Automatic Wafer Level Testing of Optical Chips

Jaap Goldhoorn, *Student, EE*

***Abstract*—Automated wafer level testing could provide a huge time saving in testing optical chips. Newly developed parabolic micromirrors could aid in making this possible for a wide spectrum of wavelengths. This thesis explores the viability of creating an automated setup equipment using equipment that is available at the photonics lab at the University of Twente. We find that alignment should be reasonably possible. Using computer vision to aid in alignment poses a larger challenge.**

I. Introduction

Testing optical chips is traditionally quite time consuming. This is due to several factors. One is that light has to be coupled into the side of the chip. This means that each chip has to be cut out of a full wafer in order to be tested. Additionally, each chip has to be seperately aligned by hand.

Automatic whole wafer testing has been developed in several places. Most of these rely on so called grating couplers, which are made to couple light in or out of the chip by means of phased diffraction [4]. These methods rely on very specific wavelengths to operate correctly.

The Integrated Optical Systems (IOS) group at the University of Twente (UT) has developed 3D printed micromirrors that can be integrated on a chip to allow coupling of a wide band of wavelengths [2].

This thesis will explore challenges in building a setup that should be able to automatically couple light into a wafer equipped with these mirrors.

II. Existing Setups

It's useful to look at what setups already exist in the world. While these differ substantially from the setup that we propose, it can be benefitial to compare to the status quo. We will look at one setup proposed in [1] and a collection in [5].

One thing all of these have in common is the use of grating couplers. These differ from micromirrors in that the light has to be coupled in from an angle rather than straight down.

Alignment in these systems is done in two different ways. One is by a spiral movement that slowly closes in on the point of maximum brightness. The other is gradient ascent. This is a method of finding the maximum by moving the probe a tiny amount around the current position, then taking a larger step in the direction of the maximum gradient.

All setups couple light into the wafer using a bare fiber. This requires tip of the the fiber to be within one micron of the wafer. To achieve this without touching the wafer, very accurate distance sensing is necessary.

III. Description of Proposed Setup

The proposed light coupling setup can be seen in figure 2. To avoid the requirement of having accurate distance sensing, a focussing system using parabolic reflectors is proposed. The aim is to obtain focus (maximum coupling) by changing the height of the mirror above the wafer. This can be done without risk of damaging the wafer as a larger distance can be maintained at all times.

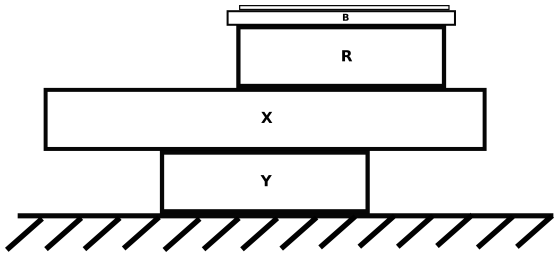


Fig. 1: Schematic side view of the proposed setup. Two orthogonal linear stages (X,Y), one rotation stage (R), and a base to hold the wafer (B)

Translation and rotation of the wafer is achieved by two linear stages on a 90 degree angle as seen in figure 1. We will call these x and y. x has a total range of about 50 mm, while y has a range of about 100 mm. A third rotating stages is placed on top to achieve rotation. The stepsize of the linear stages was measured by measuring the total distance over a large number of steps. The precision of x was measured to be $(25 \cdot 10^3)/(50 \cdot 10^3) = 0.5[\mu\text{m/step}]$ and that of y at $(78 \cdot 10^3)/(500 \cdot 10^3) = 0.156[\mu\text{m/step}]$.

IV. Translation Table

As discussed in the previous section, movement of the wafer is achieved using two orthogonal linear stages and a rotation stage. Each degree of freedom is controlled by a stepper motor. The stepper motors are controlled by an Arduino program that communicates over a serial connection.

A. Ramp Up Speed

The motors in the table cannot run at their maximum speed immediately. Rather the speed should be linearly increased and decreased at the start and end of each movement as in figure 3. Since the speed should increase linearly, the time between steps should increase with some square root relation. To have more control over the exact ramp up and ramp down time, we will work out the exact relation from control parameters

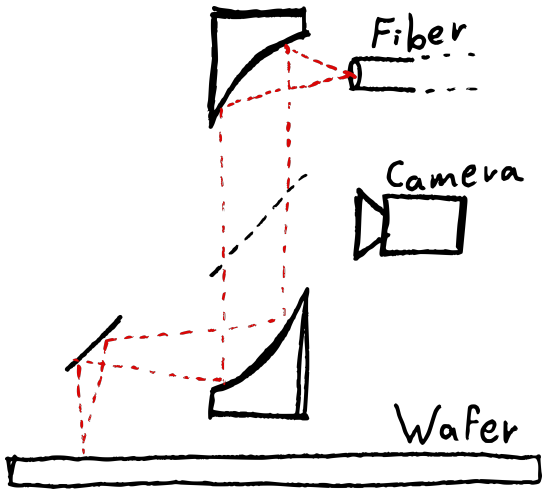


Fig. 2: Schematic vision setup. The fiber is focussed by two parabolic mirrors. A semi-transparent mirror is used for imaging to a camera.

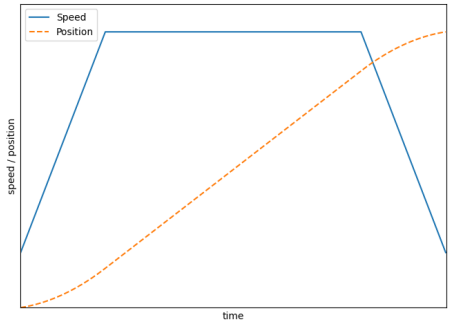


Fig. 3: Speed and resulting position of the translation table with respect to time. Both quantities are normalized and not to scale.

$v_{\text{min}}[\text{steps}/\mu s]$, $v_{\text{max}}[\text{steps}/\mu s]$, and $T_{\text{ramp}}[\mu s]$. For convenience we will also introduce the parameter $v_{\text{diff}} = v_{\text{max}} - v_{\text{min}}$. We could also use the step delay instead of the speed: $\tau_{\text{max}} = 1/v_{\text{min}}$ and $\tau_{\text{min}} = 1/v_{\text{max}}$

The speed in seps per second as a function of time is given by:

$$v_{(t)} = v_{\text{min}} + \frac{v_{\text{diff}}}{T_{\text{ramp}}} \tag{1}$$

The position in steps is easily derived by the integral. We will treat this as a continuous system for now as the analysis is easier and it will not influence the final result.

$$n_{(t)} = \int v_{(t)} dt = v_{\text{min}} t + \frac{1}{2 \cdot T_{\text{ramp}}} v_{\text{diff}}\ t^2 \tag{2}$$

What we want to know is the time at which each step is taken. Thus we need to reverse the equation using the quadratic formula.

$$t[n] = \frac{-v_{\text{min}} \pm \sqrt{v_{\text{min}}^2 + \frac{4n \cdot v_{\text{diff}}}{2T_{\text{ramp}}}}}{\left(\frac{v_{\text{diff}}}{T_{\text{ramp}}}\right)} \tag{3}$$

The $+$ of the $\pm$ will always evaluate negative so this solution can be discarded. With some rewriting, we end up with the solution.

$$t[n] = \frac{-T_{\text{ramp}}}{v_{\text{diff}}} \left(v_{\text{min}} - \sqrt{v_{\text{min}}^2 + 2n \cdot \frac{v_{\text{diff}}}{T_{\text{ramp}}}} \right) \tag{4}$$

The control of the table uses timedeltas rather than absolute time, so we need to find the derivative with respect to n. This results in.

$$\Delta t[n] = \frac{1}{\sqrt{v_{\text{min}}^2 + 2n \cdot v_{\text{diff}}/T_{\text{ramp}}}} \tag{5}$$

Now we need to know the number of steps N to reach this maximum speed. This is achieved by simply plugging T_{ramp} into equation 2.

$$N = n_{(T_{\text{ramp}})} = \left(\frac{v_{\text{diff}}}{2} + v_{\text{min}} \right) T_{\text{ramp}} \tag{6}$$

The speed has to ramp up to a maximum. The ramp down might also be different to the ramp up. When taking M steps, the delay at the n^{th} step can be described by.

$$\max\left[\ \tau_{\text{min}}\ ,\ \Delta t_{\text{up}}[n]\ ,\ \Delta t_{\text{down}}[M-n]\ \right] \tag{7}$$

This could be directly implemented into a microcontroller using the description above. It is however quite inefficient to do so as a lot of expensive floating point operations need to be executed. Thus an implementation using Lookup Tables (LUT) is preferred. If the total steps taken is larger than the comined lenghth of both LUTs, the implementation is fairly simple as the ramps don't cross. When the total number of steps is smaller, the algorithm should switch from the upLUT to the downLUT at the point that their delays are equal. The meetpoint can be saved in a seperate LUT.

Algorithm 1 Find the delay as a function of the step number.

```
procedure GETDELAY(stepNumber, totalSteps)
    if totalSteps > |upLUT| + |downLUT| then
        if stepNumber < |upLUT| then
            return upLUT[stepNumber]
        else if totalSteps - stepNumber < |downLUT| then
            return downLUT[totalSteps - stepNumber]
        else
            return minimumDelay
    else
        if stepNumber < meetStepLUT[totalSteps] then
            return upLUT[stepNumber]
        else
            return downLUT[totalSteps - stepNumber]
```

If the up ramp is equal to the down ramp, only one LUT is required and the meetpoint will always be exactly in the middle.

B. Rotating the plane about an arbitrary point

The final setup thould be able to rotate the plane around any point. The rotational stage will be attached on top of the translation stages. This means that rotations will take place around some point that will change with the movement of the translation stages. We will define this point as $\mathbf{m}$. To achieve rotation around an arbitrary point, we will first define the standard rotation r of a point $\mathbf{p}$ by angle θ around midpoint $\mathbf{m}$.

$$r_{(\mathbf{p},\theta,\mathbf{m})} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} (\mathbf{p} - \mathbf{m}) + \mathbf{m} \tag{8}$$

Now we will define an arbitrary midpoint as $\mathbf{m}'$. A rotation around this arbitrary point will rotate straight lines by the same angle as a rotation around $\mathbf{m}$, but with an added translation. This can be corrected by bringing $\mathbf{m}'$ back to its original position. In other words, translating by $\mathbf{m}' - r_{(\mathbf{m}',\theta,\mathbf{m})}$.

So the steps to rotate the plane about an arbitrary point $\mathbf{m}'$ are as follows.

- Rotate the stage by θ.
- Translate by: $\mathbf{m}' - r_{(\mathbf{m}',\theta,\mathbf{m})}$

If one wants to keep the midpoint at an approximately static position during this rotation, θ should be divided into small intermediate steps and both actions should be evaluated at each of these. To achieve only the end position of the rotation, both actions can be performed in any given order and speed.

C. Bringing an object to another position

Now let's consider an object who's position and rotation is defined by the 2-tuple $(\mathbf{p}, \theta)$. This is how we will later define rectangles for text detection. If we want to bring this object to the position $(\mathbf{p}', \theta')$, we can do this by a simple translation of $\mathbf{p}' - \mathbf{p}$ combined with the corrected rotation described in the previous section.

- Translate by $2\mathbf{p}' - \mathbf{p} - r_{(\mathbf{m}',\theta,\mathbf{m})}$.
- Rotate by $\theta' - \theta$

V. Computer Vision Methods

To recognize the position on the wafer, we will try to make use of text markings already present on the wafer. If successful, the recognized text and position can be compared to known markings along with their desired position.

Identifying text is perfomed in two steps. These are text *detection* and *recognition*. The first is meant to show you where there is text in the images, without being concerned what that text is. The second is used to convert the previously identified text regions into strings.

A. Text Detection

To detect the location of text inside the images, we will be using the neural network presented in the paper *EAST: An Efficient and Accurate Scene Text Detector* [6]. This network can be imported into Python using openCV.

1) Network Input: To input images into *EAST*, a they have to be converted into a slightly different format.

EAST only accepts images where all dimensions are multiples of 32. This can be achieved by cropping off the right and bottom sides of the captured image.

Images imported using *openCV* are arrays in the shape (height, width, channels). *EAST* expects them to have the shape (1, channels, height, width).

Another difference is *EAST* uses RGB color channels while *openCV* uses BGR. This should not matter in our case as all images are in greyscale, but we will apply this transformation for safety.

Finally, the network is trained using images where the mean value of the image is subtracted from each pixel value. Therefore, this should also be done to the images that are fed into it.

2) Filtering: The network ouputs a matrix of boxes along with confidences between 0 and 1. A lot of these will be false positives, which can be filtered by removing all boxes with a confidence lower than a certain threshold. This threshold can be fine-tuned to some extent, but since most confidences can be expected to be close to either 0 or 1, the return on investment is rather limited. After this, we will still be left with multiple boxes over each piece of detected text. The multiplicity can be reduced with a method called nonmaximum suppression (NMS). NMS removes similar rectangles by comparing them using some measure of similarity. The most common is Intersection over Uniun (IoU), which is formally defined in equation 9 and visually illustrated in figure 5. For region, this is the area of overlap over the joint area of the regions. An IoU of 1 means that the rectangles are exactly equal, while an IoU of 0 means that there is no overlap at all.

$$\text{IoU}(A, B) = \frac{A \cap B}{A \cup B} \tag{9}$$

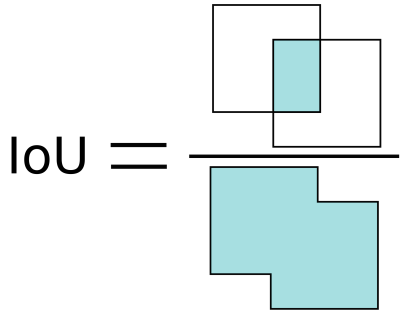
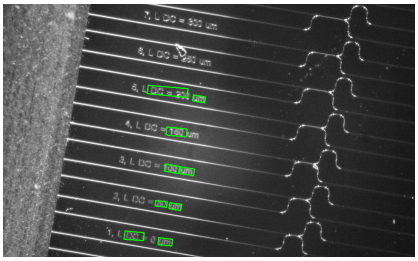


Fig. 5: The IoU is defined as the ratio between the filled regions.

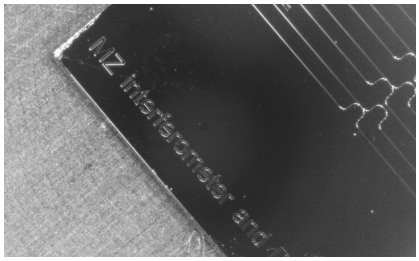
The NMS algorithm works as follows:

1) Take the box with the highest confidence from in the list.
2) Remove all boxes from the list where the IoU with the selected box exceeds some threshold.
3) Repeat steps 1 and 2 with the next highest confidence.



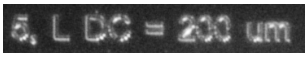
(a) Small text partially detected

(b) Large text partially detected

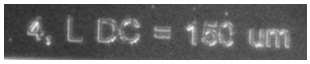


(c) Text undetected

Fig. 4: Text detection results

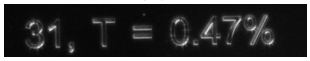


(a)



(b)

(c)



(d)

Fig. 6: Extracted bits of text used for text recognition.

NMS has standard implementations in several different Python libraries. We will use the one in *openCV* as we already use this.

B. Text Recognition

For text recognition, we will use *Tesseract* [3]. This is a well established library with a good Python API. The rectangles identified by EAST can be used to extract from the image and converted to text by Tesseract.

C. Results

1) Detection: Figure 4 shows some results of the text recognition algorithm, including NMS. While there are few false positives, a large portion of the detected boxes is offset and/or off angle.

2) Recognition: The snippets in figure 6 are some of the ones that were used to test the viability of the text detection engine. Apart from a few single characters, the results were mostly nonsensical. Notably, the images with more grain resulted in a lot more false positives, while the less grainy images often resulted in very few, if any, recognized characters.

VI. Conclusion

Automated wafer testing proves to provide many challenges. While a final setup should be feasible without large investments, it will be severely limited in its capabilities.

Computer vision methods could be used reliably, but are non-trivial. A larger effort could see these become viable.

VII. Future work

To get viable automatic coupling working a few more things have to be done.

First, the proposed focussing system has not been built yet, and should thus be tested.

The translation and rotation setup is partially built and functional. However it is still missing a number of critical features. Most notably, the rotation stage and the base to hold the wafer are still not built. The first should be fairly trivial to add as it can use the same controller box as the other two stages with some extra code. The second needs some more work as there is no clear proposed design yet.

The computer vision methods are far from viable at this point. If this method is to be explored further, it would be a good idea to look into methods of training the network for this particular application. Additionally, the text detection and recognition might be drastically improved by using a more advanced imaging system. Otherwise, it might be more sensible to forego this method altogether and use a ”blind” system where initial coupling is performed simple sweep motion. It should be possible to place a wafer fairly accurately, so the search should be fast enough.

The different components of this setup should be integrated into a full stack.

Acknowledgment

Thanks to my supervisor, Lantian Chang, for giving me the oppurtunity to work on this project. Thanks to Kenan Niu for helping me out with little bits of advice throughout the project. Thanks to Sonia Garcia Blanco for taking the time to be on my committee. I would also like to thank Jeroen Korterik for showing me the ropes in the lab and putting substantial time into helping me out with the setup.

References

[1] Scott Jordan. Photonomics: automation approaches yield economic aikido for photonics device manufacture. *Optomechanical Design and Engineering 2002*, 4771:49, 9 2002.

[2] Yujia Kong. 3d printed on-chip parabolic mirror for chip-to-fiber and chip-to-chip coupling.

[3] R Smith. An overview of the tesseract ocr engine. volume 2, pages 629–633, 2007.

[4] Yun Wang, Jonas Flueckiger, Charlie Lin, and Lukas Chrostowski. Universal grating coupler design. *https://doi.org/10.1117/12.2042185*, 8915:284–290, 10 2013.

[5] Peng Zhang, Bo Tang, Yan Yang, Bin Li, Ruonan Liu, al Peng Zhang, Ling Xie, Zhihua Li, and Fujiang Lin. Test system for wafer-level silicon-photonics testing. *https://doi.org/10.1117/12.2573456*, 11548:294–301, 10 2020.

[6] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: An efficient and accurate scene text detector. 4 2017.