International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 02, Issue – 04, April – 2017, PP – 01-06

FAULT CORRECTION IN IC HANDLER THROUGH IMAGE PROCESSING

L. MANOHARAN[1]
M. SIVA SHANKAR[1], P. SASI SEKAR[2], SHAN MOHAMED NOOH[3]
[1]*Asst. Prof, Department of Electronics and Communication Engineering,*
[2, 3, 4] *Student, Department of Electronics and Communication Engineering,*
JEPPIAAR SRR Engineering College, Padur, Chennai-603 103.

Abstract: Semiconductors are in almost every kind of electronic device imaginable. Each and every semiconductor chips undergoes extensive, complex and delicate testing process. To attain economy of scale, semiconductor products are tested in high volume. Furthermore, repair of finished semiconductor products is impractical. Therefore incorporating reliability at the design stage and reducing variation in during testing process have become essential. IC handlers are used to test IC's. Testing is performed in to check the quality of the manufactured product. A failed operation is possible due to misalignment or overlapping of IC during in pick and place operation in IC Handler. Orientation, spacing, arrangement and distribution become increasing interest in failure control. To avoid such failures we employ image processing models to identify the image patterns of the reference image. By comparing, reference image with the captured image during production, the parameters required for overall process control can be analyzed without affecting the process throughput. Reliable Hardware (High Speed Camera) becomes an important prerequisite.

Keywords: Image Capture, Image Comparison, Image Processing, Integrated Circuit Handler, Optical Character Recognition.

I. Introduction

The objective of the project is to meet the demand for more efficient production systems and product quality improvement, by eliminating human error. The proposed system requires following hardware and software prerequisite.

- To integrate an imaging hardware device to the existing system.
- Design an interface to compare reference image with the image captured.
- To perform Optical Character Recognition on the captured image.
- Program the Handler to rotate the IC based upon result of image comparator.

II. Block Diagram of System

The proposed system comprises of three modules,

- Module 1 - Imaging Capture and Optical Character Recognition (OCR).
- Module 2 - Image Comparator.
- Module 3 - Input to Handler.

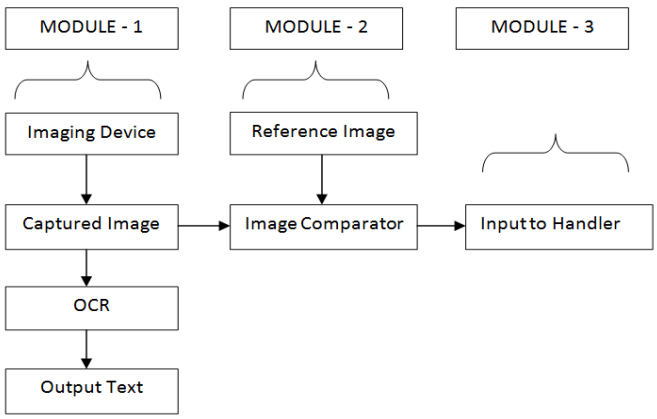

Fig 1 Block Diagram for the system

III. System Implementation

3.1. IMAGING CAPTURE

Imaging Device is used to capture and store the image in desired location. Then images are manipulated for image comparison. We use a simple program to access and use imaging device through visual Express. This program checks the availability of the device. If available it captures and displays the image. We use AForge.net Imaging library to interface the imaging device along with computer. AForge.Imaging contains image processing routine aimed at enhancement in computer vision tasks. The image capture Graphical Interface comprises of

- Two picture boxes – PictureBox1 and PictureBox2
- Three command buttons – Start, Capture and Save

The characters in IC are recognized through Tesseract Engine. Tesseract's output will have very poor quality if the input images are not preprocessed to suit it. So the images must be scaled up and skews must be corrected.

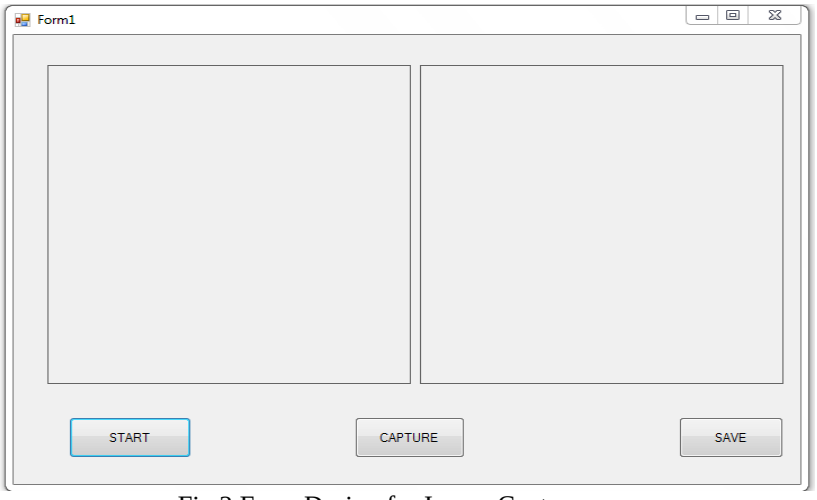

Fig 2 Form Design for Image Capture

3.2. IMAGE COMPARISON

Image comparison allow us to

- Find out the difference between images as a percentage value based on a chosen threshold value.
- Generate a difference image which shows where the images are different
- Get the difference as raw data to use for evaluation.
- The console version designed can take the paths of two images as parameters and return the difference as an error level.

The image compared is programmed using C# in Visual Studio express. If images were reduced to a much smaller size, and even gray scaled, then the differences would be both faster and simpler to find. Using .NET Framework the image is resized and gray scaled. We iterate through the pixels on both images comparing the two and then find out how many pixels were different. For each pixel, we calculate the difference of brightness value compared to other image's pixel value on the same spot and save it as a double array of bytes. Since RGB values are between (0 – 255), we count the entire double array which aren't zero, and divide this value by the amount of pixels in an image (256). We will have the difference value in percentage. In the RGB color space, brightness can be thought of as the arithmetic mean (μ) of the red, green, and blue color coordinates. The slight problem is that algorithm can also find differences which are not visible to naked eyes, so identical images with different resolutions can generate a lot of differences, making the algorithm blunt. Therefore, a threshold value is introduced which must be exceeded in order to be calculated.

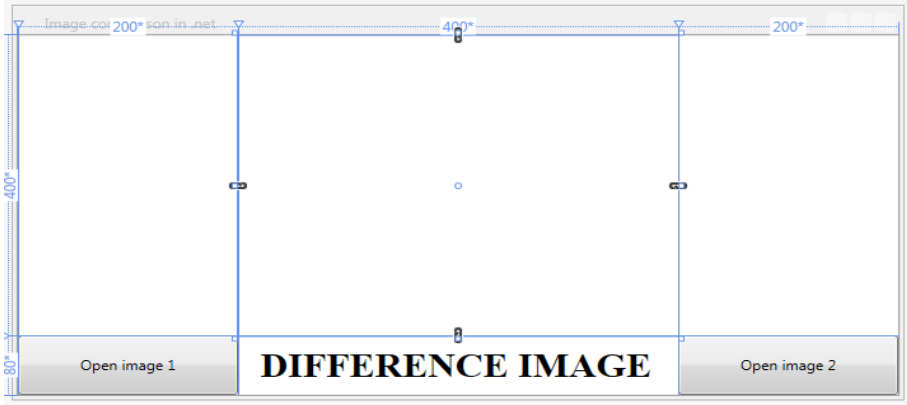

Fig 3 Form Design for Image Capture

The coding differentiates bitmap either by using a palette of black to pink corresponding to values 0 to 255 can find maximum value in palette map. This will enable to highlight small differences and keep them dark. This is achieved using the parameter adjustColorSchemeToMaxDifferenceFound ( ) as true or false.

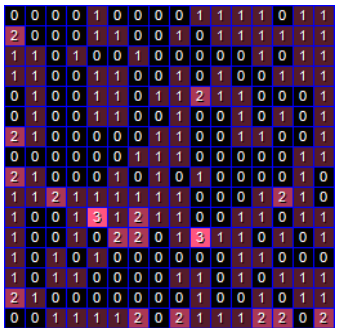

Fig 4 Visualizing the Difference

IV. Working Concept

Let us assume the four corners of reference image as PQRS and four corners of captured image as P'Q'R'S'. Based upon the results of image comparator, the rotator is aligned based upon the following algorithm.

- PP' - No rotation
- PQ' - 270 degree rotation clockwise
- PR' - 180 degree rotation clockwise
- PS' - 90 degree rotation clockwise

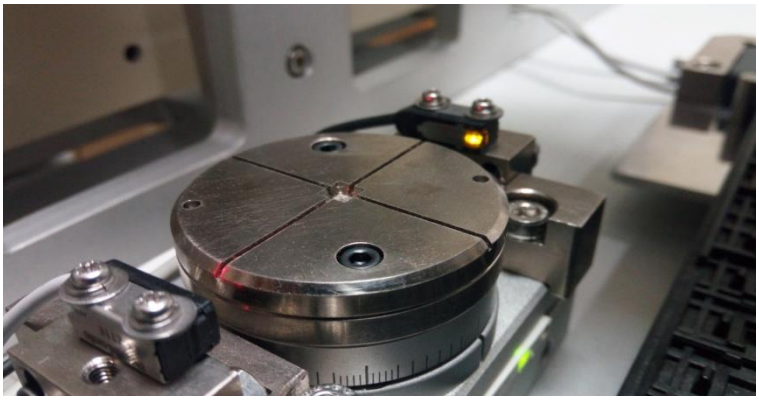

Fig 5 Rotator used to align IC

V. Result

The image capture and comparison is carried successfully and inputs are given to handler for deciding the rotator movement.

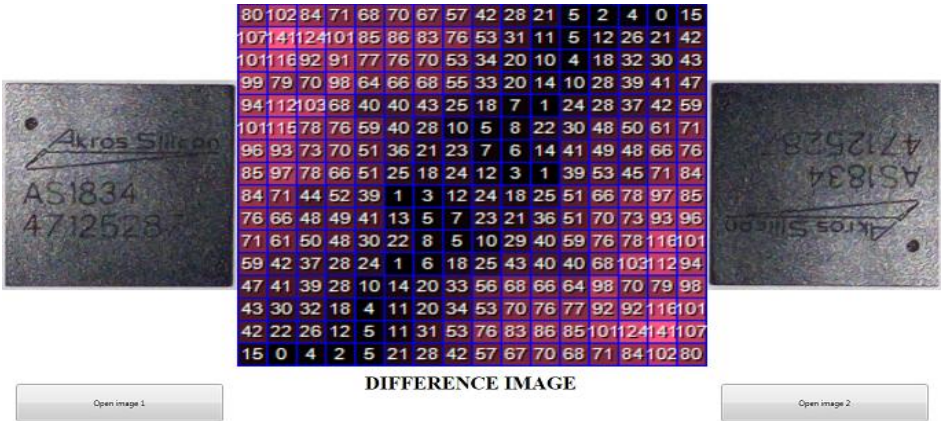

Fig 6 Image Comparison for PR'

Figure 7 shows the console version of image comparison interface. The histogram for pixels is displayed as output in Figure 8. The OCR output is displayed in Figure 9.

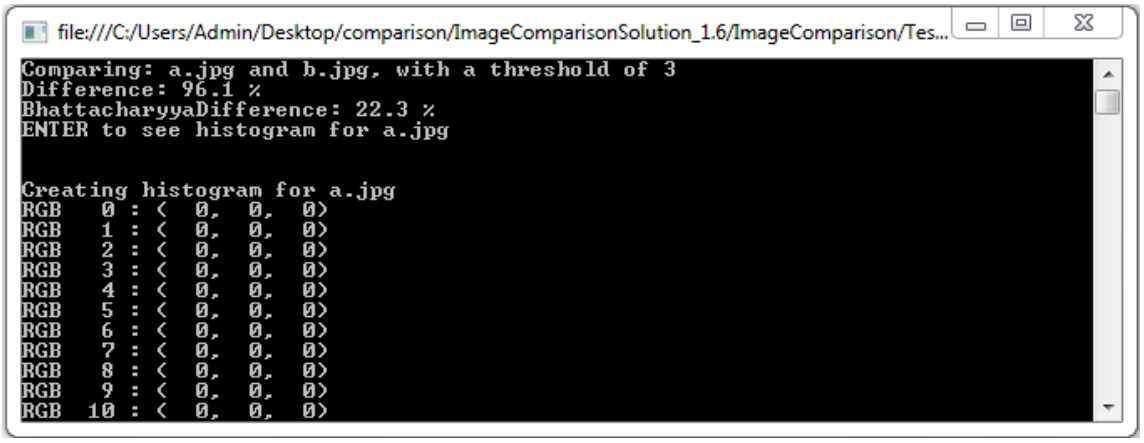

Fig 7 Console Version for Image Comparison

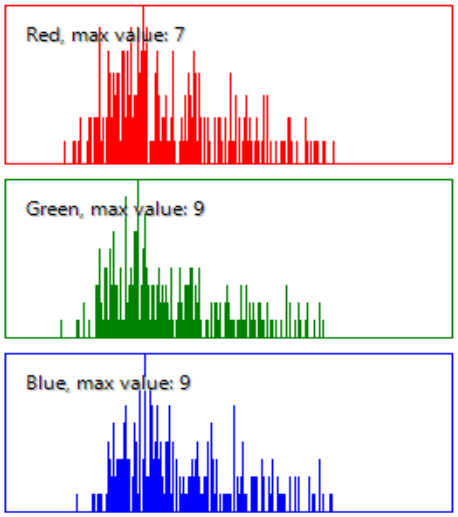

Fig 8 Generated Histogram through comparison

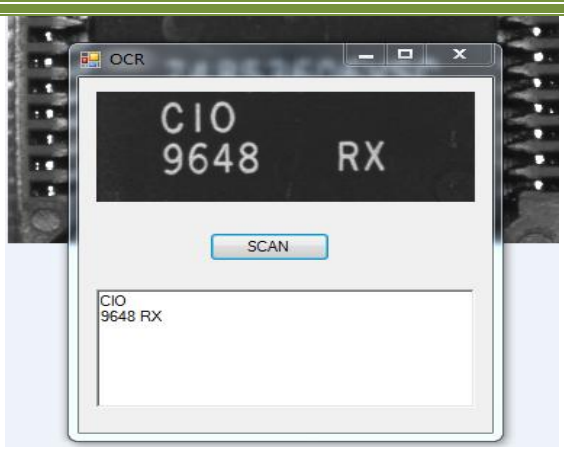

Fig 9 Optical Character Recognition

VI. Image Comparison Calculation

Total Difference = $\left(\frac{diffRed}{maxDiff} + \frac{diffGreen}{maxDiff} + \frac{diffBlue}{maxDiff}\right)/3$

diffRed = $R^2 - histR^2$

diffGreen = $G^2 - histG^2$

diffBlue = $B^2 - histB^2$

The RGB values are obtained through RGB calculator and the histR, histG, histB values are obtained through the console version of image comparator.

DATA FOR REFERENCE IMAGE

X co-ordinate = 478
Y co-ordinate = 68
R' = 226
R = append (R'G') = 451
G' =225
B = append (R'B') = 457
B' = 231
G = append (G'B') = 456
[histR, histG, histB] = [0, 0, 0]
maxDiff = 512
Substituting above values we calculate difference value for reference image as
Difference for Reference image = $\left(\frac{diffRed}{maxDiff} + \frac{diffGreen}{maxDiff} + \frac{diffBlue}{maxDiff}\right)/3 = 403$

DATA FOR CAPTURED IMAGE

X co-ordinate = 478
Y co-ordinate = 68
R' = 197
R = append (R'G') = 393
G' = 196
B = append (R'B') = 398
B' = 201
G = append (G'B') = 397
[histR, histG, histB] = [0, 0, 0]
maxDiff = 512
Substituting above values we calculate difference value for captured image as
Difference for captured image = $\left(\frac{diffRed}{maxDiff} + \frac{diffGreen}{maxDiff} + \frac{diffBlue}{maxDiff}\right)/3 = 305$

The total difference is calculated as difference between values of Reference image and captured image
TOTAL DIFFERENCE = 403 – 305 = 98

Similarly the values are generated for every pixel and are normalized through array comparer and finally the intensity values are displayed in the bitmap image along with histogram generation. A practical method of comparison uses a combination of enhancement and analysis steps to distinguish variation defects between reference image and captured image.

VII. Conclusion and Advantages

We have analyzed how image processing techniques employed can improve the system throughput. High resolution images have always played an important role in learning, but today these images can provide whole new significance.

The advantages of the designed system are,

- Detection and correction of misalignment or overlapping of die during pick and place operation.
- Hardware mechanism devised can improve system throughput and reduce time consuming steps.
- Improved reliability in the system testing operation.
- Jamming ratio in the handler device can be reduced.
- Automated and self-organized production can be achieved based on extensive data collection and information.

VIII. Reference

[1] Sheau-Chyi Lee, Serge Demidenko, Kok-Hua Lee, IC Handler Throughput Evaluation for Test Process Optimization, IEEE Instrumentation & Measurement Technology Conference, May 2007.

[2] Anna Fabijanska, Marcin Kuzanski, Dominik Sankowski, Lidia Jackowska-Strumillo, Application of image processing and analysis in selected industrial computer vision systems, International Conference on Perspective Technologies and Methods in MEMS Design, July 2008.

[3] L.Suriya Kala, Dr P. Thangaraj, Conversion of Image into Text Using Tesseract OCR Engine, International Journal of Computer Science Engineering and Technology, Vol 4, Issue 12,347-348, December 2014.

[4] Xiaojie Guo, Yu Li, Haibin Ling, LIME: Low-Light Image Enhancement via Illumination Map Estimation, IEEE Transactions on Image Processing, Volume 26, Issue 2, Pages: 982 – 993, December 2016.

