EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Department of Electrical Engineering

# Test Fixture Optimization

*Master Thesis*

F.H. Oudman

PRODRIVE TECHNOLOGIES

Supervisors:
dr. R.H. Mak
ir. M.G.M. Spierings
prof.dr. H. Corporaal

Eindhoven, Thursday 11th July, 2019

# Abstract

At Prodrive, the Automatic Electronic Test system (AET) is used to test Printed Circuit Board Assemblies (PCBAs). In an AET test, the DUT is placed on a set of test probes which can test the DUT using electric stimuli. In order for the test probes to make good contact with the DUT, the DUT is clamped between a so-called top fixture and bottom fixture. The bottom fixture holds the different probes, while the top fixture contains different push fingers that force the DUT down on the test probes. Because of non-alignment between push fingers and test probes, the DUT can deform. Strain created by this deformation can lead to DUT damage. Hence, push fingers need to be placed in such a way that board strain stays below a safe threshold, while at the same time the fixture costs are minimized.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

bladiebla

# Chapter 2

# Device Under Test

When a printed circuit board assembly (PCBA) is being tested, it is referred to as the device under test (DUT). This way, one can easy distinguish between the tested PCBA and some possible other PCBA that is not being tested but is needed in the setup for some other reason.

In this chapter, an introduction is given on the different components of a PCBA. Next, the different components of a fixture are described. Finally, the testing process is described, where the DUT and fixture interact. The strain that arises during this testing process is further described in the next chapter.

## 2.1 Printed circuit board assembly

The base of every PCBA is of course the PCB. Such a printed circuit board is made from woven glass fibres and resin, together forming the board. On this board, a thin copper foil is attached. By precisely removing parts of this copper, one can use the remaining copper as an electrical network which provides connectivity between the different components that will later be placed on the PCB. To accommodate more difficult routing, multiple copper and resin layers can be added to create the desired electrical network.

By soldering different components on the board, the PCB becomes a PCBA. One can classify components into two categories: through-hole technology (THT) and surface-mount technology (SMT). When using THT, small holes are drilled at PCB contact point locations, through which the leads of trough-hole components are inserted. At the opposite side of the board, the lead is then soldered to the board. In surface mount technology (SMT), a components is soldered directly to a soldering pad on the PCB.

When a PCB deforms due to fixture clamping, components react different depending on the used technology. In case of THT, component leads provide a bit of flexibility, thus providing some damage-free bending freedom. In case of SMD, the different components add extra stiffness to the PCB. This extra stiffness reduces strain directly under the THT component, but can create extra strain at the soldering connections at the edges of the component. This strain at the edges has a higher change of damaging the PCB or the component since their is no component lead flexibility.

Each PCBA needs to have a rectangular shape of which the length and the width have minimum and maximum dimensions to be used in an AET test. In case one needs to produce a PCBA that is smaller than the minimum PCBA size, and/or in case one needs to produce a non-rectangular shaped PCBA, one needs to apply panelization. Using panelization, one fits multiple PCBA's on a single PCBA. During the PCB manufacturing, cutouts are created around the sub-PCBA's. In order to hold the sub-PCBA's in place, small breakaway tabs are retained. Figure 2.1 shows an example of a panelized PCBA. After the PCBA is fully assembled and tested, the different sub-PCBA's are separated from the main board by milling away the breakaway tabs.

Since the different sub-PCBA's are connected to the main PCBA only through small tabs, one needs to be careful with push fingers, test probes and/or support probes around the breakaway

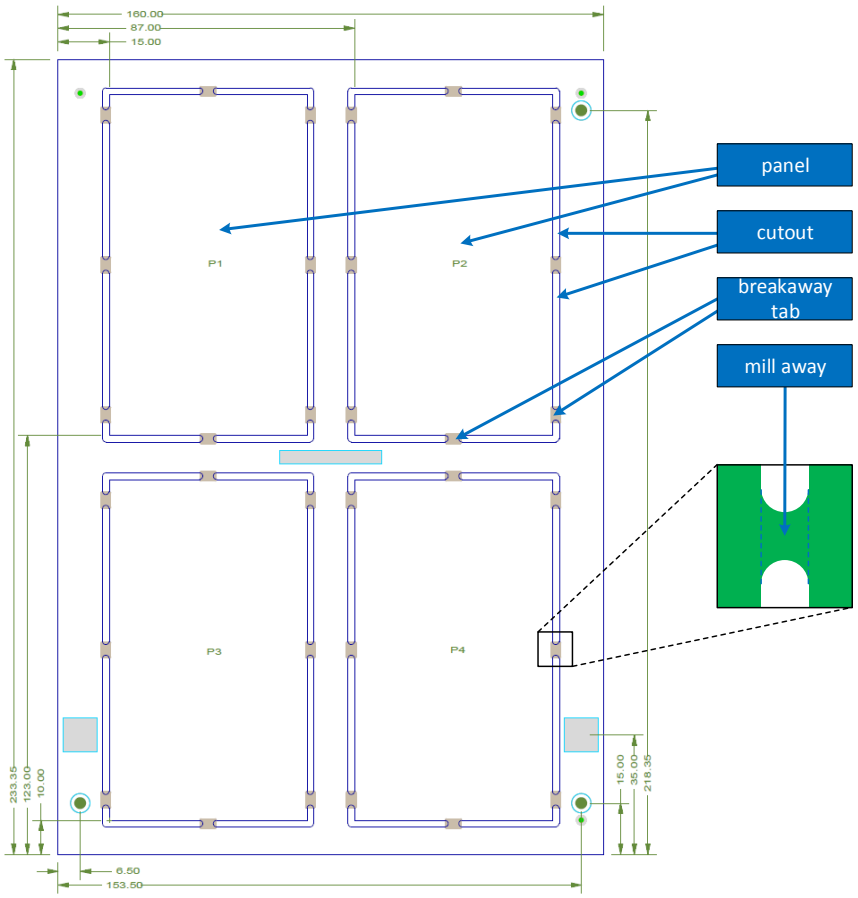



Figure 2.1: Example panelized PCBA

tabs, to prevent excessive strain at these locations.

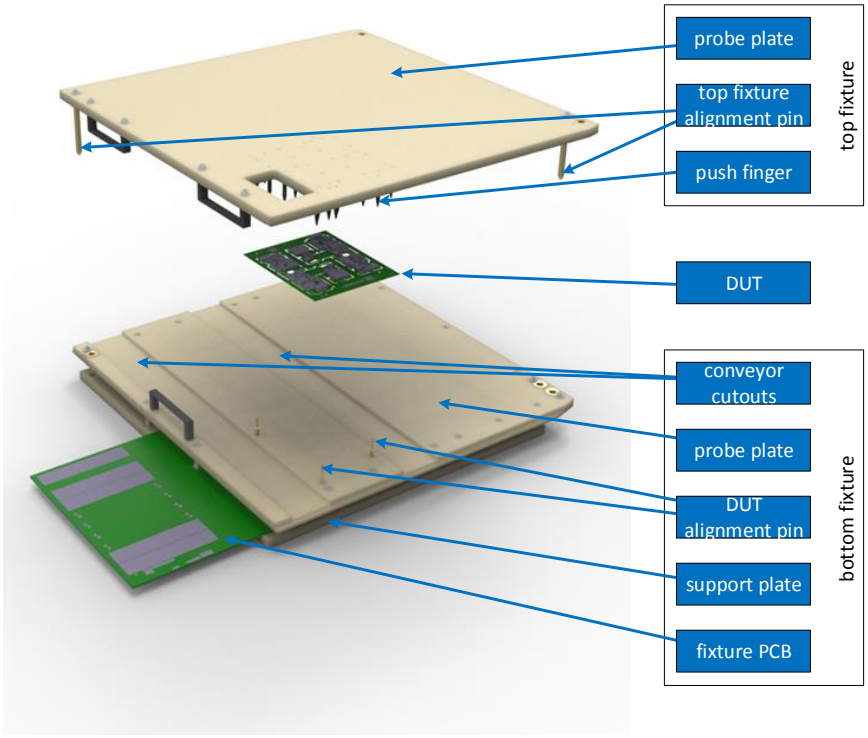



Figure 2.2: The different modules of the AET Inline system

## 2.2 Fixture

During the testing process, the DUT is clamped in the fixture. Such a fixture consists of two main parts: the top fixture and the bottom fixture. Between these two main parts, the DUT is clamped. Figure 2.2 presents a more detailed drawing of this.

### 2.2.1 Bottom fixture

The bottom fixture holds the different test probes that will test the DUT. Furthermore, the bottom fixture contains some support probes, that provide additional support for the DUT during the resting stage. Three alignment pins are used to align the DUT with the bottom fixture.

### 2.2.2 Top fixture

The top fixture consists of a plate with push fingers mounted onto it. These push fingers push the DUT downwards onto the different probes of the bottom fixture. Furthermore, it contains alignment pins to align the top fixture with the bottom fixture.

## 2.3 Testing process

This section describes the execution of an example AET test. Figure 2.3 shows a schematic drawing of the start situation. The top fixture and bottom fixture are separated. In the middle, conveyor belts bring in the DUT. One of these belts is located to the front of the machine, a second is located towards the back side. The front and back edges of the DUT are resting on these conveyor belts. When the DUT is fully shifted into the machine, it is stopped by a small stopper.

Next step is the lowering of the conveyor. The DUT is aligned with the bottom fixture through three alignment pins. The DUT rests on multiple support probes such that there is not yet any contact between DUT and test probes. Figure 2.4 shows a schematic drawing of this situation.

Next, the top fixture is lowered. First, alignment pins in the top fixture and alignment bushes in the bottom fixture allow the top fixture to align itself with the bottom fixture and therefore with the DUT. Figure 2.5 shows a schematic drawing of this situation. After alignment, the top fixture continues with the lowering, pushing down the DUT with the push fingers attached to the top fixture. This way, the DUT makes contact with the test probes. This is shown in Figure 2.6.

Next, the different electrical tests are performed trough the test probes. After all testing is completed, the top fixture is raised again, followed by the raising of the conveyor belts, followed by the conveyor belts shifting the DUT out of the AET system.

These four steps describe the most basic AET test procedure, but different extensions are possible. One commonly used extended procedure is the so-called two-stage AET test. In such a test, the bottom fixture contains two different types of test probes. In the first stage, the DUT is pushed down to make contact with all probes. In the second stage, the DUT is raised a bit, such that only a subset of the bottom fixture test probes makes contact with the DUT.

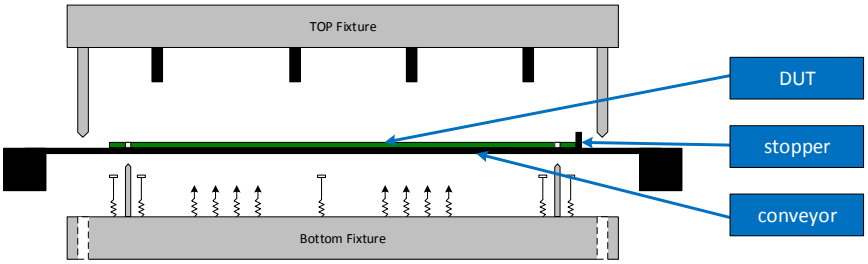



Figure 2.3: The DUT brought into the AET

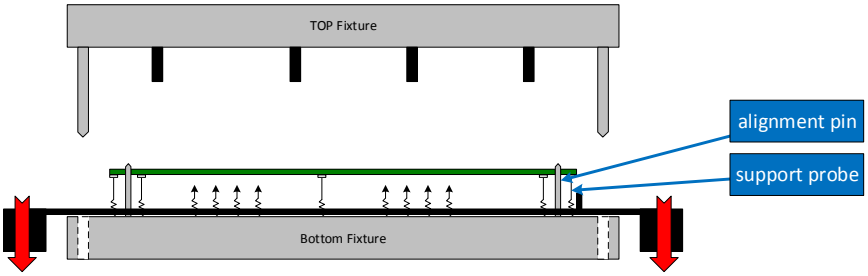



Figure 2.4: The DUT aligned with the bottom fixture

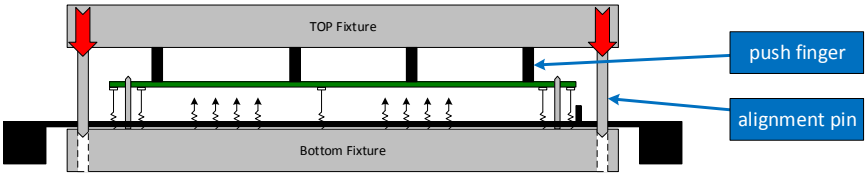



Figure 2.5: Alignment of top fixture with the bottom fixture

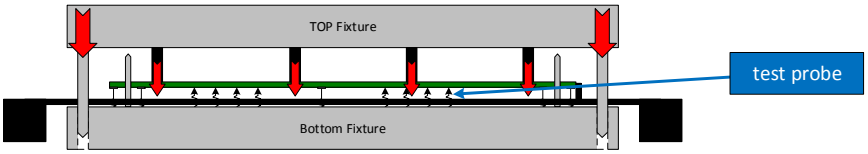



Figure 2.6: Establishing contact between DUT and test probes

# Chapter 3

# Board strain

bladiebla

## 3.1 Definition

principal / mohr's circle / Von Mises

## 3.2 Board defects

example defects / literature studies

## 3.3 IPC/JEDEC guidelines

guidelines overview

## 3.4 Finite Element Analysis

fea characteristics

# Chapter 4

# Problem statement

bladiebla

## 4.1 State-of-the-art

context

## 4.2 Objective

problem statement

## 4.3 Project scope

scope

# Chapter 5

# Optimization algorithms

Optimization is done in an iterative manner. Two main parts can be distinguished: the optimization controller and the optimization step.

An optimization step is an algorithm that performs exactly one step: given an existing fixture design, create a derived design that is hopefully better. An example is the random step algorithm, which randomly adjusts all support probe and push finger locations by a bit. Note that such an optimization step does not necessarily result in a better/optimized fixture. The name 'optimization step' refers to this algorithm being a small part of a bigger whole. One individual step might not optimize a fixture design, but this one step is needed to get to the final optimized result.

An optimization controller is responsible for applying different rounds of optimization steps. An example is the hill climb algorithm, which repeatedly applies the random step algorithm. After each application of the random step algorithm, it checks if the new iteration is better than the previous one, if so, it is used as a seed for the next iteration, if not, is thrown away. Each application of an optimization step algorithm indicates a transition, the current best known fixture and history that lead to this design indicate form the optimization controller state. The algorithm finishes after a certain strain threshold is reached, or the maximum number of iterations is reached.

This chapter describes X optimization step algorithms and Y optimization controller algorithms. Chapter 7 describes per algorithm the results when applied to some benchmark set.

Something about contact interference? Or put this in Chapter 6?

## 5.1 Random step

The random step algorithm is perhaps the most elementary optimization step algorithm: each iteration, each support probe and each push finger is adjusted in a random direction by a random distance. A possible implementation is the following: take a random value in the in the range $[0, 2\pi)$ as direction, and take a random value in the range $[0, \Delta]$ as distance.

## 5.2 Slope-based step

The slope-based step algorithm is a local optimization algorithm. This method can be seen as a variant of the gradient descent method. With the gradient descent method, one takes a step in solution space that is proportional to the derivative of the current point in solution space. However, since the fixture solution space is unknown except for the current point, the slope-based step cannot use the derivative for iteration.

The slope-based step is applied per support probe and per push finger for some test stage. For each such contact, one takes the known slope of the deformed DUT and takes a step that is proportional to this slope. Each push finger moves 'uphill', each support probe 'downhill'.

Section 6.3 describes the implementation of this algorithm. Section 7.3 describes test results acquired.

## 5.3 Force-based step

The force-based step algorithm adds and/or removes support probes and push fingers based on how much each one exerts during the different test stages. Based on a seed fixture, a new fixture is created. For each contact of old fixture, one the following three actions can be applied:

- Remove: The old contact is not used on the new fixture.
- Copy: The old contact is copied to the new fixture.
- Duplicate: Two instances of the old contact are copied to the new fixture, both placed as close as possible to the original contact position.

Two variants of the force-based step algorithm exist: one in which the optimization controller decides how many support probes or push fingers are removed and/or duplicated per test stage, and one in which the step algorithm decides which action to apply per contact. Using the optimization controller's point of view, these two variants are respectively called the manual version and the automatic version.

The manual version works in the following manner. Per test stage $i$, two numbers $n_i$ and $m_i$ are given, indicating the number of support probes or push fingers need to be respectively removed or duplicated. First, all test probes are copied. Second, the $n_0$ support probes exerting the least amount of force on the DUT during the resting stage are removed, the $m_0$ support probes exerting the largest amount of force are duplicated, and all other support probes are copied once. Third, a removal set is created by selecting per test stage $i$ the $n_i$ push fingers that exert the least amount of force, and a duplication set is created by selecting per test stage the $m_i$ push fingers exerting the largest amount of force. Finally, all push fingers that are not in the removal set are copied, after which all push fingers in the duplication set are copied a second time. Note that using this structure, a push finger that is marked for removal in one test stage, can be marked for duplication at another test stage, resulting in the push finger being copied exactly once.

The automatic version differs slightly from the manual version. A push finger goal force $gf$ is given. First, all test probes are copied. Second, all support probes which during the resting stage exert a force less than half their spring preload are removed, all support probes exerting a calculated force of more than their spring preload are duplicated, and all other support probes are copied once. Third, a copy set is created by selecting per test stage all push fingers exerting a force bigger than $gf/2$, and a duplication set is created by selecting per test stage all push fingers exerting a force bigger than $gf$. Finally, all push fingers that are in the copy set are copied, after which all push fingers in the duplication set are copied a second time.

Section 6.4 describes the implementation of these two algorithms. Section 7.4 describes test results acquired.

## 5.4 Displacement-based step

The displacement-based step algorithm adds support probes and/or push fingers based on peaks in the DUT deformation.

Per test stage, the number of to be added contacts is given. Per test stage, these are then added near the points with biggest absolute displacement. In case this displacement is in positive Z direction, a push finger is added as close as possible to this displacement peak. In this displacement is in negative Z direction, a support probe is added with a push finger directly above it.

Section 6.5 describes the implementation of these two algorithms. Section 7.5 describes test results acquired.

## 5.5 Strain-based step

bladiebla

Table 5.1: Example priority queue over time

| iteration | entry 1 | entry 2 |
|---|---|---|
| $i_0$ | $s(f_0) = 20$ | |
| $i_1$ | $s(f_2) = 11$ | $s(f_3) = 14$ |
| $i_2$ | $s(f_2) = 11$ | $s(f_4) = 13$ |
| $i_3$ | $s(f_2) = 11$ | $s(f_4) = 13$ |

## 5.6 Hill climb controller

The opening text of this chapter described an example hill climb controller. The described hill climb controller algorithm consisted of executing the random step algorithm. If the fixture produced by this step algorithm is better than the fixture used as a fixture, it replaces the seed fixture, otherwise it is ignored. The step algorithm is then repeatedly applied until $n$ succeeding iterations do not show more than $x\%$ improvement, or if a maximum number of iterations $i_{max}$ is reached.

The hill climb controller algorithm presented in this chapter is more advanced, although its mechanics are still quite simple. The controller uses depth-first search as well, but is extended with backtracing functionality.

Key element of the controller algorithm is a priority queue containing up to $n$ fixtures. Each fixture is associated with a score $s$ representing how good the fixture is, using the definition of Section 4.2. This priority is initialized with some to be optimized fixture which is marked as 'not optimized yet'.

Each iteration starts by selecting from the priority queue the fixture with the best score $s$ marked 'not optimized yet'. This fixture will next be used as a seed fixture. The 'not optimized yet' marker is removed from the seed fixture. Next, several step algorithms are executed using the seed fixture. Each step algorithm is used with some fixed parameters. These parameters are not modified by the controller algorithm, but are manually selected before using controller algorithm. Multiple step algorithm instances can be executed, each with different parameters. The different produced fixtures are then analyzed and stored in the priority queue, each with its score and a 'not optimized yet' marker. If the priority queue now contains more than $n$ entries, the worst entries are removed such that the $n$ best entries are kept.

The described process is repeatedly executed, until either $i_{max}$ iterations have been executed, or if the priority queue does not contain any element marked 'not optimized yet'. When one of these two conditions is reached, the fixture with the best score $s$ is returned as the found optimum.

As an example, take a hill climb controller with a priority queue of size $n = 2$. Two optimization algorithms $A$ and $B$ are selected, $A$ is used with some parameter $x$, $B$ is once used with parameter $y$ and once with parameter $z$. Start with an initial fixture $f_0$ with a score $s(f_0) = 20$. In the first iteration, $f_0$ is used as a seed. Applying the three step algorithms gives $f_1 = A_x(f_0)$, $f_2 = B_y(f_0)$ and $f_3 = B_z(f_0)$, with $s(f_1) = 19$, $s(f_2) = 11$ and $s(f_3) = 14$. Hence, $f_2$ is used as the next seed. Applying the same procedure results in $s(f_4) = 13$, $s(f_5) = 17$ and $s(f_6) = 15$. $f_2$ is still the best found fixture so far, but has already been used as a seed. Hence, $f_4$ is selected as a seed, producing $s(f_7) = 14$, $s(f_8) = 16$ and $s(f_9) = 18$. After this iteration, no items in the priority queue are marked 'not optimized yet'. Table 5.1 gives a summary of the contents of this priority queue over time. Figure 5.1 visualizes the explored search tree.

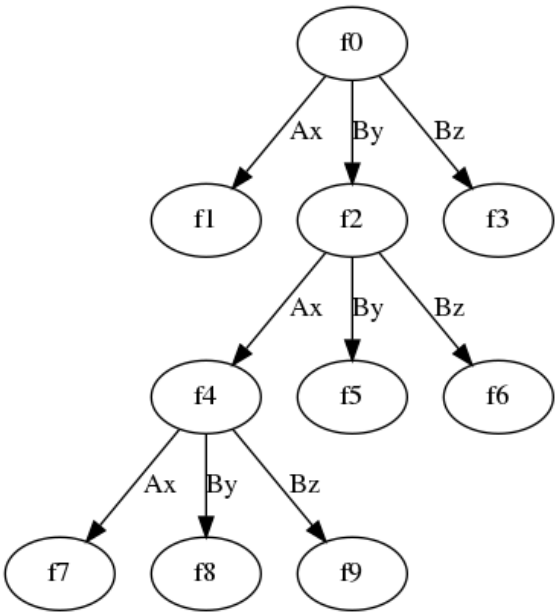



Figure 5.1: Example explored search tree

## 5.7 Simulated annealing

bladiebla

# Chapter 6

# Implementation

bladiebla

## 6.1 Input/output file formats

bladiebla

## 6.2 Domain model

bladiebla

## 6.3 Slope-based step

This section describes in a high-level manner how the slope-based step is implemented. One by one, all push fingers and test probes are adjusted according to the description given in Section 5.2.

First, the mesh representing the deformed DUT is taken. Per contact to be adjusted, a set of mesh vertices/points near the centre of the contact is taken. Using a least-squares method, a plane is fit through this set of points. This plane represents the 2D slope at the contact. From this plane, the normal is taken. This normal $n$ is then normalized, i.e. $n = n/|n|$.

Next, the normalized normal is projected on the original surface by reduction from $n = (x, y, z)$ to $n_{proj} = (x, y)$. If the to be adjusted contact is a push finger, $n_{proj}$ is added to the finger position to make the push finger 'walk upwards'. If the to be adjusted contact is a support probe, $n_{proj}$ is subtracted from the original position to make the contact 'walk downwards'.

In order to control the step size, $n_{proj}$ can be multiplied with an aggressiveness value. Section 7.3 presents the results of applying this described algorithm and the effects of different aggressiveness values.

Add pictures?

## 6.4 Force-based step

This section describes in a high-level manner how the two force-based step algorithms are implemented. Based on a seed fixture, a new fixture is designed. Each old contact is copied zero, one or two times to the new fixture, all according to the description given in Section 5.3.

The manual version is implemented in the following manner. First, all test probes of the old fixture are copied to the new fixture. Second, all old support probes are sorted on the amount of force each one produces during the resting stage. Using this ordering, the $n_0$ support probes exerting the least amount of force are ignored, but all other support probes are copied to the new

fixture. Again using the ordered list of old support probes, the $m_0$ support probes exerting the largest amount of force are copied a second time and inserted as close as possible to their original locations, while keeping a safe distance to PCB edges, DUT components and other probes. Third, all push fingers are per test stage sorted on force exerted, then copied zero, one or two times to the new fixture, using the same steps as used on the support probes.

The automatic version is implemented in the same way, but using the different copy criteria described in Section 5.3.

Section 7.4 describes test results acquired after application of the described algorithm implementation.

## 6.5 Displacement-based step

This section describes how the displacement based step algorithm is implemented. Based on a seed fixture, a new fixture is designed. All old contacts are copied once, and a given number of support probes or push fingers is added, all according to the description given in Section 5.4.

First, the mesh representing the deformed DUT is taken. Per test stage, all vertices are ordered on absolute Z displacement. Next, the vertex with the biggest displacement is picked. In case this displacement is in positive Z direction, a push finger is added as close as possible to the vertex position. In case the displacement is in negative Z direction, a support probe is in a similar way, after which a push finger is added above the support probe. This is repeated for the desired number of added contacts.

Section 7.5 describes test results acquired after application of the described algorithm implementation.

## 6.6 Strain-based step

bladiebla

## 6.7 Hill climb controller

bladiebla

## 6.8 Simulated annealing

bladiebla

# Chapter 7

# Experimental results

This chapter describes the results obtained by applying the algorithms described in Chapter 5 using the implementations described in Chapter 6.

First, a set of PCBA's is introduced that will be used as the benchmark set. Next, test methodologies are described. The rest of the chapters then describes per algorithm the benchmark results and draws some intermediate conclusions. Based on the results described in this chapter, Chapter 8 deduces a final conclusion.

## 7.1 Benchmark set

The benchmark set consists of four different DUT's. These four DUT's span a wide PCBA spectrum: both test and production boards are selected, both panelized and non-panelized boards, both thick and thin PCB's are represented, and both empty and high-density boards are among the benchmark set.

The first benchmark DUT is the *FTS* board. This board is designed for testing purposes and has a size of only $100 \times 50 \times 0.5mm$. No components are present and, except for the three alignment holes, no cut-outs.

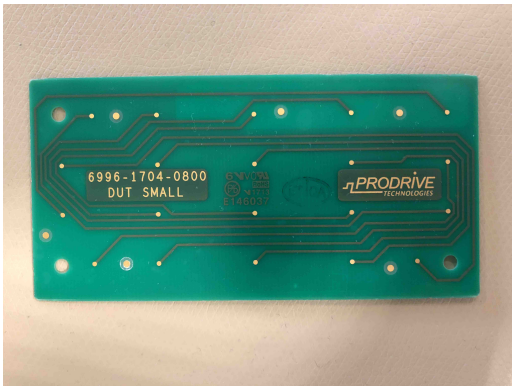
The second benchmark DUT is the *ODDD* board. This board is quite large, with a size of $400 \times 350 \times 2.4mm$. Several cutouts are present. The board has a medium component density on the top side and a very low component density on the bottom side, leaving lots of space for push fingers and support probes. The test probe density is quite low.

The third benchmark DUT is the *TNXTD* board. This board has a size of $388 \times 251 \times 1.6mm$. The board consists of four identical panels. Per panel, a few cutouts are present. At some places, the board has a very high test probe density. Space for support probes and push fingers is locally limited.

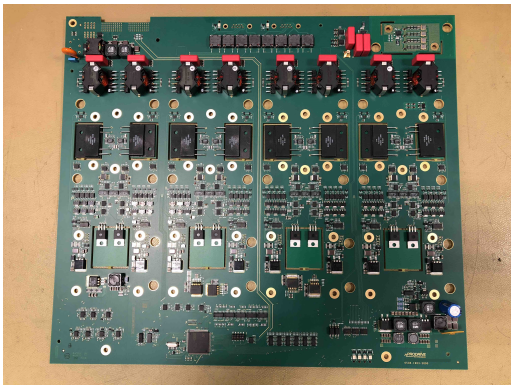
The fourth benchmark DUT is the *P1SMD* board. This board has a size of $249 \times 161 \times 0.8mm$. The board consist of eleven identical panels. Per panel, almost no space is available for the placement of push fingers, creating an optimization challenge.

Figure 7.1 provides photos of the top and bottom sides of the four benchmark DUT's. Table 7.1 gives an overview of the number of test probes, support probes and push fingers used in the manually designed fixtures.

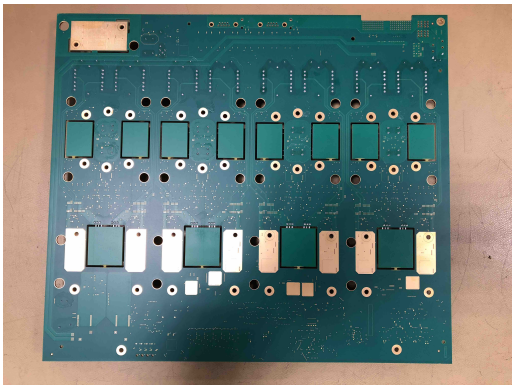

(a) FTS, top



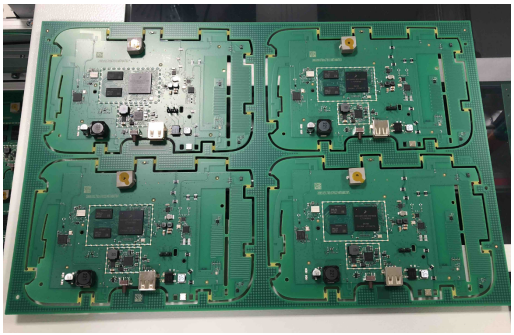

(b) FTS, bottom



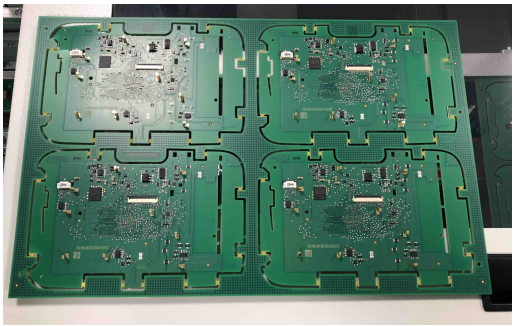
(c) ODDD, top



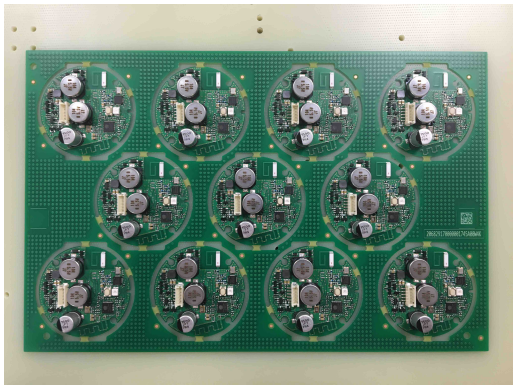
(d) ODDD, bottom



(e) TNXTD, top



(f) TNXTD, bottom

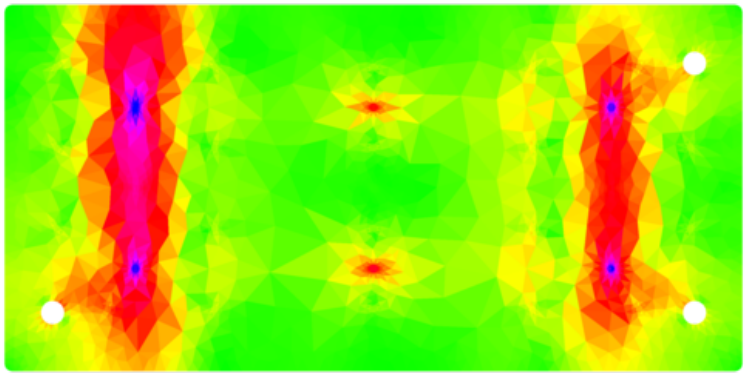


(g) P1SMD, top

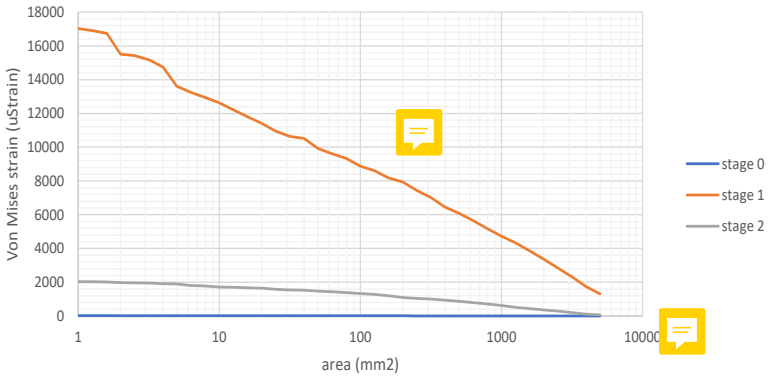
Figure 7.1: Benchmark set: DUT photos

Table 7.1: Benchmark set: fixture statistics

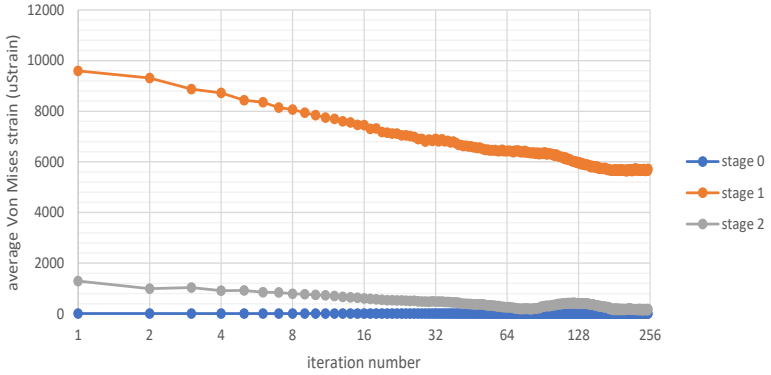
| DUT | test probes | support probes | push fingers |
|---|---|---|---|
| FTS | 20 | 9 | 9 |
| ODDD | 158 | 51 | 155 |
| TNXTD | 432 | 31 | 158 |
| P1SMD | 264 | 127 | 83 |



(a) Example strain map




(b) Example strain versus area




(c) Example strain versus iteration

Figure 7.2: Example results

## 7.2 Result representation

In accordance with the project objective stated in Section 4.2, analysis of the test results will focus on DUT strain and fixture costs. Additionally, convergence of results will shortly discussed.

A first method of strain presentation is a strain map of the PCB. Define the maximum surface strain to be $x$ $\mu strain$. Then, area's marked green represent 0 $\mu strain$, area's marked yellow represent $1/4x$ $\mu strain$, area's marked red represent $1/2x$ $\mu strain$, area's marked purple represent $3/4x$ $\mu strain$, and area's marked blue represent $x$ $\mu strain$. An example of such a strain map is shown in Figure 7.2a. Key advantage of this strain presentation is that it clearly shows the weak points of a fixture. A drawback is that only one stage of one fixture can be shown at a time.

A second method of strain presentation is a strain versus area plot. An example of such plot is shown in Figure 7.2b. The vertical axis shows strain, while the horizontal axis shows what board area fraction has that level of strain. Key advantage of this presentation is that it clearly shows what percentage of the board area exceeds a certain strain threshold. A drawback is that this plot does not show where these weak points are located. One has to be careful to correctly interpret this plot. Since FEA is an approximation of reality, the calculated strain values can deviate from reality. On larger surfaces, these effects can be negligible, but at small surfaces, these deviations can be bigger. In the example of Figure 7.2b, the spike at the top 0.02% strain area is caused by FEA inaccuracy.

A third method of strain presentation is a strain versus iteration plot. This presentation method can be seen as a summarized version of the previous strain presentation method. Per test stage, a weighted average of the board strain is taken. This way, different fixtures can quickly be compared. An example of such comparison is shown in Figure 7.2c.

A first method of fixture cost representation consists of adding up the prices of the individual probes and push fingers. In this thesis, this can be done by attaching fictional but reasonable prices to the different component types. As an example, one can set the price of one support probe to be equal to four push fingers. Prices of test probes are not relevant here, since these are given and cannot be optimized.

A second method of fixture cost representation consists of simply presenting the total number of support probes and push fingers, in a manner comparable to Table 7.1. Drawback of this method is that comparing two different fixtures is not always possible, e.g. fixture A has more support probes than fixture B, but fixture B has more push fingers than fixture A.

Computational costs can be expressed in total number iterations needed to need the final result. It is true that computation times can fluctuate between iterations, but such small differences can safely be neglected.

Convergence speed is a more detailed variant of the computational cost presentation. By plotting strain or fixture costs per iteration, one can see how consistent optimization algorithms are: e.g. the differences between every two iterations are consistent, or the main part of optimization is done in the first few iterations. An example of such plot is shown in Figure 7.2c.

## 7.3 Slope-based step

This section describes the results of applying the benchmark set described in Section 7.1 to the algorithm described in Section 5.2 and implemented according to Section 6.3. As described in Section 6.3, this algorithm has precisely one variable, *aggressiveness*. Hence, not only need is the algorithm tested against the different benchmark DUT's, but also using different aggressiveness values per DUT.

Using an aggressiveness of 50.0, the algorithm gives divergent results. After only a few iterations, the generated fixtures are worse than the fixture used as a seed. However, after ten iterations, the system starts to converge. From iteration 40, the slope step algorithm does no longer improve. This is shown in Figures 7.3 and 7.4.

Using an aggressiveness of 40.0, the algorithm consistently gives convergent results. Figures 7.5 and 7.6 show that the algorithm converges after approximately 160 iterations.

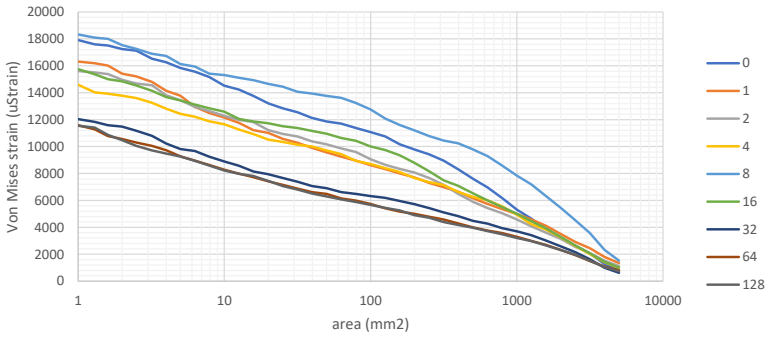



Figure 7.3: FTS; slope step; aggressiveness 50.0; first stage; strain versus area

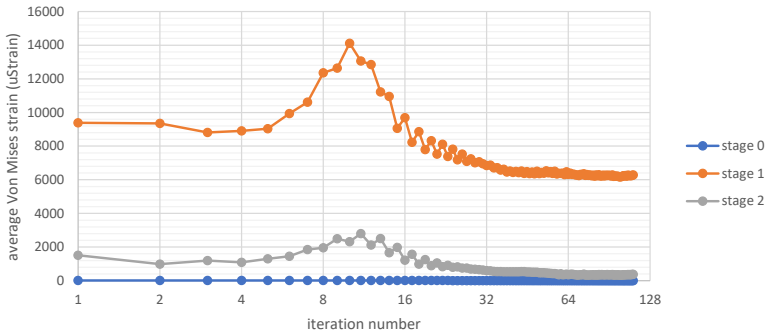



Figure 7.4: FTS; slope step; aggressiveness 50.0; strain versus iteration

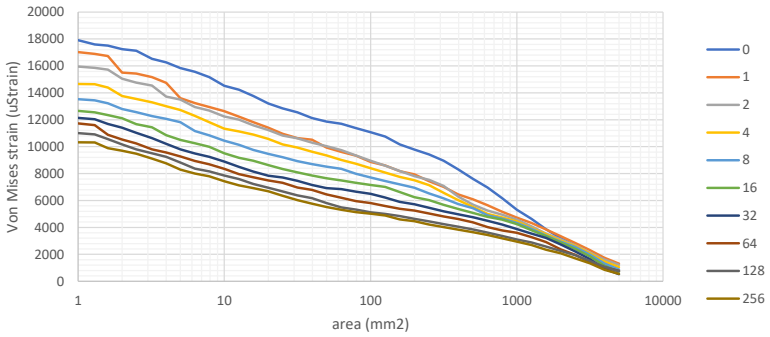



Figure 7.5: FTS; slope step; aggressiveness 40.0; first stage; strain versus area

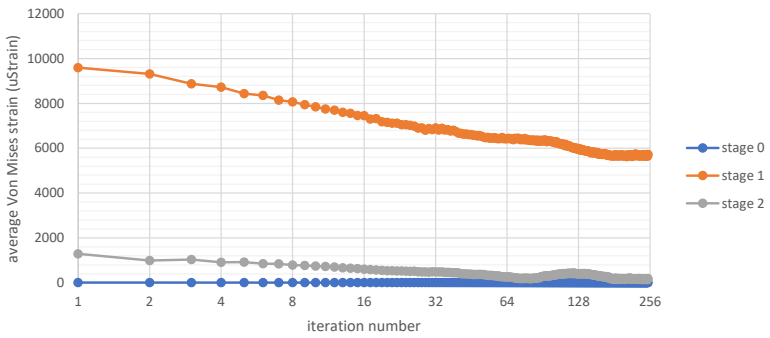
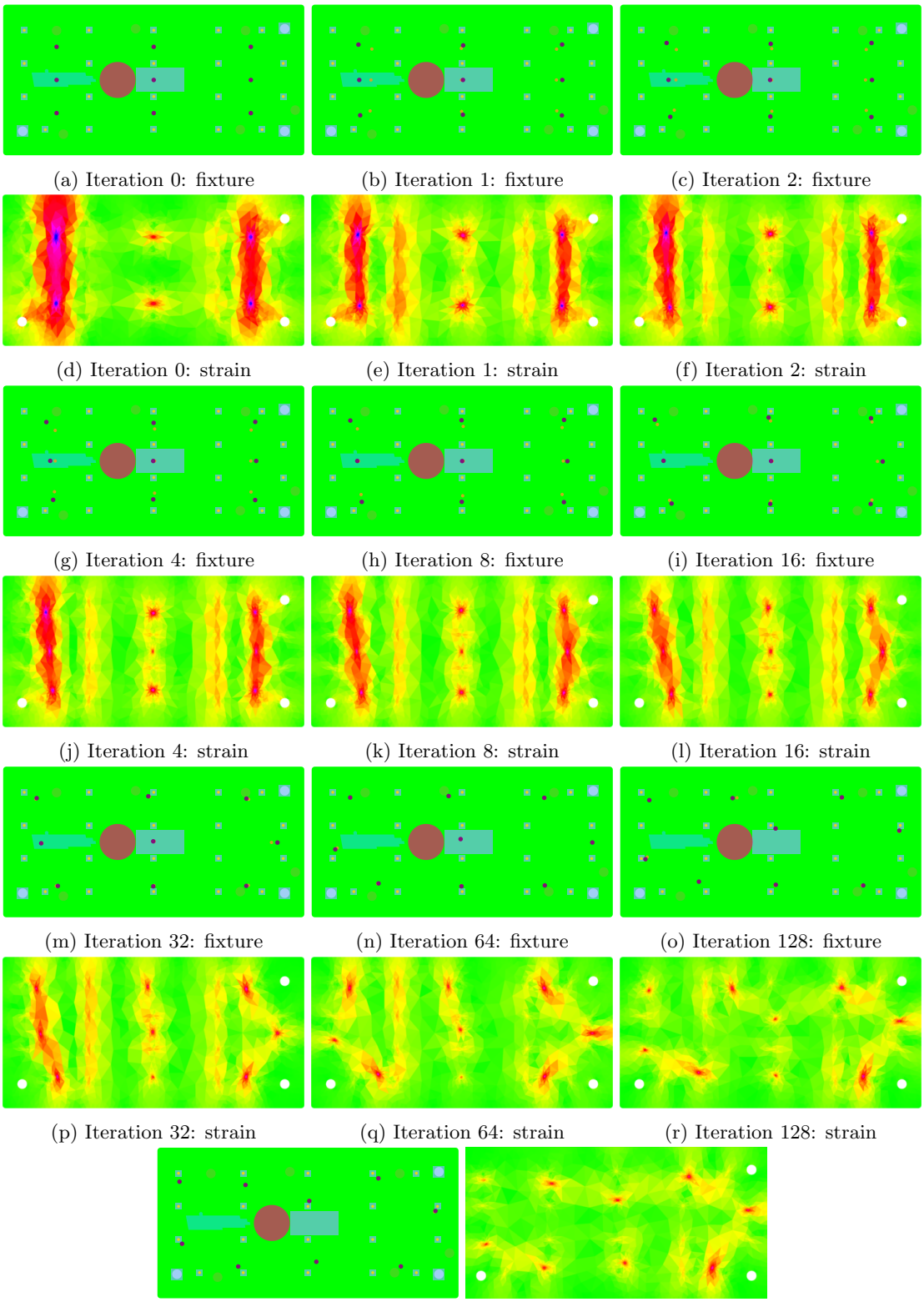



Figure 7.6: FTS; slope step; aggressiveness 40.0; strain versus iteration



(a) Iteration 0: fixture (b) Iteration 1: fixture (c) Iteration 2: fixture

(d) Iteration 0: strain (e) Iteration 1: strain (f) Iteration 2: strain

(g) Iteration 4: fixture (h) Iteration 8: fixture (i) Iteration 16: fixture

(j) Iteration 4: strain (k) Iteration 8: strain (l) Iteration 16: strain

(m) Iteration 32: fixture (n) Iteration 64: fixture (o) Iteration 128: fixture

(p) Iteration 32: strain (q) Iteration 64: strain (r) Iteration 128: strain

(s) Iteration 256: fixture (t) Iteration 256: strain

Figure 7.7: FTS; slope step; aggressiveness 40.0; first stage; strain map

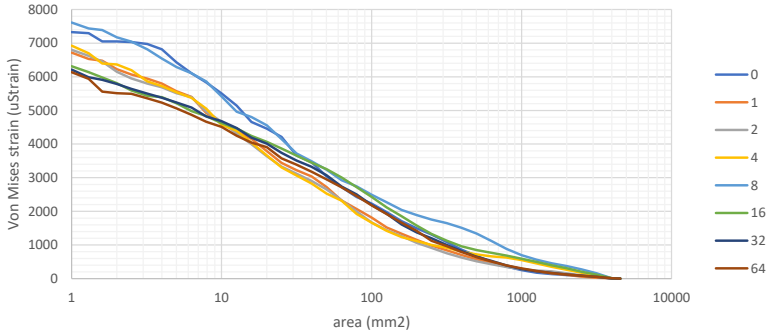



Figure 7.8: P1SMD; slope step; aggressiveness 500.0; first stage; strain versus area

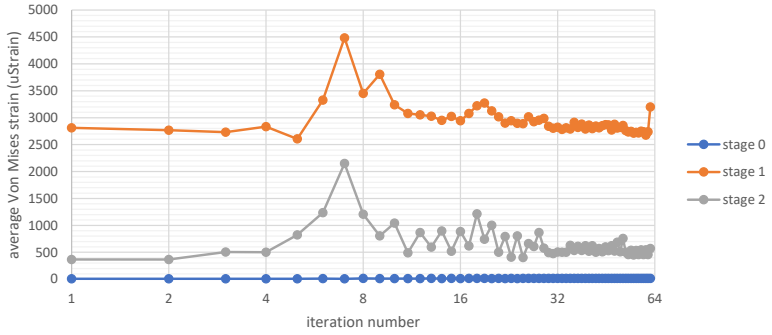


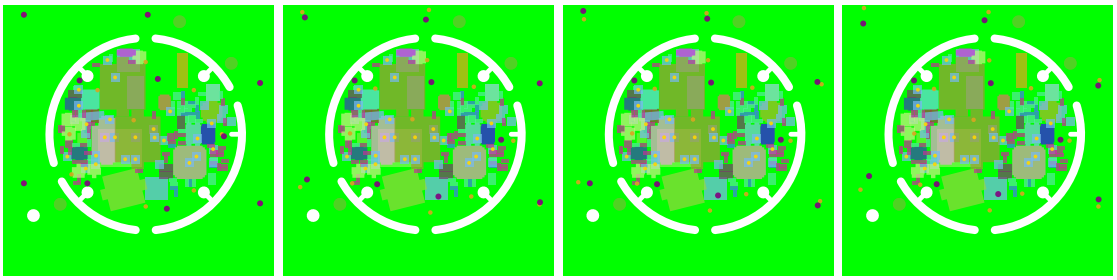

Figure 7.9: P1SMD; slope step; aggressiveness 500.0; strain versus iteration

Figure 7.7 gives a deeper looking at the converging process. By design, the DUT is almost symmetric. This symmetry almost causes the algorithm to not further improve from iteration 25 onward, but the small design irregularity causes the top centre push finger to move a bit to the left, allowing a restructuring of the original $3 \times 3$ push finger configuration to the craggy $4 \times 2 + 1$ configuration. Another thing to note is that the push fingers and support probes keep matching each other.

The *ODDD* and the *TNXTD* boards have not been tested against the slope-based step algorithm.

Instead of applying the slope-based step algorithm on the original P1SMD board, only one part of the board has been used.

Using an aggressiveness of 500.0, the algorithm gives divergent results. New generated fixtures are sometimes worse, sometimes better than the fixture used as a seed. The system does not



(a) Iteration 0 (b) Iteration 1 (c) Iteration 2 (d) Iteration 3

Figure 7.10: P1SMD; slope step; aggressiveness 500.0; generated fixtures

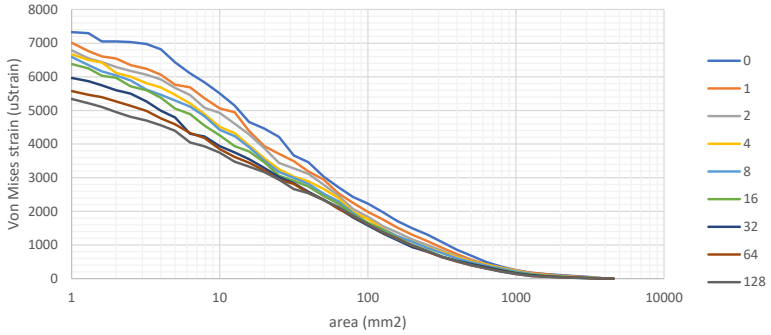



Figure 7.11: P1SMD; slope step; aggressiveness 250.0; first stage; strain versus area

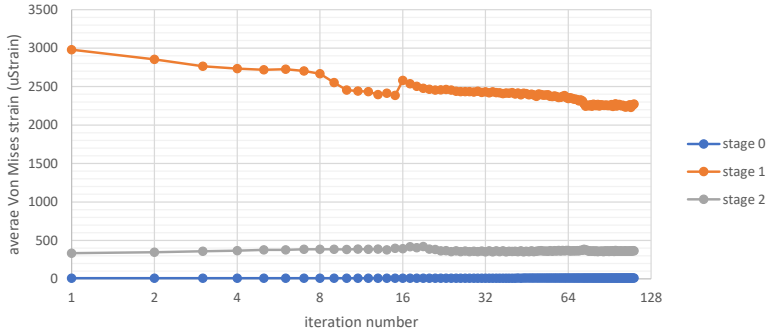



Figure 7.12: P1SMD; slope step; aggressiveness 250.0; strain versus iteration

converge. This is shown in Figures 7.8 and 7.9.

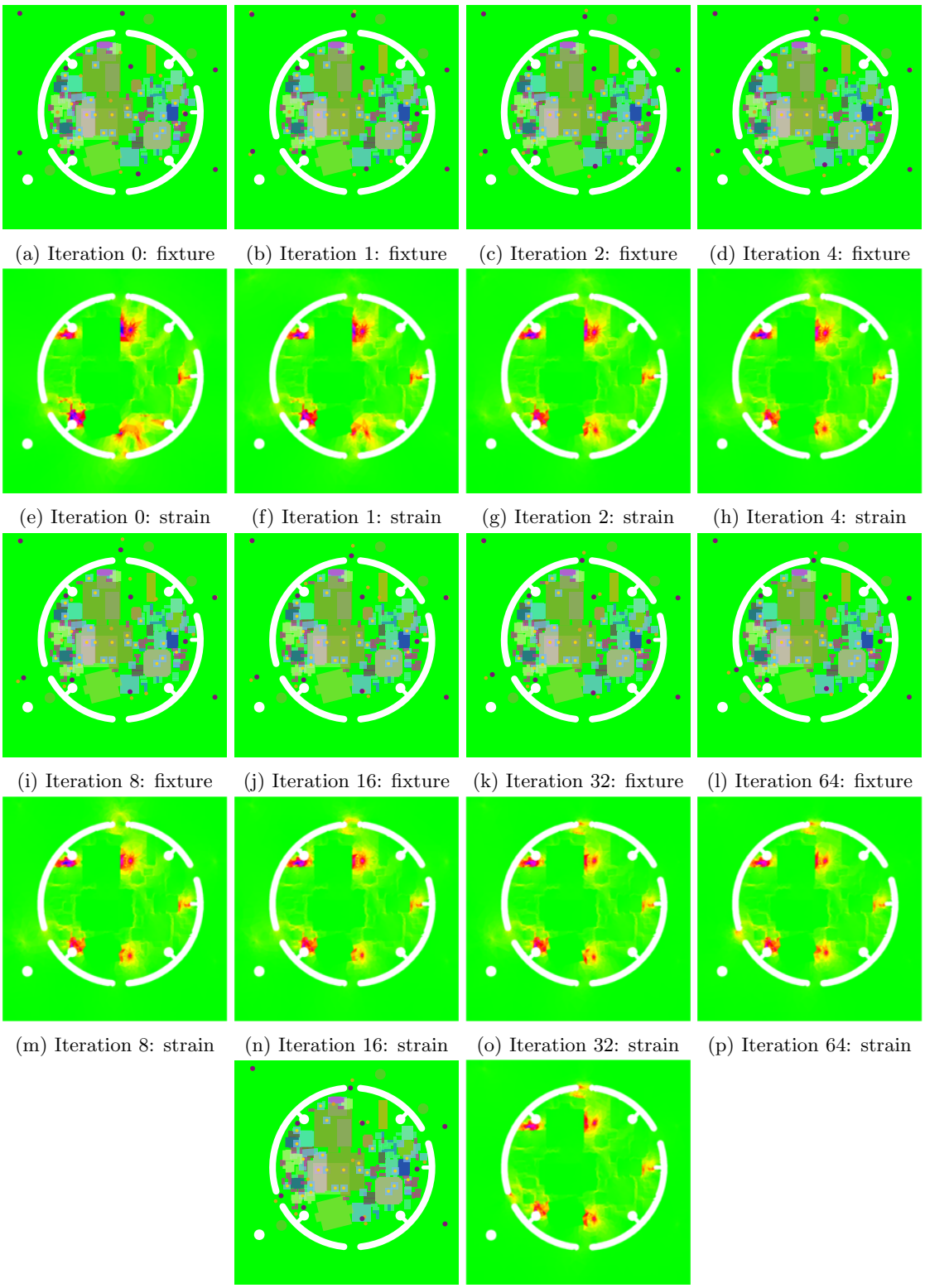
The lack of convergence can be explained by taking a look at the generated fixture designs, shown in Figure 7.10. The different push fingers and support probes on the panel move slowly towards a local optimum position, while the different push fingers and support probes on the empty PCB surrounding the panel overshoot each iteration.

Using an aggressiveness of 250.0, the algorithm does give convergent results. Figures 7.11 and 7.12 show that the algorithm converges until iteration 15, switches to worse design, continues to converge, then stabilizes around iteration 75.

Figure 7.13 gives a deeper looking at the converging process. The original fixture has five push fingers on the panel and five push fingers on the surrounding PCB. Nine support probes support the panel, while five support probes support the surrounding PCB. Twenty-four test probes exert forces on the panel, causing strain to concentrate on the panel push fingers. The slope-based step algorithm causes two of the outer push fingers to move onto the panel, while the different support probes move towards the closest by push finger. After convergence, some push fingers have multiple support probes close by.

Based on these benchmarks, the slope-step algorithm seems to be a very efficient algorithm. The algorithm is able to converge to a local optimum in only a few iterations. Such local optimum is then often improved to an even better local optimum in around one hundred iterations.

The algorithm has also some drawbacks. First, the aggressiveness needs to be controlled. If this value is too low, convergence will be slow. If this value is too high, the algorithm overshoots, giving diverging results. Hence, this algorithm needs an optimization controller that is able to choose a fitting aggressiveness value. Second, the algorithm can only move contacts, not add or remove them. In the FTS example, adding one extra push finger at the right side might have improved the optimization process. In the P1SMD example, multiple support probes converging towards the same push finger could have been replaced by a single support probe.



(a) Iteration 0: fixture (b) Iteration 1: fixture (c) Iteration 2: fixture (d) Iteration 4: fixture

(e) Iteration 0: strain (f) Iteration 1: strain (g) Iteration 2: strain (h) Iteration 4: strain

(i) Iteration 8: fixture (j) Iteration 16: fixture (k) Iteration 32: fixture (l) Iteration 64: fixture

(m) Iteration 8: strain (n) Iteration 16: strain (o) Iteration 32: strain (p) Iteration 64: strain

(q) Iteration 110: fixture (r) Iteration 110: strain

Figure 7.13: P1SMD; slope step; aggressiveness 250.0; first stage; strain map

## 7.4 Force-based step

This section describes the results of applying the benchmark set described in Section 7.1 to the automatic force step algorithm described in Section 5.3 and implemented according to Section 6.4. The manual version of the algorithm is not benchmarked. Since that version needs an controller algorithm which adjusts the parameters of the step algorithm per iteration, such benchmarks would mainly test the quality of the controller algorithm, instead of the quality of the step algorithm.

The automatic force step algorithm has precisely one adjustable variable, *push finger force goal*. Hence, this section will not only describe the algorithm tested against the different benchmark DUT's, but also using different *force goal* values.

Using the FTS board, the algorithm is repeatedly applied, each time using a different force goal. Figure 7.14 shows the average strain per iteration. Furthermore, the fixture cost relative to the original fixture is shown. Using a force goal of 4500mN, the results converge after only one iteration. It does however not stabilize, each iteration some push fingers are removed and some are duplicated. The same behaviour can be observed when using a force goal of 3000mN and 1500mN: the results converge after a couple of iterations, after which 'shaky' behaviour is shown.

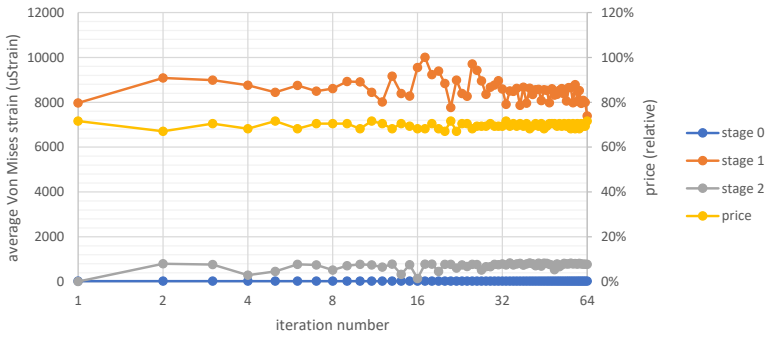
Figure 7.15 gives a more in-depth look at the algorithm behaviour when using a force goal of 1500mN. At the first stage, each test probe exerts 1500mN force on the DUT, resulting in the generated fixtures converging towards approximately 30 push fingers, slightly more than test probes and support probes present. However, the algorithm is not capable of placing these different push fingers directly above the different probes. If the different placed push fingers would be shifted to these probe positions, all board strain would approximately zero, while the number of push fingers would be reduced as well.

Similar behaviour can be observed when benchmarking the P1SMD board. Again, force goals of 4500mN, 3000mN and 1500mN are used. Convergence is shown in Figure 7.16. Using a force goal of 4500mN, results converge after a single iteration. Using a force goal of 3000mN, results converge after 12 iterations, but showing a second improvement after the 50th iteration. Using a force goal of 1500mN, results converge after 16 iterations, after which results still differ a bit, but do not show major strain differences.

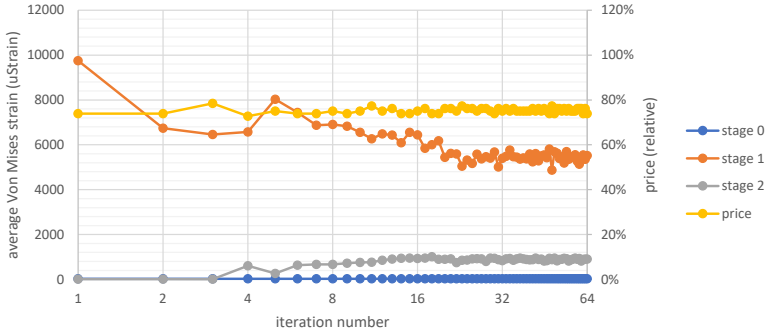
Figure 7.17 shows a more in-depth look on the converging process. During the first iteration, most of the support probes are removed, leaving a minimum number of four support probes. Note that because no force recalculations are done between individual removals, the result is a fixture that would be unstable during the resting stage. In practice, when applying the step algorithm to a complete PCB instead of a single panel, the generated fixture would be stable. However, a controller algorithm should not accept this generated fixture, but for example apply instead several iterations of the manual force step algorithm.

During the second iteration, the push fingers that no longer have any support probe under them are removed as well. Furthermore, from this iteration onwards, the algorithm tries to add extra push finger near the push fingers on the panel. However, this is not always possible because of DUT component interference. As a result, no extra push fingers are added, or a push finger is added on the surrouding empty PCB, only to be removed again in the next iteration. This behaviour is shown in iterations 16, 32 and 64.

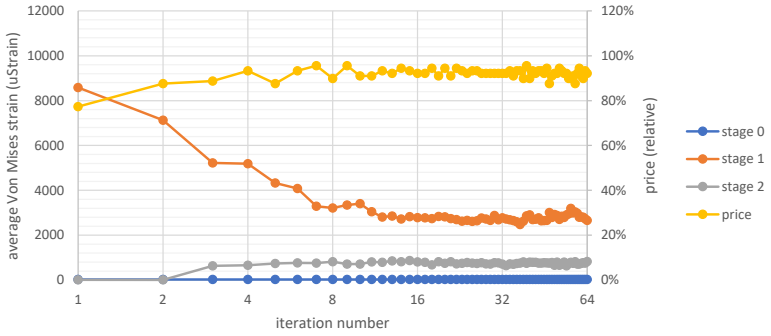
In conclusion: just like with the FTS board, the final result is quite good, but would benefit from small adjustments to the positions of the different contacts.




(a) Force goal 4500mN

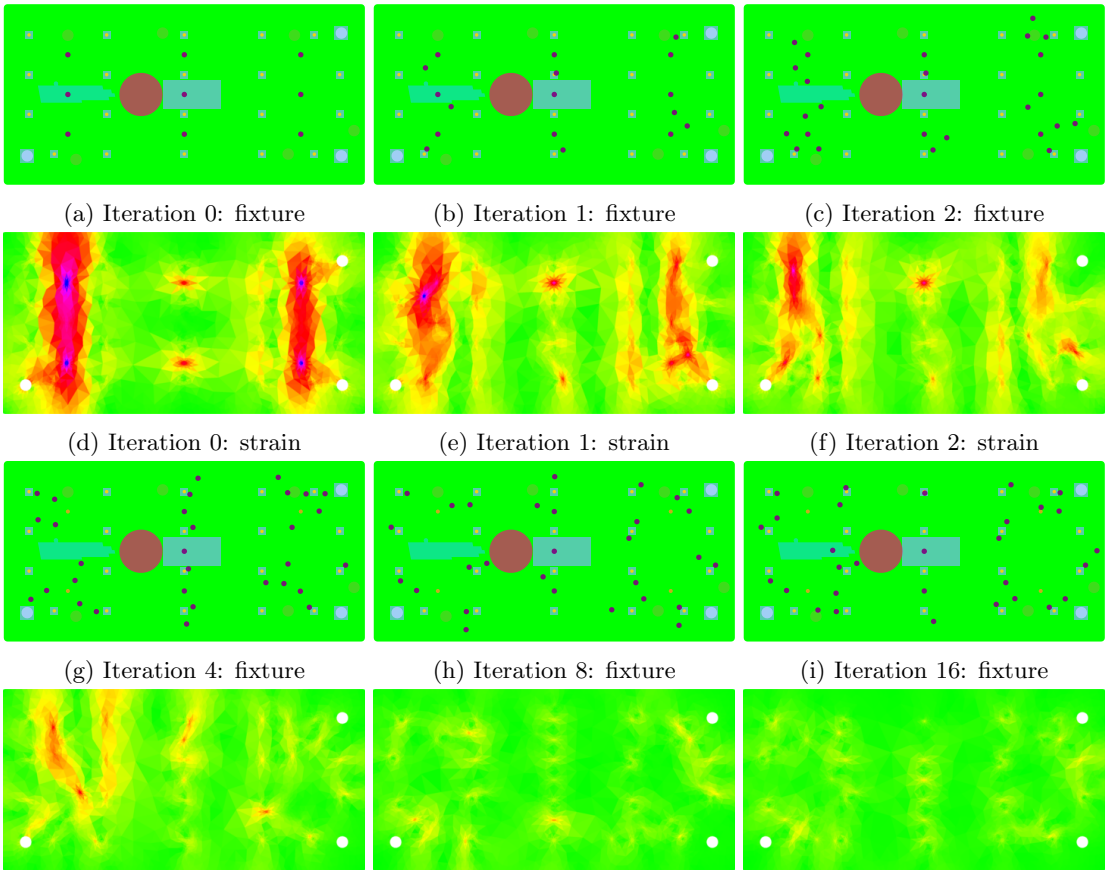



(b) Force goal 3000mN




(c) Force goal 1500mN

Figure 7.14: FTS; force step; strain/price versus iteration



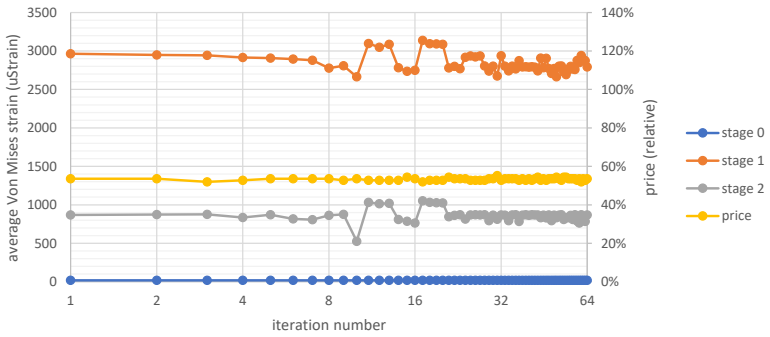
(a) Iteration 0: fixture (b) Iteration 1: fixture (c) Iteration 2: fixture

(d) Iteration 0: strain (e) Iteration 1: strain (f) Iteration 2: strain

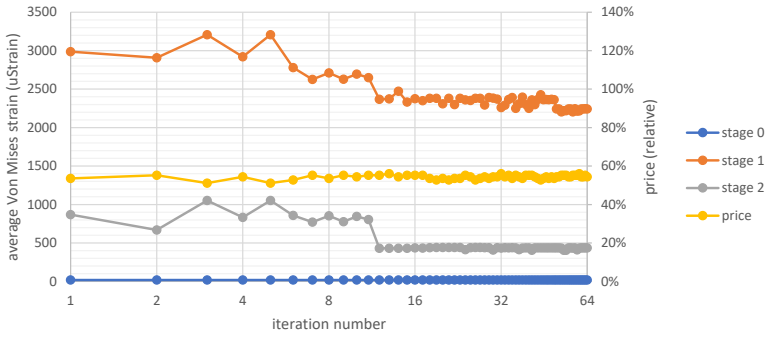
(g) Iteration 4: fixture (h) Iteration 8: fixture (i) Iteration 16: fixture

(j) Iteration 4: strain (k) Iteration 8: strain (l) Iteration 16: strain

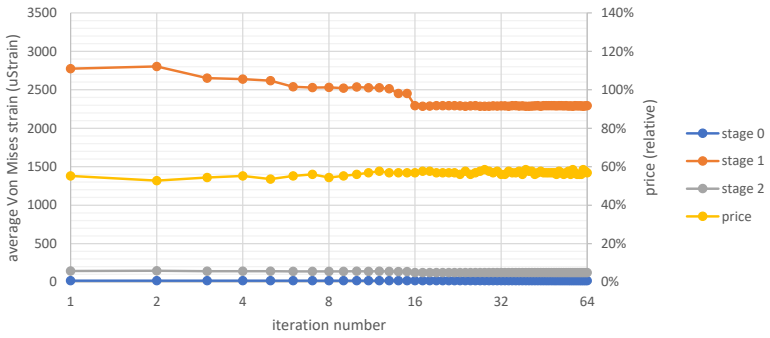
Figure 7.15: FTS; force step; force goal 1500mN; first stage; strain map




(a) Force goal 4500mN

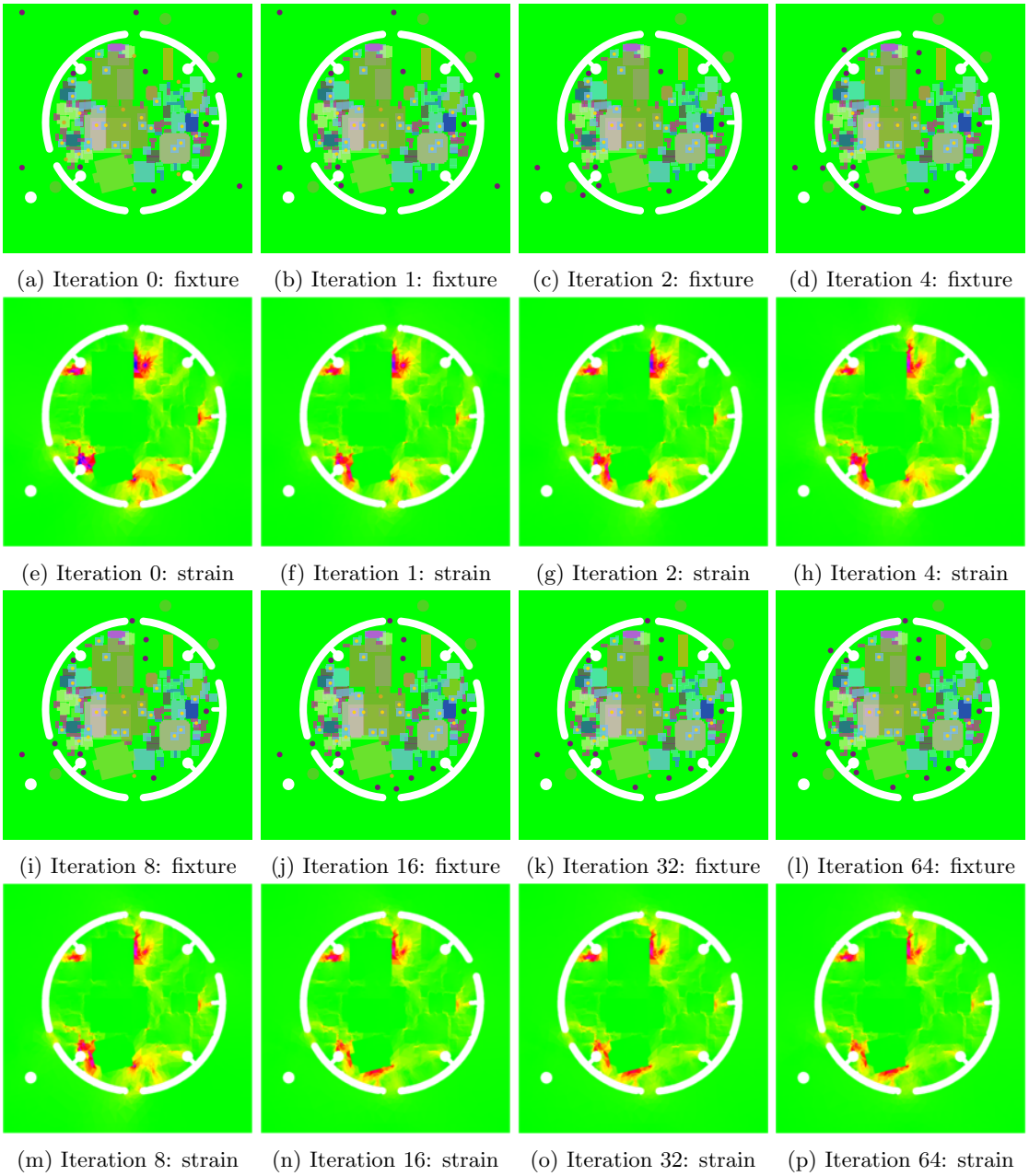



(b) Force goal 3000mN




(c) Force goal 1500mN

Figure 7.16: P1SMD; force step; strain/price versus iteration



(a) Iteration 0: fixture (b) Iteration 1: fixture (c) Iteration 2: fixture (d) Iteration 4: fixture

(e) Iteration 0: strain (f) Iteration 1: strain (g) Iteration 2: strain (h) Iteration 4: strain

(i) Iteration 8: fixture (j) Iteration 16: fixture (k) Iteration 32: fixture (l) Iteration 64: fixture

(m) Iteration 8: strain (n) Iteration 16: strain (o) Iteration 32: strain (p) Iteration 64: strain

Figure 7.17: P1SMD; force step; force goal 1500mN; first stage; strain map

## 7.5 Displacement-based step

This section describes the results of applying the benchmark set described in Section 7.1 to the displacement step algorithm described in Section 5.4 and implemented according to Section 6.5. As described in Section 6.5, this algorithm has per test stage one adjustable parameter, the number of contacts to add.

As a first benchmark, this step algorithm is repeatedly applied to the FTS board. Since this board has no much weight, there is not much use in focusing on the resting stage. Instead, each iteration one contact will be added based on the deformation of the first stage. Figure 7.18 shows per iteration the average strain per test stage and costs of the generated fixture relative to the original fixture.

One can see that the generated fixture converge after approximately 40 iterations. Figure 7.20 shows why no improvements are made after this iteration: there is no free space above any probe to place a push finger on. Instead, push fingers are added at the closest by free space, resulting in a push finger explosion. Just like the force step algorithm, contacts which are placed just off the perfect position block further improvement near that position.

Applying the displacement step algorithm to the P1SMD board gives comparable algorithm characteristics. Or more precisely, even worse results. Figure 7.19 shows the convergence results, Figure 7.21 the corresponding strain maps. The high DUT component density on the panel causes push fingers to be placed on the surrounding PCB, producing another push finger explosion.

In conclusion, the displacement based step algorithm is not very useful if used in a stand-alone manner.

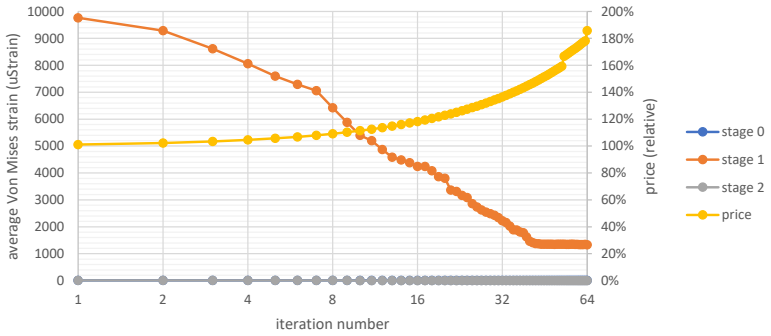



Figure 7.18: FTS; displacement step; strain/price versus iteration

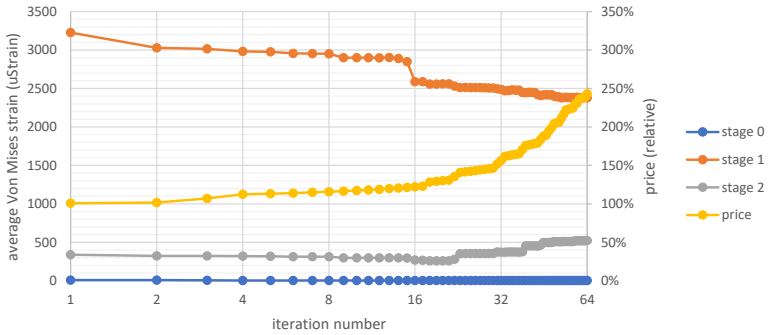
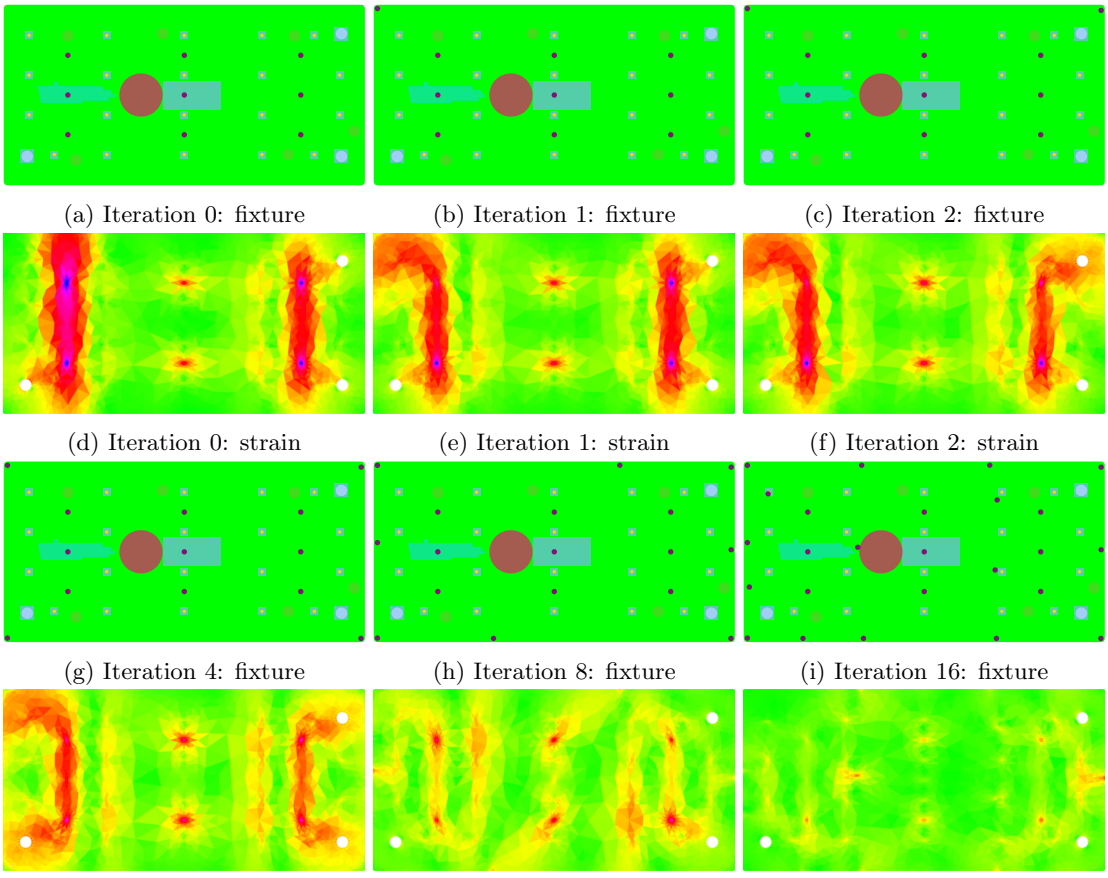



Figure 7.19: P1SMD; displacement step; strain/price versus iteration



(a) Iteration 0: fixture (b) Iteration 1: fixture (c) Iteration 2: fixture

(d) Iteration 0: strain (e) Iteration 1: strain (f) Iteration 2: strain

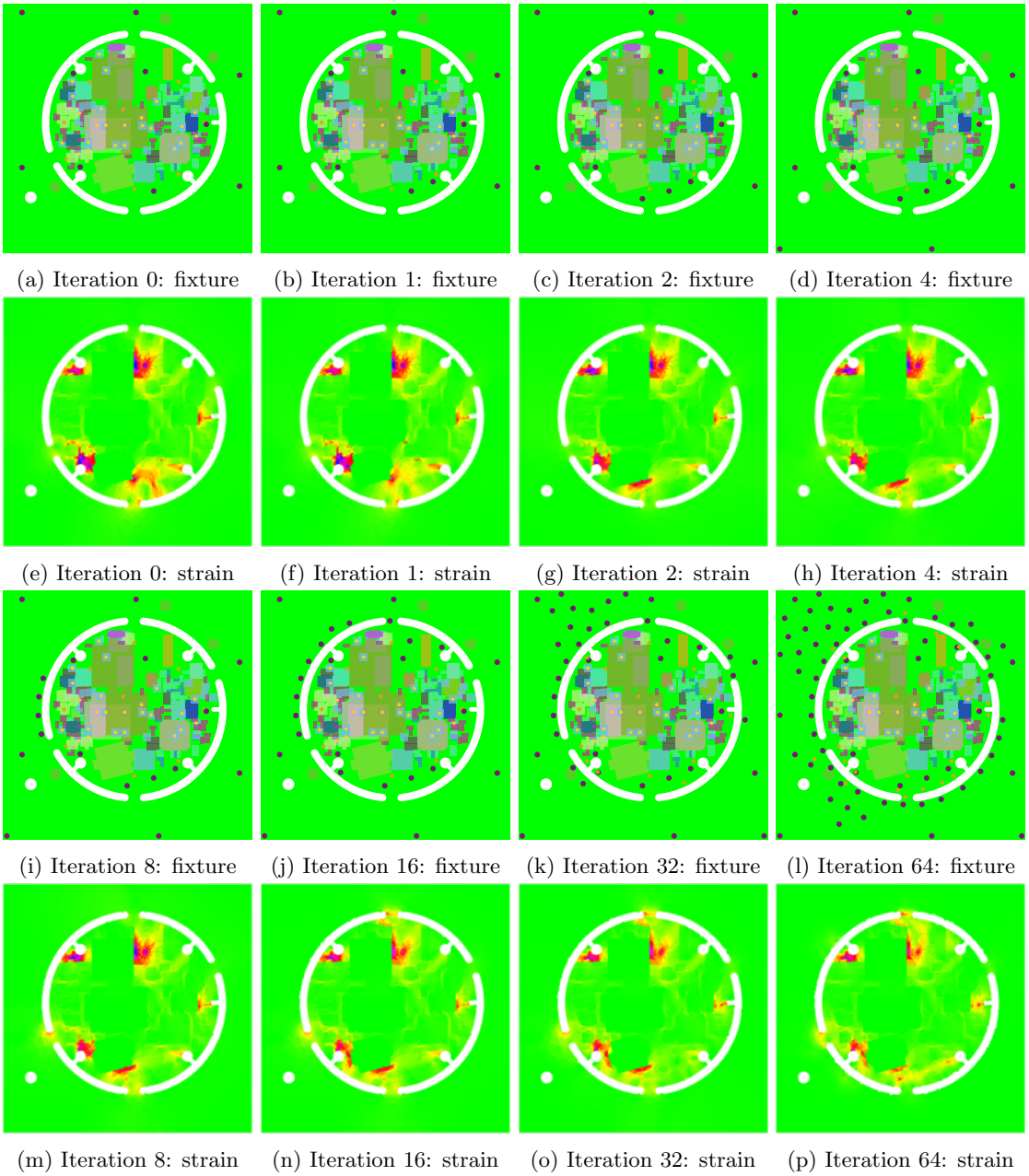
(g) Iteration 4: fixture (h) Iteration 8: fixture (i) Iteration 16: fixture

(j) Iteration 4: strain (k) Iteration 8: strain (l) Iteration 16: strain

(m) Iteration 32: fixture (n) Iteration 64: fixture

(o) Iteration 32: strain (p) Iteration 64: strain

Figure 7.20: FTS; displacement step; first stage; strain map



(a) Iteration 0: fixture (b) Iteration 1: fixture (c) Iteration 2: fixture (d) Iteration 4: fixture

(e) Iteration 0: strain (f) Iteration 1: strain (g) Iteration 2: strain (h) Iteration 4: strain

(i) Iteration 8: fixture (j) Iteration 16: fixture (k) Iteration 32: fixture (l) Iteration 64: fixture

(m) Iteration 8: strain (n) Iteration 16: strain (o) Iteration 32: strain (p) Iteration 64: strain

Figure 7.21: P1SMD; displacement step; first stage; strain map

## 7.6 Strain-based step

bladiebla

## 7.7 Hill climb controller

bladiebla

## 7.8 Simulated annealing

bladiebla

# Chapter 8

# Conclusion

bladiebla

## 8.1 Discussion

bladiebla

## 8.2 Future work

bladiebla

## 8.3 Conclusion

bladiebla

# Appendix A

# Abbreviations

| | |
|---|---|
| AET | Automated Electrical Test |
| BCS | Bottom Clamping Side |
| BGA | Ball Grid Array |
| CAD | Computer Aided Design |
| CFD | Computational Fluid Dynamics |
| CNN | Convolutional Neural Network |
| DOF | Degree Of Freedom |
| DUT | Device Under Test |
| ESD | Electrostatic Discharge |
| ETS | Electrical Test System |
| FEA | Finite Element Analysis |
| FEM | Finite Element Modelling |
| LBM | Lattice Boltzmann Method |
| MTTF | Mean Time To Failure |
| NMAE | Normalized Mean Average Error |
| PCB | Printed Circuit Board |
| PCBA | Printed Circuit Board Assembly |
| PN | Prodrive Number |
| PCA | Principle Component Analysis |
| PSO | Particle Swarm optimization |
| PWA | Printed Wiring Assembly |
| SDF | Signed Distance Function |
| SIMP | Solid Isotropic Microstructures with Penalty |
| SMD | Surface Mount Device |
| SMT | Surface Mount Technology |
| TCS | Top Clamping Side |