DETECTING SEMICONDUCTOR DEFECT FEATURES AND CLASSIFYING WAFER

https://doi.org/10.59982/18294359-23.14-ds-16

Garegin Sargsyan
Candidate of sciences in physics and mathematics
AMD ARMENIA
garegin.sargsyan.v@gmail.com
Artur Matevosyan
SYNOPSYS ARMENIA
artur.matevosyan99@gmail.com

Abstract

This scientific work focuses on the development of a software tool for semiconductor defect feature detection and wafer classification. The tool utilizes advanced image processing techniques to extract relevant information from semiconductor wafers and classify them based on their defects. The development process involves the integration of various algorithms and machine learning models to optimize the tool's performance. The results of the study demonstrate the effectiveness of the tool in accurately detecting and classifying semiconductor defects, which can aid in improving semiconductor manufacturing processes and reducing defects. The proposed software tool has the potential to become an essential tool for the semiconductor industry and contribute to the advancement of semiconductor technology.

The aim of the work is to classify semiconductor disk (wafer) defects. The input data, presented in tabular form, contains the coordinates of physical defects in the disk coordinate system, their sizes, belonging to a particular production process. Based on the location of defects and their properties, the processed software classifies disks into groups in order to clarify and eliminate further causes of damage.

Keywords: wafer, C++, microelectronic, wafer defect.

Introduction

A wafer is a round plate made of silicon material. There are square elements on the wafer, each of which represents a microcircuit. That square element is called the Die, which is the chip that provides the functionality. There are different steps in the production of a microcircuit and at each step some layer of the microcircuit is added and defects may occur in that layer. Defects may occur for various reasons. This may be due to particles in the outer atmosphere or to the device in which we are trying to place the plate, etc. Defects cause disruptions in the functionality of microcircuits and these microcircuits are quite expensive. For this, it is necessary to be able to fix the defect very quickly and automatically. The aim of the work is to create a software for classifying plates with a defect in real time, from which we can understand the cause of the defect. An algorithm is any well-defined sequence of computational operations, the input of which is a certain value or a set of values, and the result is a certain output value or a set of output values. In other words, an algorithm is a sequence of computational operations that transforms input data into output data. The algorithm can also be considered as a tool designed to solve a well-posed computational problem. The problem setting usually specifies the relationship between input and output data; an algorithm describes the exact process by which a relationship between input and output is achieved.

There can be many algorithms for solving the given problem and they can be so different from each other that one of them can solve the required problem more efficiently than the other. Therefore, it is important to choose the most efficient of the available algorithms. To evaluate the efficiency of

the algorithm, a sufficiently large amount of input data is given to the input of the algorithm in order to observe the growth rate of the algorithm performance time. The latter is also called the asymptotic efficiency of the algorithm. This means that we are interested in the finite performance time of the algorithm when the size of the input data tends to infinity. Usually, an asymptotically more efficient algorithm is more productive for all but very small input sets. The designations in estimating the asymptotic running time of algorithms use functions whose domain of definition is the set of natural numbers.

Conflict setting

These designations allow to estimate the worst-case performance time of the algorithm using a function defined only for natural numbers, which is the size of the input data.

θ — designation

For some function g(n), θ(g(n)) denotes the set of such functions:

$$\theta(g(n)) = \left\{ \begin{matrix} f(n)\text{: there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ numbers,} \\ \text{such as } 0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n) \text{ for all } n \geq n_0 \end{matrix} \right\} \quad (1)$$

$f(n)$ function f(n) belongs to the set θ(g(n)) if there are such positive constants c_1,c_2 that allow this function to include in the bounds $c_1 * g(n)$ և $c_2 * g(n)$ for sufficiently large n.

O — designation

θ designation determines the asymptotic upper and lower bounds of the algorithm. If for the given problem it is necessary to know only the asymptotic upper bound, then we can use the designation O: For some function g(n), O(g(n)) denotes the set of such functions:

$$O(g(n)) = \left\{ \begin{matrix} f(n)\text{: there exist positive constants } c \text{ and } n_0 \text{ numbers,} \\ \text{such as } 0 \leq f(n) \leq c * g(n) \text{ for all } n \geq n_0 \end{matrix} \right\} \quad (2)$$

O designation is used when it is necessary to show the upper bound of the algorithm performance with constant accuracy.

Ω — designation

In a similar way, as the designation O is intended to estimate the upper bound of the algorithm performance, the designation Ω determines the lower bound of the algorithm performance. For some function g(n), Ω(g(n)) denotes the set of such functions:

$$\Omega(g(n)) = \left\{ \begin{matrix} f(n)\text{: there exist positive constants } c \text{ and } n_0 \text{ numbers,} \\ \text{such as } 0 \leq c * g(n) \leq f(n) \text{ for all } n \geq n_0 \end{matrix} \right\} \quad (3)$$

For arbitrary 2 g(n) and f(n) functions $f(n) = \theta(g(n))$ only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$:

Decoding the input file. Translators

In the first phase of the program, the user imports an input file that needs to be decoded and rendered, specifying the necessary settings. That's why, it is necessary to implement a translator that will check the correctness of the input file, extracting from it the information necessary for the program in any format. For this purpose, in various problems, special software tools - interpreters - are used, in order to check the specific file or description of the given problem.

General structure of the project

It was appropriate to use the MVC (Model-View-Controller) design pattern [Straustrup, 450] to implement the project. It is a software design pattern that makes data processing independent of the user interface. Like all design patterns, MVC is only a

problem-solving principle, and modified versions of MVC can be used to solve each specific problem. The template consists of 3 main parts [Cormen, 637]:

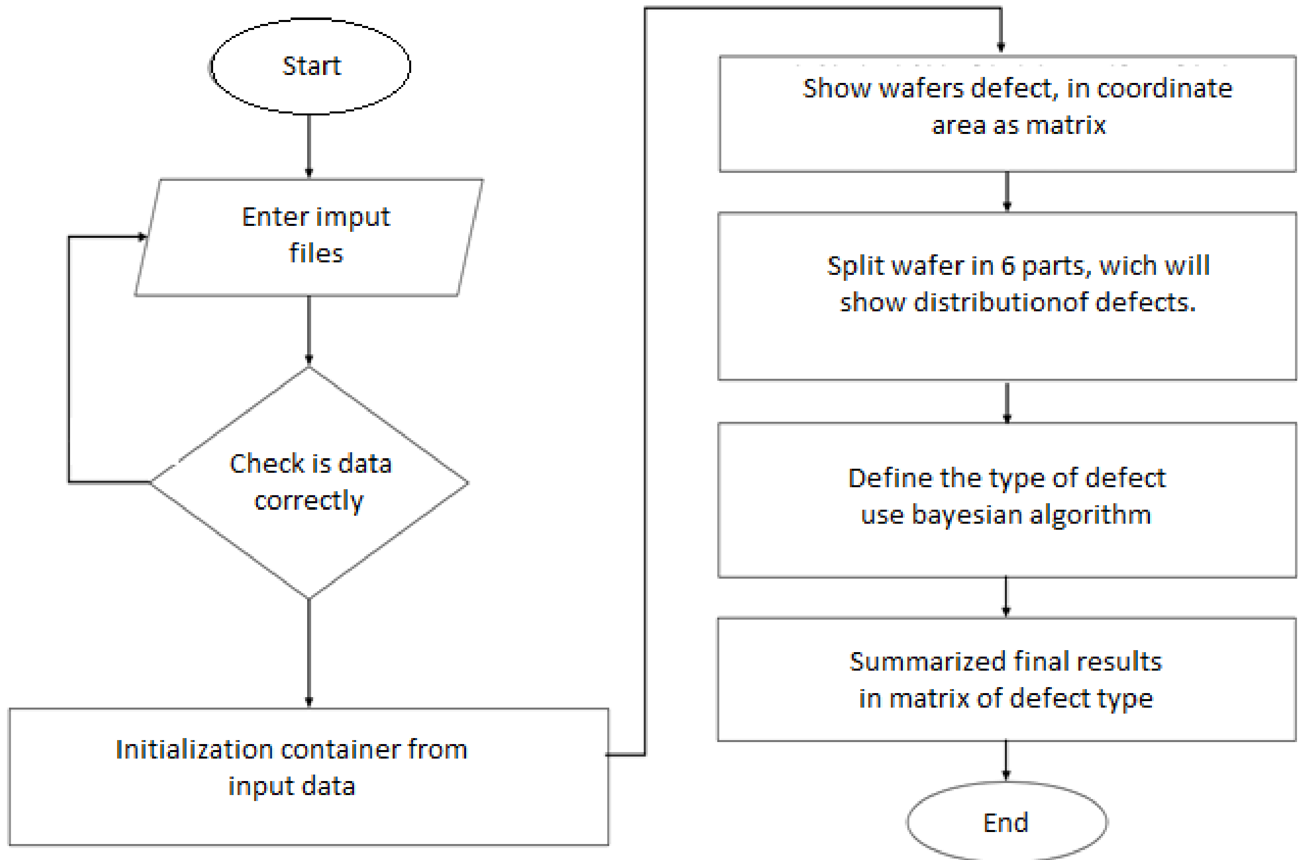
1. Model - provides program work with data, logic

2. View/Presentation - is responsible for building the appearance of the program with which the user works.

3. Steering part - provides a link between the model and the view.

Operation of the software algorithm. The block diagram of the algorithm of the processed software is presented below (Pic. 1).

Pic. 1



Block diagram of the processed algorithm.

The software tool starts its work by receiving data in the form of a file (Pic 2). This file contains the name of the Lot, the name of the wafer in this Lot and the coordinates of the corresponding defects. Then the availability and reliability of the necessary data are checked, because without them the program cannot work. If the requirements are met, the algorithm starts its work.

Pic.

```
Input.txt
1  Lot1 wafe1 (2,4) (4,4) (4,5)(4,6)(10,15)(11,16)(20,10)(22,11)(14,2)(18,9)(3,9) (3,10 ) (4, 8) (15,5):
2  Lot1 wafe2 (2,4) (3,8) (3,9) (3,10 ) (4, 8) (15,5):
3  Lot2 wafer1 (3,11) (20,9) (12,12) (10,15)(11,16)(20,10)(22,11)(14,2)(18,9):
4  Lot2 wafer2 (3,6) (4,9) (15,15):
5  Lot2 wafer3 (3,6) (4,9) (15,15):
6  Lot3 wafer1 (8,11)10,15)(11,16)(20,10)(22,11)(14,2)(18,9):
7  Lot3 wafer2 (3,6) (4,9) (15,15):
8  Lot3 wafer3 (8,11):
9  Lot3 wafer4 (8,8) (8,15) (12,7) (12,16) (19,8) (20,12):
10 Lot4 wafer2 (8,8) (8,15) (12,7) (12,16) (19,8) (20,12):
11 Lot5 wafer3 (8,8) (3,9) (3,8) (15,5) (12,5) (10,5):
12 Lot5 wafer6 (2,4) (3,8) (3,9) (3,10 ) (4, 8) (15,5):
13 Lot5 wafe8 (2,4) (3,8) (3,9) (3,10 ) (4, 8) (15,5):
14 Lot5 wafer9 (3,6) (4,9) (15,15):
15 Lot5 wafer10 (3,6) (4,9) (15,15):
16 Lot6 wafer1 (3,6) (4,9) (15,15):
17 Lot6 wafer2 (8,11):
18 Lot6 wafer3 (8,8) (8,15) (12,7) (12,16) (19,8) (20,12):
19 Lot6 wafer4 (8,8) (3,9) (3,8) (15,5) (12,5) (10,5):
20 Lot6 wafer5 (3,6) (4,9) (15,15):
21 Lot7 wafer2 (8,11):
22 Lot7 wafer3 (8,8) (8,15) (12,7) (12,16) (19,8) (20,12):
23 Lot7 wafer4 (8,8) (3,9) (3,8) (15,5) (12,5) (10,5):
```

Sample of an input file

The data from the input file is read with the help of the interpreter and after checking the accuracy of the data, it is added to the content of the Container [Stephen, 83]. After that, the wafer and the coordinates of its defects are displayed in the coordinate field in the form of a matrix. Then, the types of wafer defects are defined. These include: notch, straight line, oblique line, point defects, random scattered defects, evenly distributed defects, concentrated defects. The detection of these defects is performed in the coordinate field according to the coordinates of the given wafer defects. In addition, the wafer is divided into 6 parts.

The division of the wafer into parts is done in order to understand in which part the majority of defects in the wafer are concentrated. Through this division, we can understand the distribution of defects [Melikyan et al.,71], that is, if the coordinates of all the defects of the wafer are concentrated in only one of those 6 divided parts, then we can say that the wafer has a concentrated defect, and it is indicated in which part they are concentrated. Or, for example, in the case of an evenly distributed defect, all 6 divided parts contain the same number of defects. Thus, at the end, the Bayesian classification algorithm is applied, and through it the probability of each defect indicated in the wafer is calculated, and the type of defect with the highest probability is assigned to the given wafer.

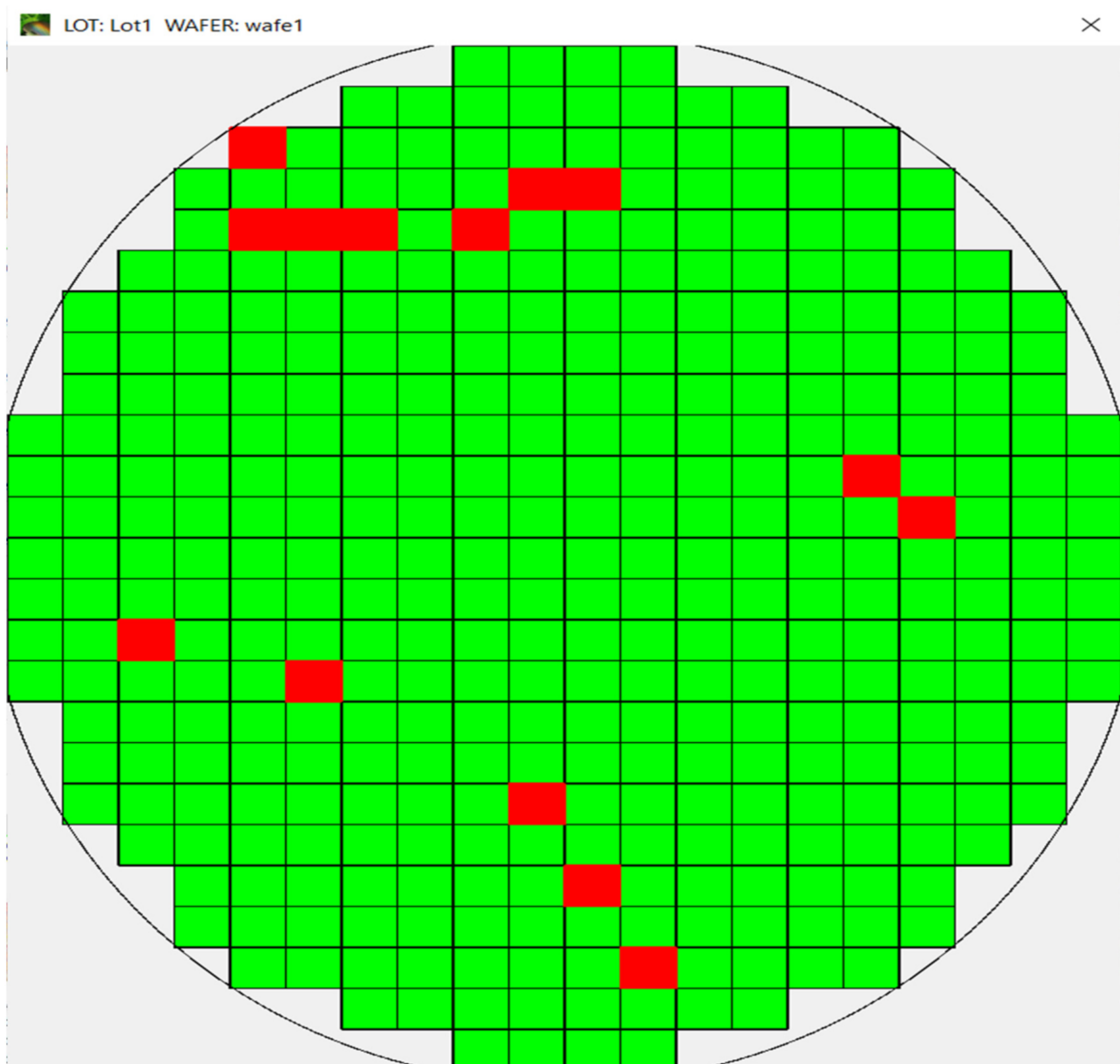
Thus, at the end, in the content of the Container, we have the names of the Lots and the corresponding wafers in them, with their names and the types of received defects. After that, these data are displayed in a table, which already represents the graphic part [Qt Online Documentation]

The provided graphic interface allows to see the final table obtained as a result of the work of the defect detection algorithm, where the names of the columns are the names of the types of defects, and the names of the Lot and the wafer containing the given defect are written in the cells of the table.

This table also helps us to understand where most of the defect coordinates are concentrated in the wafer.

In this graphic part, it is also possible to see the picture of the wafer we want with its defect coordinates. To see it, you need to click on the cell containing the name of our preferred Lot, wafer and click on “show” button below. The window of the following picture will open (Pic. 3).

Pic. 3



Wafer with its defect coordinates

Conclusion

It was processed and implemented an algorithm, a software tool, which enables to detect what defect the wafer contains. It was researched the necessary literature to write and implement a wafer defect detection algorithm. It was implemented a wafer defect detection algorithm and its optimized version, which are the basis of the program's work. The program provides a graphical interface, as well as tools for working with it, which contribute to the user's comfort. The work was carried out using the C++ programming language in the Qt environment for Windows and Linux operating systems. Furthermore, the proposed software tool is scalable and can be integrated into existing semiconductor manufacturing processes, making it a cost-effective solution for the industry. The tool can be used for both research and production purposes, enabling manufacturers to improve their understanding of semiconductor defects and optimize their manufacturing processes.

In summary, the proposed software tool has shown great potential for the semiconductor industry, as it offers an effective and efficient solution for detecting and classifying defects in semiconductor wafers. Future work could focus on further refining the tool's performance and expanding its capabilities to cover a broader range of semiconductor defect types.

References

1. **Melikyan V. Sh., A.G. Harutyunyan, A.A. Gevorgyan.** Methods of physical design of microelectronic circuits. 2015, p.173
2. **Stephen Prata**, C++ Primer Plus (Developers Library), Sixth Edition, USA, Addison-Wesley Professional, November 25, 2013, p.268
3. **Straustrup B**., The C++ Programming Language, Third Edition, AT&T ed, USA, 1997, p.571
4. **Thomas H. Cormen.** Introduction to Algorithms, Third Edition, The MIT Press, London, England, 2009, p.1037
5. Qt Online Documentation. 29.march.2023, https://doc.qt.io/

Ներկայացվել է՝
29.03.2023թ. Ուղարկվել է
գրախոսման՝ 05.05.2023թ.