier the semiconductor wafer is almost never a perfect circle). Even with the restriction of the upper and lower limits, the RMI value can go up to 10^4 range. Four arbitrarily chosen RMI are plotted in Table 2. The results show that despite various sources of error, the RMI of the rotated cluster still produces a similar complex number. In addition, cross validating various RMI values yields reliable classification of a given cluster.

Table 1 Two variations of Blob, Edge, Hat and Line defect patterns and their corresponding symbol used in RMI space

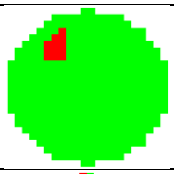
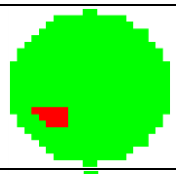
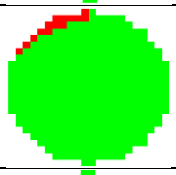
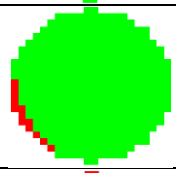
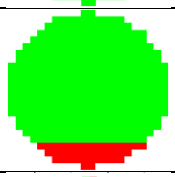
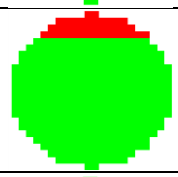
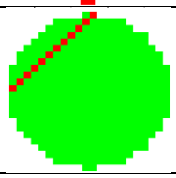
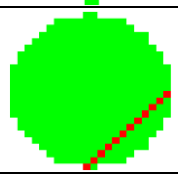
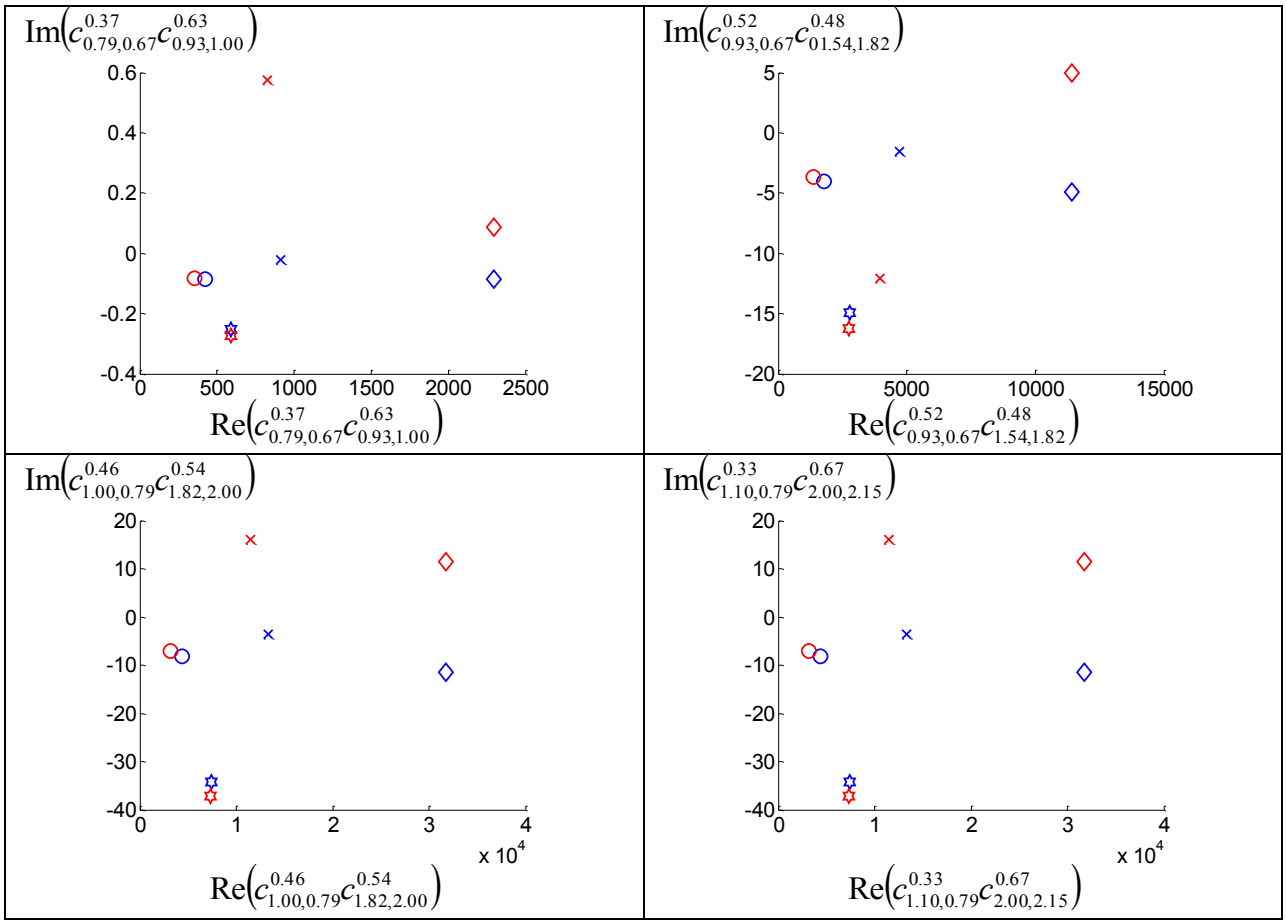
	Defect cluster		Rotated defect cluster	
	Wafer Yield Map	**Symbol in RMI space**	**Wafer Yield Map**	**Symbol in RMI space**
Blob				
Edge				
Hat				
Line				

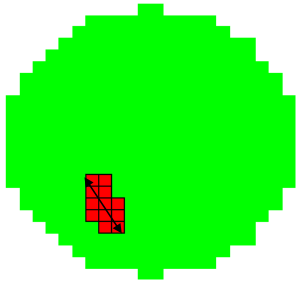
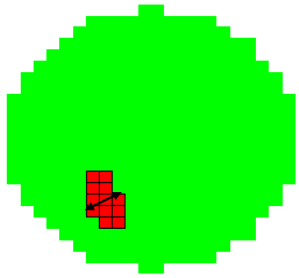
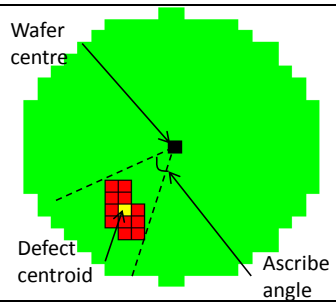
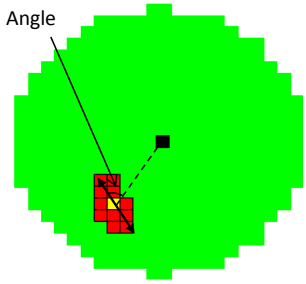
Table 2 Results of RMI for the Wafer Yield Maps using their corresponding symbols shown in Table 1

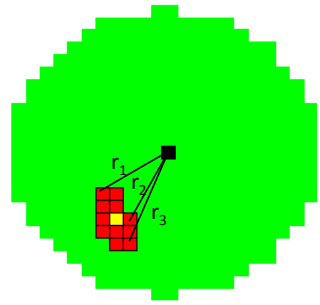
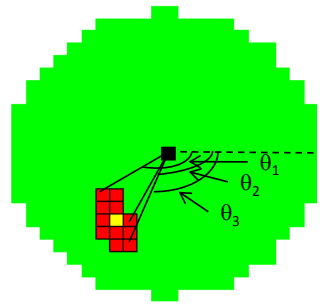
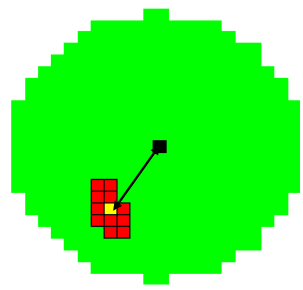
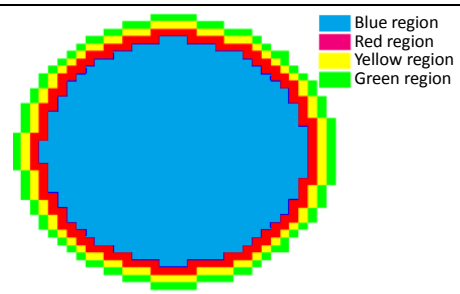


3.1.3 Geometrical Features

It is important to note that the PFT and RMI features discussed in Sections 3.1.1 and 3.1.2 do not consider any translational property of the defect clusters. These properties are necessary to distinguish between different defect classes. Thus, Table 3 provides a set of geometric-based rules that are normally used to distinguish between the types of defect clusters for inclusion into the feature set so to increase the classification accuracy (Guise, Poe, Stafford, & Wahba, 2002).

Table 3 Geometrical features to improve classification accuracy

Customised Features	**Illustration or Description**
1. Major axis length	
2. Minor axis length	
3. Aspect ratio	Aspect ratio = $\frac{Major\ axis\ length}{Minor\ axis\ length}$
4. Ascribe angle	
5. Angle between major axis and radial direction of the centroid	

6. Vectorial average of defect locations or Mean (r_i) where i references each location of the dies within the cluster	
7. Standard deviation of r_i	
8. Standard deviation angle, θ_i	
9. Distance between the centroid and the wafer centre	
10. Proportion of a defect in Region 1	 $\frac{\textit{Total dies in cluster within Green region}}{\textit{Number of dies in the cluster}}$
11. Proportion of a defect in Region 2	$\frac{\textit{Total dies in cluster within Yellow region}}{\textit{Number of dies in the cluster}}$
12. Proportion of a defect in Region 3	$\frac{\textit{Total dies in cluster within Red region}}{\textit{Number of dies in the cluster}}$
13. Proportion of a defect in Region 4	$\frac{\textit{Total dies in cluster within Blue region}}{\textit{Number of dies in the cluster}}$

3.2 Classification algorithms

Classification is the most popular application of all the machine learning methods. Classifiers utilise a *mapping function* to link the features to the respective class space (Kononenko & Kukar, 2007). The mapping function can be either given to the classifier or, more commonly, it is learnt from a given set of high quality training data. Different classifiers have different ways to represent the learning and mapping algorithms. Table 4 summarises the common classification methods in machine learning.

Table 4 Classification methods (Kononenko & Kukar, 2007; Jain, et al., 2001)

Classifier	Description	Comments
Decision trees and rules	Selects features and subsets of their values according to their quality, and uses them as a conjunctive rule's antecedent.	Fast classification (testing) time. At the same time, decisions trees generally tend to overlearn and pruning is needed. Rule sets that are generated for big trees are very complex and difficult to interpret.
Bayesian classifiers	Calculates the conditional probability of all classes and assigns a pattern to a class that has maximum likelihood. *Bayesian classifiers* (BC) use statistical probability calculations to achieve minimum error rate. *Naive Bayes classifier* (NBC) is a variation of BC that makes a statistical assumption on independence of features.	BC implementation is generally slow and complex. In addition BC has high computational requirements. If the number of features is too high, BC will face the *curse of dimensionality* (Bramer, 2007). NBC is faster and simpler to implement. However the independence assumption is often mot justifiable.
Nearest neighbours	Stores all the learnt data and assigns patterns to the most similar example according to a distance metric.	Easiest classifier to implement. No learning or training required, but has slow classification time for large dataset. Unable to handle spatial transformations such as rotation and translation. Requires special data structure to handle high dimensional datapoints. Prone to high error rates.
Discriminant functions	Linear classifier that finds the optimal classification boundary using mean-squared error optimisation.	Only works for linearly separable classes. Has limited accuracy for many applications.
Support vector machines	Maximises the distance between different classes by selecting the minimum number of support vectors. Can be understood as a non-linear extension of discriminant functions	Iterative and thus has high computational requirements. Doesn't work well for high-dimensional multiclass system.
Neural networks	Abstract classification that selects weights for each connection between neurons by learning them from training set. Assigns classes based on the calculated weights	Classification rules are not transparent. Assignment of weights during learning is not made clear. Prone to over learning. Prone to local minimum, which results in suboptimal solution
Hybrid functions	Combines two or more of the above approaches.	Introduces at least one extra layer of complexity where an interface is required to integrate the output of various classifiers into a single decision.

The semiconductor wafer datasets used in the reported research are *non-linear* with *multi-classes*. Thus *Nearest neighbours*, *Support vector machines* and *Discriminant functions* are typically unsuitable for this application. *Neural network* classifiers were initially considered. However, the neural network classification rule sets are not transparent to the user. This poses a serious problem in the industrial application. If there are any inaccuracies in classification, there is no method to determine their exact root-cause which impedes process improvement efforts. Therefore the use of this tool in the manufacturing process would not satisfy the *Six-Sigma quality audits* (Pande, et al., 2000), which are performed regularly by both the company and their clients. *Hybrid functions* are very complex to implement, and should only be considered if all other classification methods fail to provide suitable results. It is a general rule-of-thumb to use the "*Occam's Razor",* principle stating that given several classifiers with the same training error, the simpler classifier is more likely to generalise better.

Thus for this particular application, the suitable classifiers are the *Bayesian classifier* and variations of *Decision trees* which are: T*op-Down Induction Decision Trees* (TDIDT), *Bagging on TDIDT*, and *Boosting on the Alternating Decision Tree*.

Section 4 provides the experimental flow to determine the most suitable classifier for this application. To aid in reader comprehension, the following subsections provide some theory on the selected classifiers.

3.2.1 Bayesian Classifier

The *Bayesian classifier* is a simple statistical classifier based on Bayes' theorem (Bramer, 2007). Assuming that each class is C_k for k = [*1, ..., number of classes*] and n is a number of features F, the Bayes rule is written as Equation (8) and (9). The probability that an instance belongs to a particular class C_k given a combination of features ($F_1, F_2, ..., F_n$) is shown in Equation (10).

$$P(C_k, F_1, F_2, \ldots, F_n) = P(C_k) P(F_1, F_2, \ldots, F_n | C_k) \quad (8)$$

$$P(C_k, F_1, F_2, \ldots, F_n) = P(F_1, F_2, \ldots, F_n) P(C_k | F_1, F_2, \ldots, F_n) \quad (9)$$

$$P(C_k | F_1, F_2, \ldots, F_n) = \frac{P(C_k)}{P(F_1, F_2, \ldots, F_n)} = P(F_1, F_2, \ldots, F_n | C_k) \quad (10)$$

The prior probability of an instance belonging to class is $P(C_k)$. It is calculated even before any of its features are obtained (Bramer, 2007). Another prior probability is $P(F_1, F_2, ..., F_n)$, which is the likelihood of observing the $(F_1, F_2, ..., F_n)$ combination in the data. These two priors are normally estimated from historical datasets. The priors will normally help to adjust the prediction of class for better accuracy. For example, if historically 90% of the observed defect clusters belong to Edge defect pattern and 10% belongs to Bull's Eye, the prior of Edge cluster will be 0.9 and the prior of Bull's eye cluster 0.1. This will increase the likelihood that a cluster will be categorised as an Edge class. The Bayesian classifier can be readily used with discrete features. For continuous features, discretisation must first be performed, thus some discretisation error will inevitably be introduced into the system (Bramer, 2007).

3.2.2 Top-Down Induction Decision Trees

Decision tree is a flowchart-like tree structure that consists of internal nodes, branches and leaves (or terminal nodes) (Bramer, 2007). Each path starts at the topmost node (also known as the *root node*). Conditions are evaluated at each internal node, which corresponds to a test on an attribute. The outcome of the test is represented by each branch, which ends in a tree leaf. Each leaf corresponds to a discrete class label or decision rule (Kononenko & Kukar, 2007). A good choice of feature or attribute is absolutely vital for high accuracy in prediction. Some features are far more informative than others, and thus the best attribute must be determined by a measure of impurity (Kononenko & Kukar, 2007). This research uses *entropy* (Quinlan, Induction of Decision Trees, 1986) to measure the impurity. It is an information-theoretic measure of the uncertainty in the training set in the presence of one or more possible classification. Decision trees result in a symbolic and logical representation of the classification function, which normally conforms to a physical knowledge of the problem. Decision trees generally do not make any pre-determined assumptions on the statistical distribution of the dataset. This allows the features to be selected and used in an independent manner. The biggest problem for decision trees is the *empty* or *null leaf* phenomena (Kononenko & Kukar, 2007) (Bramer, 2007), where there is a valid path with no corresponding learning examples. This results in an unclassified instance.

3.2.3 Bagging on TDIDT

The natural extension of TDIDT is to apply *bagging* paradigm to reduce the problem of overlearning or over-fitting of the data (Polikar, 2006). Bagging attempts to improve classifier performance by generating many different classifiers. This is achieved

by training them on slightly different training sets as illustrated in Fig. 9 whereby *n* decision trees are produced. The class is chosen by a combined decision (or majority voting) of the different trees. The bagged TDIDT classification accuracy is normally comparable with that of the traditional TDIDT while it displays less sensitivity to noise from the training set and generalises better.

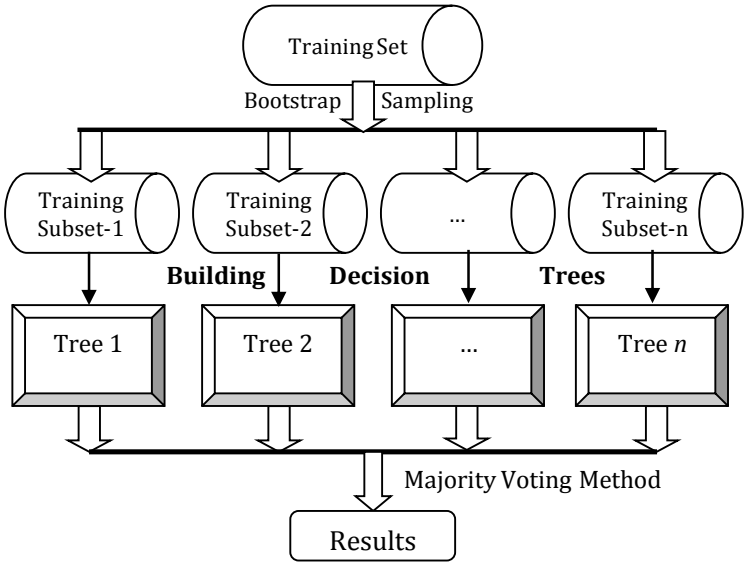


Fig. 9 Illustration of bagging on TDIDT

There are several drawbacks of using bagging on TDIDT. The rules used to classify an instance into a particular class are obscured when many trees are used in this manner. Bagging does not prevent the generation of contradictory rules from different trees. This negates the main advantage of decision trees over *black-box* approaches such as neural network, transparency of decision rules.

This paper only considers bagging on decision trees, which is equivalent to the standard *ID3* algorithm used for bagging in the machine learning community (Quinlan, 1986). The later extension of this algorithm is *C4.5*, which is widely considered as the best decision tree and includes *tree pruning*, *continuous attribute values*, *missing values* and *explicit rule generation* (Quinlan, 1992). Tree pruning is a process to remove unreliable tree branches that lead to a poor generalisation performance. According to (Bauer & Kohavi, 1999), bagging on decision trees works well without pruning because bagging itself reduces variance through the majority vote of many specialised trees. Pruning these trees on the other hand will generate many similar trees that give similar level of trade-off between specialisation and generalisation performance. Thus pruning bagged decision trees actually negates the benefit of statistical voting. A similar conclusion has also been drawn from another empirical study (Quinlan, 1996) whereby comparison between the *C4.5* and its bagged version produces similar error rates.

3.2.4 Boosting on the Alternating Decision Tree

Boosting was introduced in 1990 by Schapire (Schapire R. , 1990). It uses a set of several weak classifiers rather than one strong high performance classifier. In general, the boosting process involves training the next set of weak classifiers on a set of data points, which are reweighted according to the mistakes of the preceding classifiers. Freund and Schapire introduced an improved boosting algorithm known as *AdaBoost* in 1997 (Freund & Schapire, 1997), which is now among the top ten data mining algorithms (Wu, et al., 2007). There are other popular boosting algorithms available such as *LogitBoost* (Friedman, Hastie, & Tibshirani, 2000) and *BrownBoost* (Freund Y. , 2001). The BrownBoost was developed to address the learning

instability problem caused by outliers in the training set. It has a longer computation time since its termination condition requires solving a non-linear equation (Freund Y. , 2001). LogitBoost is an alternative approach to AdaBoost with comparable performance in most cases. However it can be far more complex to implement because it uses a regression model depending on the dataset (Friedman, Hastie, & Tibshirani, 2000).

All three boosting methods have comparable performances. In some cases, AdaBoost may outperform LogitBoost and BrownBoost, while in others - either LogitBoost or BrownBoost may provide the lowest error rate (Friedman, Hastie, & Tibshirani, 2000) (Freund Y. , 2001) (Schapire R. E., 1999). During implementation, parameters within the chosen boosting algorithm should be tweaked until the best classification performance is obtained. The fabricated wafer yield does not have a problem of outliers. Thus in order to lower the complexity of the classifier, AdaBoost is chosen over BrownBoost and LogitBoost for implementation in the reported research.

In general, boosting on the TDIDT generates better results than those of the bagging IDIDT (Quinlan, 1996). However it has a similar problem with bagging on TDIDT, whereby it generates complex trees with every boosting iteration. Thus, the final classification rules may be difficult (if not impossible) to interpret (Freund & Mason, 1999). Additionally, theoretical and empirical studies in (Reyzin & Schapire, 2006) and (Schapire, et al., 1998) show that if the base classifier is too complex for its intended application, the performance of the boosted learning process will be negatively affected. Therefore, the *alternating decision tree* (ADTree) was specifically developed for boosted learning.

ADTree controls the complexity of each base classifier by considering only two conditions at any time. Thus every node has only two branches. Each node in the ADTree has a prediction value, and the final prediction value is the sum of all predictions in the transverse path. A positive sum represents one class, while the negative sum represents the other class for a two-class problem. In this manner, ADTree is able to return a measure of confidence known as a *classification margin* apart from simply classifying an instance (Comite, et al., 2003). A higher classification margin indicates a higher probability of the cluster belonging to that class. Every training iteration adds a decision "*stump*", which is one node and two leafs, to the ADTree. A stump is the simplest form of a decision tree.

4. Experimental study

4.1. Experimental Setup

One of the world's leading semiconductor manufacturers was involved in and supported the reporting study by providing the necessary resources for testing and analysis. Those included several million units of integrated circuits of five types that were in the mainstream high-volume production (see Table 5). These devices were selected for experimental study in consultation with the manufacturing company and their characteristics were obtained from their respective design documents. They were specifically chosen to be of different technologies (half-pitch sizes), different complexity, number of metallization layers and dies/wafer counts. The true names and functional descriptions of the devices have intentionally been changed in this paper for confidentiality reasons.

Table 5 Devices used in the experimental trials

Devices	Half-Pitch size (nm)	Dies / Wafer	Metallisation Layers	Total ICs

A	250	984	3	2,460,000
B	250	794	3	1,985,000
C	130	402	6	1,105,500
D	90	384	7	960,000
E	250	235	3	587,500

Three experiment rounds were performed to address the challenges discussed in Section 2. The first one (Section 4.2) was on classifier training, whereby the following three classifier training and implementation methods were compared:

1. Using historical production data to train a classifier with implementation on raw production test data;
2. Using simulated data (with noise) to train a classifier with implementation on raw production test data;
3. Using simulated data (noiseless) to train a classifier with implementation on filtered production test data.

Section 4.3 below describes an experiment that was performed to determine the best combination of rotational feature vector (discussed in Section 3.1.1 and 3.1.2) and classifier (discussed in Section 3.2). Finally, the defect cluster recognition system is concluded in Section 4.4 with an experiment on the optimal feature set. Section 4.5 shows the proposed system flow for implementation on the semiconductor production test floor.

4.2 Classifier training

In order to perform this experiment, the type of classifier must be held constant while changing the training method. The boosting on ADTree was selected for the experiment due to its superiority over the other classifiers from the list. This is because the boosting on ADTree was originally developed to overcome some of the weaknesses of previous versions of decision trees. It is important to note that although the ADTree was used in this experiment, the best classifier for this application was still unknown at the time of experimentation. At the same time, it could be expected that the classifier training results would scale accordingly when applied to other decision trees, as they would face the same problems during the training and implementation cycle.

4.2.1 Experimental Flow

Fig. 10 shows the experimental flow for classifier training methods. At this point, the optimal feature set was still unknown, hence only the RMI features discussed in Section 3.1 were used during the Feature Extraction stage. This is because using all the features would result in an extremely large feature set, which would result in an extremely lengthy computation time.

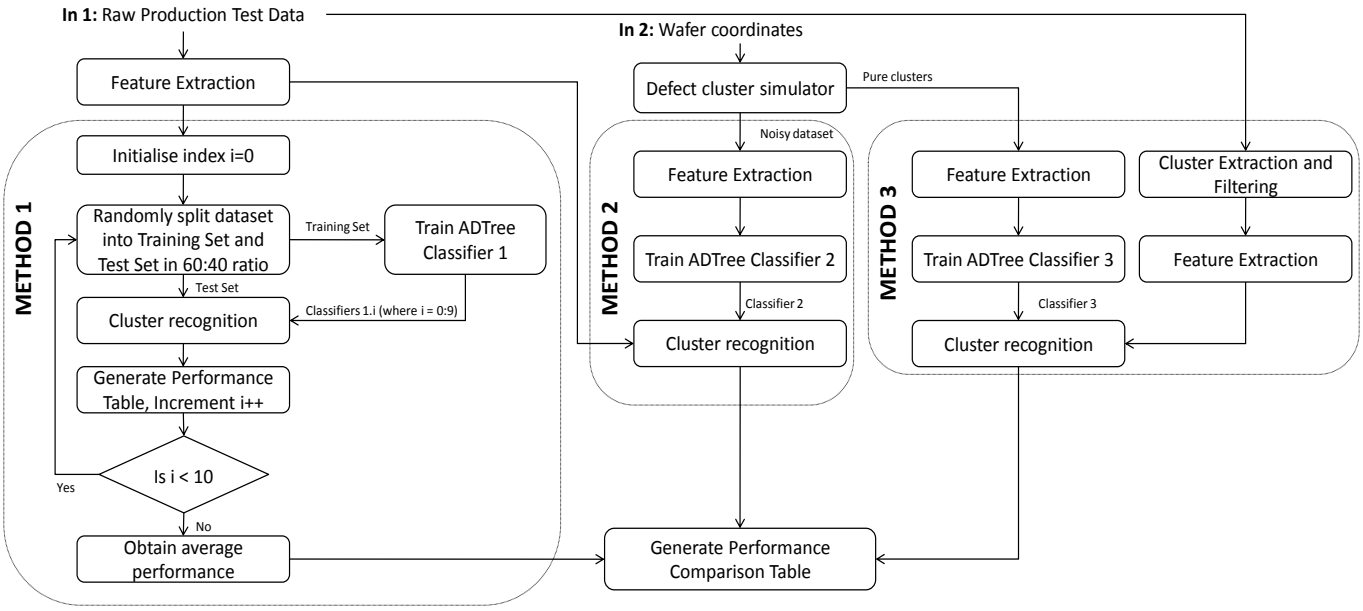


Fig. 10 Experimental flow to determine the best training and implementation method for the classifier

In the first flow (Method 1 shown in Fig. 10), raw production test data was used to train and test the ADTree classifier. Hence, the dataset was divided into a training set and a test set. This is the standard train-and-test method used in supervised learning algorithms (Kononenko & Kukar, 2007). In order to reduce variability and to provide a better generalisation of the classifier performance, ten-fold cross-validation was performed, whereby the dataset was randomly divided into training and test sets in ten iterations and the validation results were averaged.

The second and third experimental flows (Methods 2 and 3 shown in Fig. 10) involved the use of a defect cluster simulator, which will be discussed in detail below. To train the classifier, Method 2 uses "noisy" simulated wafers, or wafers that are simulated to emulate real production dataset. The amount of "noise" or random failure depends on the production yield. Thus, the average wafer yield for each device must be known prior to simulation. This is done to ensure that sufficient numbers of samples of high quality training data are available for training. Method 3 trains the classifier using pure (noiseless) defect clusters. Thus to test the classifier, raw production data must be "filtered" to remove the random failure and to extract the defect clusters. The accuracy of the classifier would thus depend not only on the training set, but also on the effectiveness of the noise filtering technique. While other filtering methods can be used such as (Wang, 2007; White, et al., 2008), this research applied the *Segmentation, Detection and Cluster Extraction* (SDC) algorithm in (Ooi M. P.-L., et al., 2011) as the filtering method of choice.

Any wafer can be simulated if the valid device locations (*x*-*y* positions on the wafer) are known. Thus, each defect cluster shown in Fig. 3 was simulated using the defect simulator algorithm shown in Fig. 11.

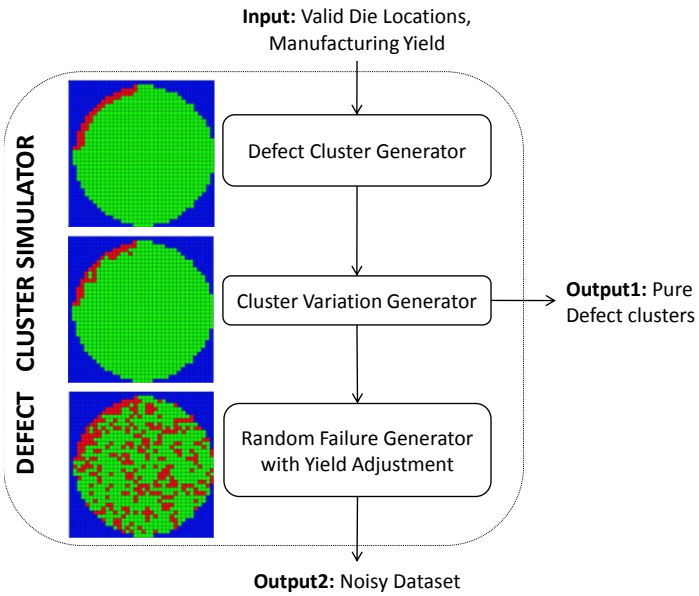
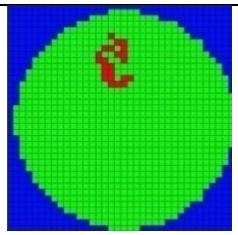
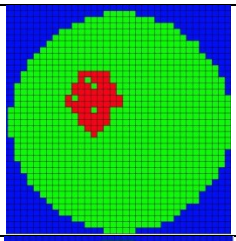
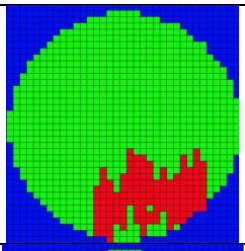
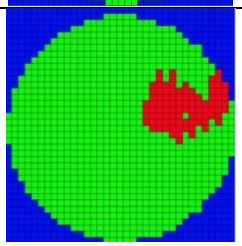
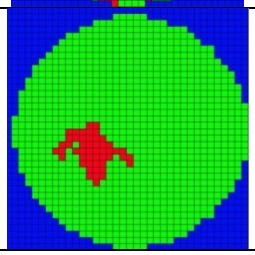
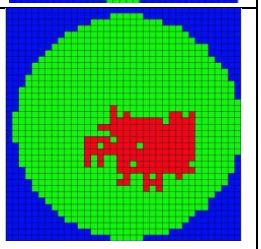


Fig. 11 The defect cluster simulator

The simulator produces "perfect" defect cluster patterns using the set of knowledge rules provided in (Ooi M. P.-L., et al., 2011) in the first stage. In the second stage, cluster variations are included to obtain more realistic defect patterns. These variations are achieved using the rules shown in Table 6. In the final stage, random failures are included into the wafer map based on a predetermined wafer yield. The outputs of the second and third stages were used in Methods 3 and 2 respectively.

Table 6 Rules for generating cluster variations for the defect cluster simulator

Rules and description	**Variation1**	**Variation2**
1. Defect pattern failing probability *Number of dies that fail in a specified cluster can be reduced by setting a lower failing probability, as shown in Variation 1. Alternatively, a higher failing probability will result in more failing dies in the cluster, as shown in Variation 2.*		
2. Defect pattern jitter size *Noise is added into the cluster to simulate imperfect cluster shapes. This is termed as "jitter". The jitter size can be large (Variation 1) or small (Variation 2).*		
3. Defect pattern jitter failing probability *The additional jitter can have 100% passing probability, or 100% failing probability, which changes the defect cluster size as shown in Variations 1 and 2 respectively.*		

4.2.2 Experimental results and discussions on classifier training methods

In the reported research data equivalent to over 300,000 different wafers were generated using the defect cluster simulator for the classifier training threshold selection. This included 5,000 wafers for each of the 6 defect types, with and without random failure and 5 device types. The reason for simulating 5,000 wafers for each point was to ensure that the classification criteria were observed to converge to an asymptotical value. (Ooi, Sim, Kuang, Demidenko, Kleeman, & Chan, 2011) provides some examples on verification of the simulation stability.

Table 7 shows the classification results for the three trialed methods. It must be noted that Devices A, B and D did not encounter all the defect types during the three month experimental period. For Device A, only Edge, Blob and Hat defect types were encountered, while for Devices B and C only Edge and Blob defect types were found. Thus, although Method 1 appears to be fairly accurate for Devices A, B and C, it is likely to fail if the production lines encountered a different defect type in future. Overall, Method 3 performs fairly consistently across the different device types compared to Methods 1 and 2. Thus, it can be concluded that Method 3 gives a better generalization for application across many different devices being produced by the manufacturer.

Table 7 Correct identification rates

Device	Method 1 Accuracy (%)	Method 2 Accuracy (%)	Method 3 Accuracy (%)
A	55.26	51.47	69.5
B	92.06	29.06	66.7
C	36.3	55.38	51.1
D	52.94	17.19	64.8
E	29.17	27.25	48.4

4.3 Experiment on combinations of rotational feature vectors with different classification algorithms

Two feature extraction methods capable of extracting defect cluster information while accounting for different orientation on the wafer were discussed earlier in Section 3.1.1 and 3.1.2. These are: *Rotational Moment Invariants* (RMI) and *Discrete Polar Fourier Transform* (DPFT). Both feature sets are subsequently used to train the four shortlisted classifiers, which are the *Bayesian classifier*, *TDIDT*, *bagging on TDIDT* and *boosting on the ADTree*.

4.3.1 Experimental flow on feature-classifier combination

Two feature vector sets were obtained as shown in Fig. 13 - the RMI and the DPFT feature sets respectively. The shortlisted classifiers were trained based on the respective feature sets (Fig. 13). Thus after the training, eight different classifiers were generated and compared against each other. The classifiers were trained and implemented using the method determined in Section 4.2.2, whereby data from the defect cluster simulator were employed to generate the training set. Production test data from the devices shown in Table 5 were used to generate the classifier performance table (after applying the SDC algorithm).

Fig. 13 Experimental flow to obtain the most optimum combination of feature extraction and classifier

4.3.2 Experimental results and discussions on feature-classifier combination

Table 8 shows the performance for every feature-classifier combination for each defect cluster type when applied on the production dataset. It can be observed that the TDIDT and bagging on decision trees have poor average hit rates. The poor performance of the TDIDT is mostly caused by the unresolved missing branch (or null hypothesis) problem. Although bagging was expected to increase the performance, it did not manage to achieve so in this application.

It is clear that the pairing of RMI with boosting on ADTree results in the overall performance, while PFT with boosting on ADTree has comparable performance. However, PFT describes clusters in frequency domain, which makes any refinement or modification on its features more difficult to perform. Additionally, RMI has a smaller feature spaces dimension, which makes it less affected by the curse of dimensionality when geometrical features are included into the feature set (Section 4.4). This leads to the conclusion that the feature-classifier combination that best suits the wafer-detection and classification application for this research is RMI with ADTree classifier. Its overall accuracy is still below 70% because geometrical features have not been considered. Thus, the description of some defect clusters (in particular, the Blob defect type) is poorly defined, leading to very low identification rates. To overcome this weakness, a third and final experiment on the optimal feature set is discussed in the next section to increase its suitability for the real-world implementation.

Table 8 Correct identification rates in (%) on production test data

Classifier	**Bayesian Classifier**		**TDIDT**		**Bagging on Decision Tree**		**Boosting on ADTree**	
Feature Used	**PFT**	**RMI**	**PFT**	**RMI**	**PFT**	**RMI**	**PFT**	**RMI**
Bull's Eye	50.0	80.8	57.7	7.7	50.0	7.7	50.0	84.6
Blob	75.5	37.4	53.6	24.8	31.1	32.5	50.7	19.5
Hat	52.8	26.4	30.2	24.5	0.0	18.9	49.1	39.6
Ring	71.4	57.1	71.4	42.9	0.0	0.0	100	100
Line	16.5	45.9	25.9	36.5	43.5	62.4	4.7	52.9
Edge	49.3	41.9	51.4	32.7	0.0	23.0	88.4	65.2
Average	**52.6**	**48.3**	**48.4**	**28.2**	**20.8**	**24.1**	**57.2**	**60.3**

4.4 Experiment on optimal feature set for defect cluster recognition system

Two different features sets were used to generate two different ADTree classifiers each for *Devices A-E* (Table 5). The *RMI Feature Set* consisted of only RMI features, while the *Extended Feature Set* comprised RMI and geometrical features (refer to Section 3.1.3). This was done to compare the differences in accuracy with and without the geometrical features. The results are shown in Table 9.

It is clear from the experimental results that the ADTree classifier generated by *Extended Feature Set* is far superior in terms of accuracy. It offers average percentage point improvement of approximately 30% while achieving a very good recognition accuracy of up to 96% depending on the product type. Thus it can be well recommended as the Defect Cluster Classifier for semiconductor wafers.

Table 9 Accuracy comparison between the classifiers generated by different feature sets

Device	RMI Feature Set Classifier Accuracy (%)	Extended Feature Set Classifier Accuracy (%)	Percentage point improvement
A	69.5	90.3	20.8
B	66.7	94.7	28.0
C	60.3	81.5	21.2
D	64.8	95.9	31.1
E	48.4	84.1	35.7

4.5 Proposed defect cluster recognition system

The defect cluster recognition system for fabricated semiconductor wafers shown in Fig. 14 is derived using the results of the three experiments discussed above in Sections 4.2, 4.3 and 4.4. The *offline* process involves the generation of the features and training of the ADTree classifier, which can take up to a few minutes depending on the performance of the employed computer system. For example in this research, the training time was approximately 10 minutes on a commodity computer powered by a Intel ® Pentium ® T4400 (2.2GHz) processor. Using a faster processor will reduce the training time. It is important to note that processor speeds increases every year, thus the training time will be greatly shortened within the next few years. Additionally, this process is performed one time only for each device in an offline mode, and thus it does not affect the manufacturing throughput. On the other hand, the classification time is practically negligible for all the data in the manufacturing dataset due to the built-in simplicity of ADTree. As a result, the proposed system is capable of running in an *online* mode of application without negatively impacting the manufacturing throughput.

It is important to note that since simulation is used to produce the training set for the classifier, thus the type of defect cluster that is recognizable by the system is limited to the type of defect that is being trained. In this paper, the types of defect that the system is trained to recognize are limited to Bulls-eye, Blob, Line, Edge, Ring and Hat defect types. However, it can be trained to recognise a new defect type by

specifying its geometry and simulating it.

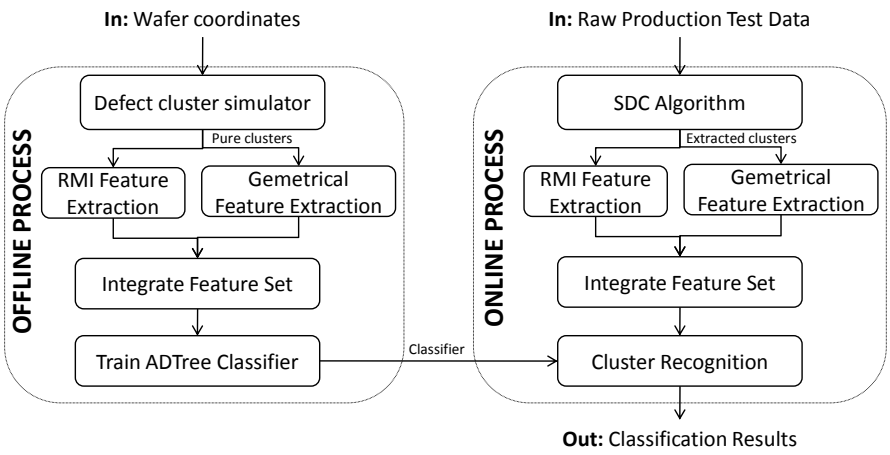


Fig. 14 Proposed defect cluster recognition system for semiconductor wafers

The defect cluster simulator overcomes the problem of insufficient training samples, while complementary application of the SDC algorithm (Ooi M. P.-L., et al., 2011) filters the noisy raw production dataset. The combination of these algorithms overcomes the problem of the low signal-to-noise ratio discussed in Section 2.1. The application of RMI feature extraction method with extended geometrical features provide good distinguishing for defect clusters encountered in semiconductor wafer fabrication, while its combination with the ADTree classifier results in an overall performance of up to 95% depending on the device type.

5. Conclusions

This paper presents a defect cluster recognition system that automatically recognizes several types of known defect clusters found on fabricated semiconductor wafers. The proposed system generates an ADTree classifier that achieves classification accuracy of up to over 95% depending on the product type. Incorporation of the classifier into an online production data analysis system allows any type of known defect cluster encountered during the manufacturing process to be quickly and automatically identified. This provides a very valuable information that can be used for fast fault diagnosis and rapid root cause identification, which in turn leads to a better process control.

Acknowledgements

The authors would like to thank the industrial partner, Freescale Semiconductor for the provision of data, resources and equipment for this research and Monash University for the scholarship support. This research was supported by Malaysian Ministry of Higher Education Fundamental Research Grant Scheme FRGS/1/2011/SG/MUSM/03/1.

References

Averbuch, A., Coifman, R., Donoho, D., Elad, M., & Isreali, M. (2006). Fast and accurate Polar Fourier Transform. *Applied and computational harmonic analysis, 21*, 145-167.

Bauer, E., & Kohavi, R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning, 36*(1-2), 105-139.

Bramer, M. (2007). *Principle of data mining.* Springer.
Comite, F., Cilleron, R., & Tommasi, M. (2003). Learning multi-label alternating trees from texts and data. *Machine Learning and Data Mining in Pattern Recognition*, 251-274.
Flusser, J. (2000). On the independance of rotation moment invariants. *Pattern Recognition 33*, (pp. 1405-1410).
Flusser, J. (2000). On The Independence of Rotational Moment Invariants. *Pattern Recognition, 33*, 1405-1410.
Flusser, J., & Suk, T. (2006). Rotation moment invariants for recognition of symmetric object. *IEEE Transaction on Image Processing, 15*(2), 3784-3790.
Freund, Y. (2001). An Adaptive Version of the Boost by Majority Algorithm. *Machine Learning, 43*(3), 293-318.
Freund, Y., & Mason, L. (1999). The alternating decision tree learning algorithm. *Proceedings of the 16th International Conference on Machine Learning*, 124-133.
Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences, 55*(1), 119-139.
Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting. *Annals of Statistics , 28*(2), 337-407.
Guise, M., Poe, D., Stafford, J., & Wahba, A. (2002). Automated Defect Pattern Recognition: An Approach to Detect Classification and Lot Characterisation. *IEEE System and Information Design Symposium*, (pp. 63-68).
Hsieh, S., Lin, S.-C., Lee, M.-H., Wang, J.-R., Lin, C., Huang, C.-W., et al. (1999). Novel Assessment of Process Control Monitor in Advanced Semiconductor Manufacturing: A Complete Set of Addressable Failure Site Test Structures (AFS-TS). *Proceedings of Semiconductor Manufacturing Conference*, (pp. 241-244).
International Technology Roadmap for Semiconductors. (2009). *Test and Test Equipment 2009 Edition.* Report, International Roadmap for Semiconductors, http://www.itrs.net/Links/2009ITRS/Home2009.htm.
Jain, A. K., Duin, R. P., & Mao, J. (2001, January). Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(1), 4-36.
Jolliffe, I. (2002). *Principal Component Analysis (Springer Series in Statistics)* (2nd edition ed.). New York, United States of America: Springer.
Kameyama, K., & Kosugi, Y. (1999). Semiconductor defect classification using hyperellipsoid clustering neural networks and model switching. *International Joint Conference on Neural Networks*, (p. 3505).
Kononenko, I., & Kukar, M. (2007). *Machine Learning and Data Mining: Introduction to Principles and Algorithms.* Horwood.
Mukundan, R., & Ramakrishnan, K. (1998). *Moment Functions in Image Analysis: Theory and Applications.* Danvers: World Scientific Publishing Co. Pte. Ltd.
Ooi, M. P.-L., Sim, E. K., Kuang, Y. C., Demidenko, S., Kleeman, L., & Chan, C. W. (2011, March). Getting More From the Semiconductor Test: Data Mining With Defect-Cluster Extraction. *IEEE Transactions on Instrumentation and Measurement, PP*(99), 1.
Pande, P. S., Neuman, R. P., & Cavanagh, R. R. (2000). *The Six Sigma Way: How GE, Motorola, and Other Top Companies are Honing Their Performance* . New York, United States of America: Mcgraw-Hill.
Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine, 6*(3), 21-45.
Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning, 1*(1), 81-106.
Quinlan, J. (1992). *C4.5: Programs for Machine Learning.* USA: Morgan Kaufmann.
Quinlan, J. (1996). Bagging, Boosting and C4.5. *Proceedings of the 13th National Conference on Artificial Intelligence, AAAI 96*, (pp. 725-730). Portland.

Reyzin, L., & Schapire, R. E. (2006). How Boosting the Margin Can Also Boost Classifier Complexity. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 753 - 760). Pittsburgh: ACM.

Schapire, R. (1990). Thre strength of weak learnability. *Machine Learning, 5*(2), 197-227.

Schapire, R. E. (1999). Theoretical Views of Boosting and Applications. *Proceedings of the 10th International Conference on Algorithmic Learning Theory* (pp. 13 - 25). Tokyo, Japan: Springer-Verlag.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics, 26*(5), 1651-1686.

Semiconductor Industry Association (SIA). (2011, March 7). Semiconductor Industry Reports January Chip Sales Grew 14.0% Year over Year. WASHINGTON, D.C.

Wang, C.-H. (2007). Recognition of Semiconductor Defect Patterns Using Spectral Clustering. *IEEE International Conference on Industrial Engineering and Engineering Management*, (pp. 588-591). Singapore.

White, K., Kundu, B., & Mastrangelo, C. (2008, May). Classification of Defect Clusters on Semiconductor Wafers Via the Hough Transformation. *IEEE Transactions on Semiconductor Manufacturing, 21*(2), 272.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., et al. (2007). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1-37.

Xu, R., & Wunsch II, D. C. (2009). *Clustering.* Danvers, United States: Wiley.

Zhang, D., & Zhou, Z. (2005, December). (2D)2 PCA: Two-directional two-dimensional PCA for efficient face representation and recognition. *Neurocomputing, 69*(1-3), 224-231.

Zhao, X., & Cui, L. (2008). Defect Pattern Recognition on Nano/Micro Integrated Circuits Wafer. *Proceedings of the 3rd IEEE International Conference on Nano/Micro Engineered and Molecular Systems, NEMS 2008*, (pp. 519-523). Sanya, China.