Eindhoven University of Technology

MASTER

Automatic generation of in-circuit tests on Prodrive's AET system for board assembly defects

van Schaaijk, H.A.H.

Award date:
2017

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Technische Universiteit Eindhoven
University of Technology

Department of Electrical Engineering
Electronic Systems Research Group

Automatic Generation of In-Circuit Tests on Prodrive's AET System for Board Assembly Defects

Master Thesis

H.A.H. van Schaaijk
0873871

Supervisors:
dr.ir. Bart Mesman
ir. Erik Jan Marinissen PDEng.
dr. Alex Alvarado Segovia
ir. Martien Spierings

Eindhoven, Sunday 16th July, 2017

Abstract

Automatic Generation of In-Circuit Tests on Prodrive's AET System for Board Assembly Defects

by H.A.H. van Schaaijk

Prodrive has created its own modular inline test equipment that can be used to detect printed circuit board assembly defects. The latest addition to this modular system is an in-circuit testing extension card, with the intention of adding in-circuit test capabilities to the system. In-circuit testing is an electrical testing technique that verifies if the assembled circuit board complies with the schematic design of the product. Now that the hardware is finished Prodrive needs a tool to automate the process of generating the ICT tests for the device. However, generating these tests have proven to be difficult and solutions proposed by the industry are flawed. This research proposes to use graph theory to automatically propose effective tests. The effectiveness is determined and optimized by means of its PCOLA-SOQ score

keywords: PCBA, ICT, DFT, Graph Theory

Contents

1 Introduction **1**

2 Printed Circuit Board Assembly **2**
2.1 Printed Circuit Board 2
2.2 Electronic Components 3
2.3 PCB Assembly 3
2.4 PCBA Defects 4

3 Related Prior Work **7**
3.1 Defect Classification and Test Coverage 7
3.2 Defect Detection Methods 9
3.2.1 Automated Optical Inspection 9
3.2.2 Automated X-Ray Inspection 9
3.2.3 Functional Test 9
3.2.4 In-Circuit Test 10
3.2.5 Boundary Scan 12

4 The Prodrive AET-ICT System **15**
4.1 AET System Architecture 16
4.2 ICT with the AET 17
4.2.1 ICT Hardware 17
4.2.2 ICT Test Generation 19

5 Problem Definition **21**
5.1 Objective 21
5.2 Software Tool 22
5.3 Project Scope 22

6 System Inputs **24**
6.1 Reference Designator Specification 24
6.2 Board Properties 25
6.2.1 Netlist 25
6.2.2 Component Database 25
6.2.3 Maximum Save Voltage 26
6.3 AET-ICT Description 26

7 Netlist Handling **29**

8 Test Generation for PCL-O of Passive Components **31**
8.1 Graph Decomposition 31
8.1.1 Connected Components and Biconnected Components 32
8.1.2 Graph collapsing 34

8.1.3 Minimal-Guards-Set Finding . . . 35
8.1.4 Passive Decomposition . . . 38
8.2 Test Mapping . . . 43
8.2.1 Generic Test Method . . . 43
8.2.2 Library Mapping . . . 49
8.2.3 Passive Devices Test Mapping . . . 52

9 Graph Decomposition for Boundary Scan **55**

10 Experimental Results **57**
10.1 Experimental Setup . . . 57
10.2 System Input Handling . . . 59
10.3 Passive Decomposition . . . 60
10.3.1 Connected Components, Blocks, and Collapsing . . . 60
10.3.2 Minimal Guards . . . 62
10.3.3 Mold matching . . . 64

11 Discussion, Future Work, and Conclusion **67**
11.1 Discussion . . . 67
11.1.1 Future Work . . . 67
11.1.2 Conclusion . . . 69

Bibliography **71**

Appendix **73**

A Math Properties of False Paths **73**

Acronyms

Acronym	Meaning	Pages
AC	alternating current	18, 68
AET	automated electrical test	15–17, 20–22, 26, 44, 46, 47, 57, 58
AOI	automated optical inspection	9
AXI	automated X-ray inspection	9
BGA	ball grid array	9
BS	boundary scan	11–13
BTC	board test coverage	7, 46
CNFP	compound no false path	43
CUT	component-under-test	10
CWFP	compound with false path	43
DC	direct current	18
DFT	design-for-test	21
DUT	device-under-test	9–11, 13, 16, 69
EDIF	electronic design interchange format	25
ESD	electro-static discharge	5
ETS	electrical test system	16–20
FT	functional testing	9, 11
GUI	graphical user interface	22
IC	integrated circuit	1, 3, 6, 21, 30, 55, 59, 68
ICT	in-circuit test	10–13, 15, 17–22, 24–26, 29, 34, 43, 57–59, 67–69
INFP	isolated no false path	43
IWFP	isolated with false path	43
KCL	Kirchhoff's current law	44
KVL	Kirchhoff's voltage law	44
MNA	modified nodal analysis	46, 48, 52, 68

PCB	printed circuit board	1–4, 7, 10, 11, 16, 19, 21–24, 26, 29, 30, 43, 57
PCBA	printed circuit board assembly	2, 3, 9
PSU	power supply unit	9, 61
refdes	reference designator	24, 25, 30
SMT	surface-mount technology	3, 6, 10, 17, 21, 26
SUT	structure-under-test	11, 26, 31–33, 35, 36, 45–47, 50–52, 62, 69
TAP	test access port	13
THT	through-hole technology	3, 4, 6, 10, 57

Glossary

black-box test Black-box testing is a method of testing that examines the functionality of a product without observing its internal structures or workings. Even if the internal structure is known by the test engineer, if only the externals are observed is still considered black box. 9, 11

defect A physical anomaly. 9

false path All paths that form a circuit from source to measurement during electrical testing, but that are not part of the CUT or SUT. False paths usually have a negative influence on the measurement if no precautions are taken. 10, 35

fault A behavioral anomaly. 9

guard An ICT technique to mitigate the influence of false paths. 10, 17–19, 35

pogo pin A spring-loaded pin used in electronics to establish a (usually temporary) connection between two printed circuit boards. 10, 12, 16

reference designator A reference designator unambiguously identifies a component in a netlist or on a printed circuit board (PCB). The reference designator usually consists of one or two letters followed by a number. 24

test fixture A device or setup designed to hold the device-under-test in place and allow it to be tested by being subjected to controlled electronic test signals. For ICT this usually consists of two parts; the top fixture that moves down to clamp the DUT in place and the bottom fixture holding the pogo pins through which the stimuli is applied. 10, 12, 16, 25

test pad A small copper (or other metal) area on the PCB dedicated to providing probe access to electrical test equipment. 10, 25

white-box test White-box testing is a method of testing that tests internal structures or workings of a product, as opposed to its functionality (i.e. black-box testing). 10, 24, 67

Chapter 1

Introduction

In the process of assembling printed circuit boards (PCBs) assembly defects can occur in every step of this process. Common examples of defects are open circuits, weak solder connections, and short circuits. Preferably these these defects are detected before products are shipped to customers.

To detect these defects a broad range of test equipment exist. Some use vision techniques, others use electric test methods, or other techniques. Every testing techniques has its own strengths and weaknesses. How well a test of a certain tester is can be expressed in the board score. This method combines certain defect properties into a single score.

In-circuit testing excels at testing the Presence, Correctness, Liveliness and Opens defects. In-circuit testing is an electrical testing technique that uses a so called bed-of-nails fixture to make electrical contact between the test equipment and the electric wires on a circuit board. However, since it tests components that are already assembled on the board the surrounding components will influence the measurements if no precautions are taken.

Prodrive has built its own tester; the AET. Originally this machine was not built to execute ICT tests. However, the AET has a modular design through which new functionality can be added. Prodrive designed a ICT extension card to add the ability to execute ICT test to the machine.

Prodrive does not have a software tool to automatically generate real ICT tests based on the design files of a product. Prodrive has tried to implement self-learning principles that learn the impedance pattern of a known-good-board, but this approach did not satisfy. Therefore this research is conducted to find out if such a tool can be created.

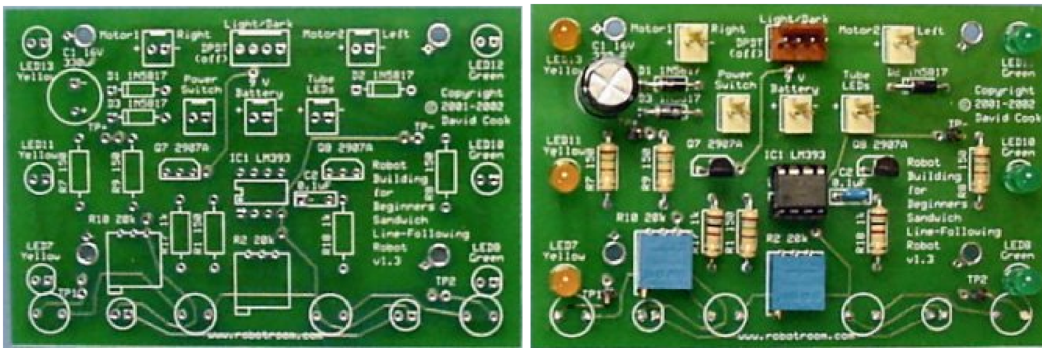
The tool uses the product design files, a component database, and test equipment description files as its input. The tool will compile the information subtracted from these inputs into a graph. Using graph theory and some new algorithms the tool will proposal for an ICT test. Since this proposal can be created based on the design files, the test can be optimized before the circuit boards and components are already ordered.

This thesis starts with describing the environment in which in-circuit testing is used, listing the related prior work, and describing Prodrive's AET machine. Then in chapter 5 the problem is described more thoroughly and are the research questions stated. In Chapter 6 the system inputs are described. In Chapter 7 it is described how the information form these inputs are combined to form the graph. Chapter 8 introduces the algorithms that are used to map tests tho this graph. Chapter 9 shows using a proof-of-concept that the graph could also be used to find the interconnects of integrated circuits (ICs) that can be used to create boundary scan tests. Finally in Chapter 10 the results of the algorithms are shown.

Chapter 2

Printed Circuit Board Assembly

Printed circuit board assembly (PCBA) is the process of populating a PCB with electronic components. Figure 2.1 shows the same PCB *before* assembly (Figure 2.1(a)) and *after* assembly (Figure 2.1(b)). This chapter gives an overview of the different types of components, describes the manufacturing process, and describes the defects that can occur during that process.



(a) Bare PCB.

(b) Assembled PCB.

Figure 2.1: An example of a bare and assembled PCB.

2.1 Printed Circuit Board

A bare PCB is a board used to electrically interconnect electronic components. The PCB contains pin-pads to which the components are mounted and metal tracks (*nets*) to interconnect these components. Simple PCBs could have all the tracks on a single side of the board, but more complex boards typically use multiple layers interconnected by so called vias. These vias can be (1) through hole, (2) blind, or (3) buried. Through hole vias are vias that connect the the top, bottom and all layers in between. Blind vias are exposed only on one side of the board. Buried vias only connect internal layers without being exposed on either surface.

Figure 2.2 shows the cross section of a PCB. It shows the pin pads to which components will be assembled, multiple metal tracks connecting the components on multiple layers and the three types of vias.

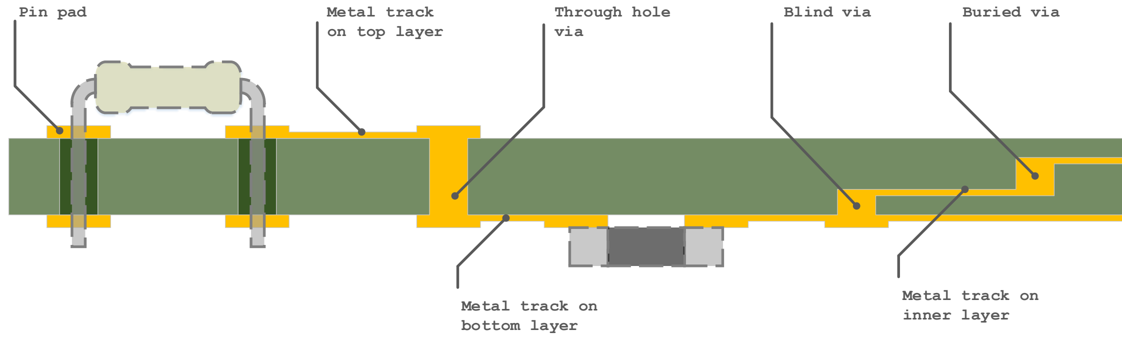


Figure 2.2: Cross section of a PCB.

2.2 Electronic Components

Electronic components can be classified as either passive, active, or electro-mechanical. Passive components are components that cannot introduce energy into the system; examples are resistors, capacitors, and inductors. Active components rely on a source of energy and can inject power into a circuit. Examples of active components are diodes, transistors, and ICs. Electro-mechanic parts often have moving parts, such as relays.

Components are available in different kinds of packages. Components are packaged as stand-alone (*discrete*) devices, arrays, or as circuits. An array is a number of the same isolated components packaged into a single device. In a circuit the devices are interconnected. Examples of passive discrete, array, and circuits are a stand-alone resistor, a resistor array, and a voltage divider respectively. Examples of different packaged active components would be a single transistor, a package containing multiple and-ports, and a microprocessor.

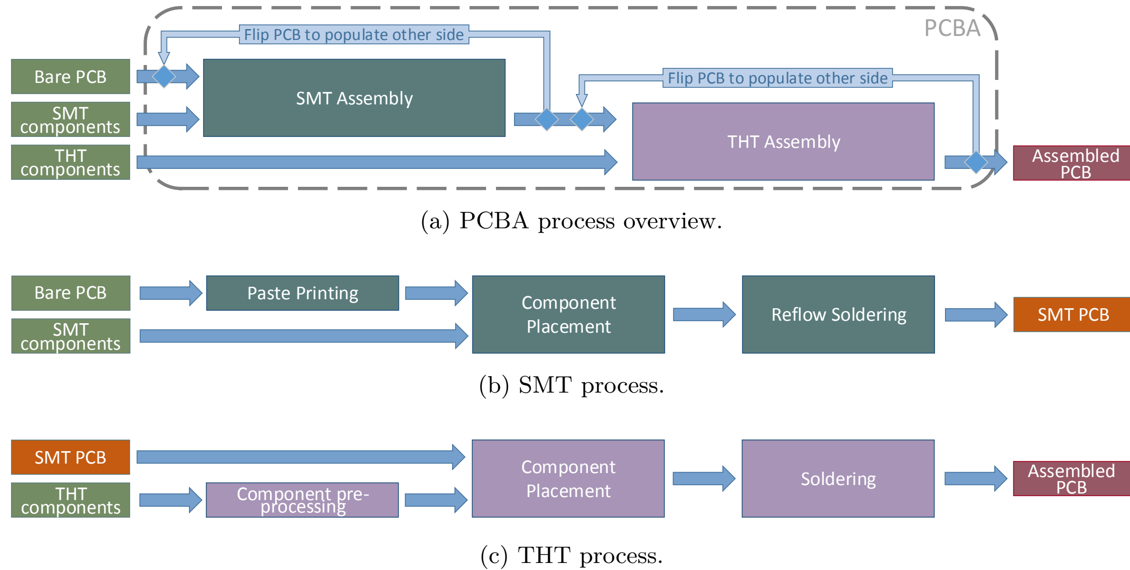
Packaged components have two main mounting methods, viz. surface-mount technology (SMT) and through-hole technology (THT). SMT components are placed directly onto the surface of the PCB, while THT components have leads that are inserted into holes drilled into the PCB.

2.3 PCB Assembly

The PCBA process has two inputs; (1) known-good PCBs, and (2) known-good components. The output of the PCBA is assembled PCBs. Assembly exists of two main parts; one in which SMT components are assembled, and one in which THT components are assembled. The SMT components are usually much smaller, and do not have leads that need to be inserted into holes in the PCB and therefore their assembly process is often easier to automate. However, THT components are sometimes inevitable in the PCB design, for example when large currents are needed.

Figure 2.3(a) shows an overview of the manufacturing process. The bare PCBs, the SMT components, and the THT components are coming in from external suppliers. If the product contains components of a certain technology on both the top and bottom layer, the PCB has to pass the manufacturing process twice. Once the PCB has passed both SMT and THT assembly, the board is fully assembled.

The process of populating SMT components onto a PCB typically consists of three main steps: (1) solder-paste printing, (2) component placement, and (3) reflow soldering. This is shown in Figure 2.3(b). In the paste-printing step solder paste is applied on the PCB to the pads where the SMT components will be placed. In the next step the components are placed in the correct



(a) PCBA process overview.

(b) SMT process.

(c) THT process.

Figure 2.3: A simplified overview of the PCBA process.

position on the board. In the last step the PCB with the applied solder paste and components are heated in order to melt the paste. A physical and electrical connection is created once the melted solder paste solidifies again.

Assembling THT components to a PCB consists of three main steps, as shown in Figure 2.3(c). First the components are pre-processed to make sure that the leads of the components will fit into the PCB. Subsequently the leads of each components are inserted into the holes destined for that component, and finally the components are soldered to the board.

2.4 PCBA Defects

In the assembly process, problems can occur. If these problems result in a physical anomaly that is an unacceptable deviation from the norm, these are called assembly defects. A behavioral anomaly as a result of such a defect is known as a fault.

Table 2.1 shows the most common assembly defects [1]. Some of the risk of these defects can be minimized by taking precautions during design time and assembly, but fully preventing all defects is impossible. The earlier in the process a assembly defect is found, the lower the costs resulting from this defect will be. This means that to keep these costs low, products should be thoroughly tested. Of course creating and executing tests introduce costs as well, so for each product individually the costs versus benefits of testing should be considered.

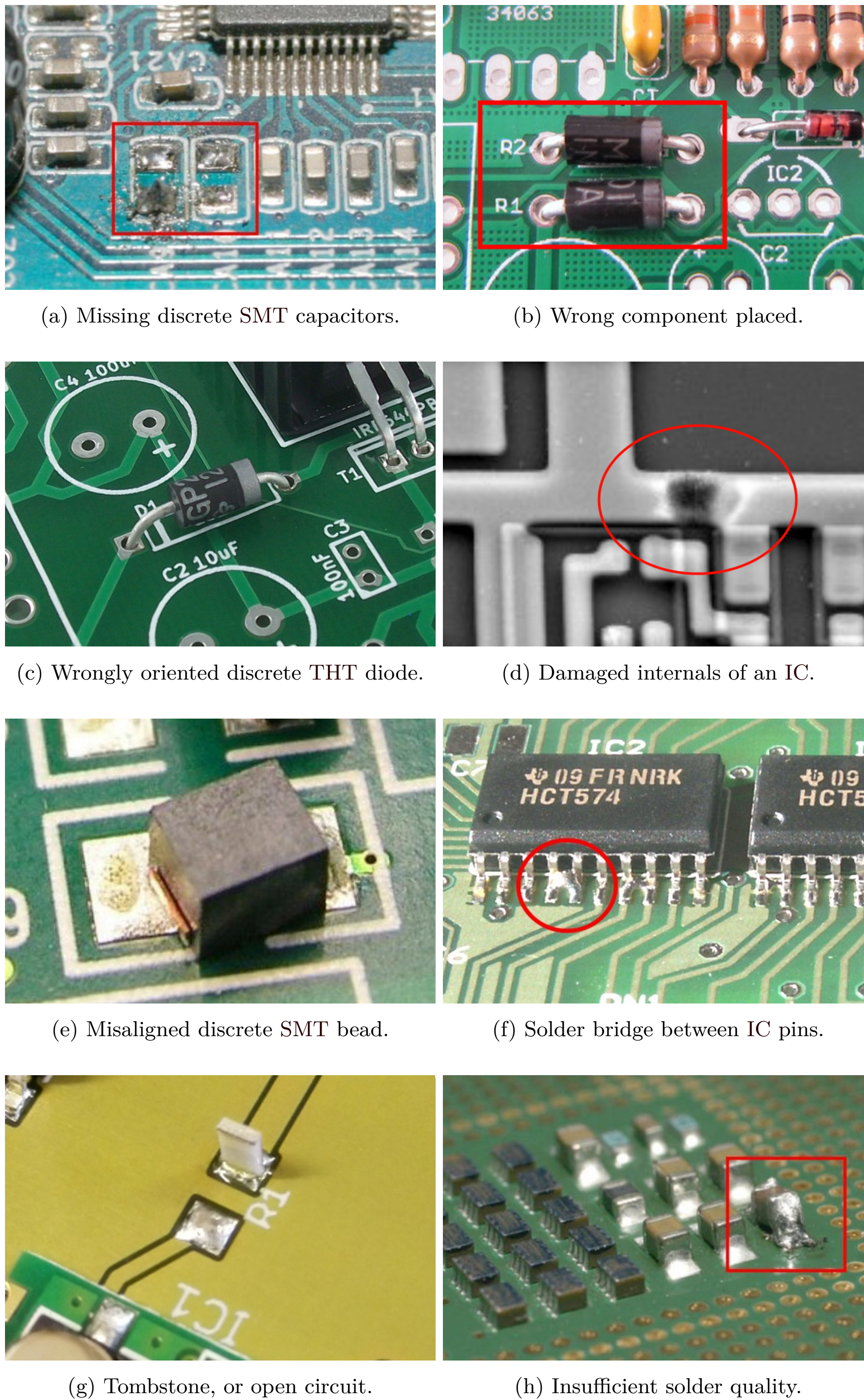
Some examples of common manufacturing defects, shown in Figure 2.4, are the following.

- A component is not placed or accidental removed from the board in between manufacturing steps. This could for example happen when components accidentally get detached by their own weight when their solder connections are melted again the second time the PCB passes the reflow oven to mount components on the other side. (See Figure 2.4(a).)
- A wrong component can be placed due to an operator putting the wrong tape, or incorrectly labeled tape, into the pick and place machine. (See Figure 2.4(b).)

Table 2.1: Common defect types.

Defect Type	Approximate Occurence Rate	Solder-Related
Open circuit	25%	Yes
Insufficient solder	18%	Yes
Short circuit	13%	Yes
Missing electrical component	12%	No
Misaligned component	8%	Yes
Defective electrical component	8%	No
Wrong component placed	5%	No
Excess solder	3%	Yes
Missing non-electrical component	2%	Yes
Wrongly oriented component	2%	No
Defective non-electrical component	2%	No
Excess solder	2%	Yes

- A polarized component is oriented incorrectly during the component placement step. This can happen for example when components are manually inserted into the PCB as shown in Figure 2.4(c).
- A component that is not working properly is placed onto the board. This can happen for example if the components are stored or handled incorrectly, or due to electro-static discharge (ESD) as shown in Figure 2.4(d).
- A component is not aligned correctly. This can happen during the component placement step or if the component starts floating on the molten solder. (See Figure 2.4(e).)
- During paste printing the stencil could be misaligned or too much solder-paste is applied causing solder bridges as shown in Figure 2.4(f).
- The tension caused by solder melting in the reflow oven has caused a surface-mounted component to *tombstone* as shown in Figure 2.4(g).
- If the temperature in the reflow oven is incorrect solder might not melt to the desired temperature, causing solder connection of a low quality. An example is shown in Figure 2.4(h).



(a) Missing discrete SMT capacitors.

(b) Wrong component placed.

(c) Wrongly oriented discrete THT diode.

(d) Damaged internals of an IC.

(e) Misaligned discrete SMT bead.

(f) Solder bridge between IC pins.

(g) Tombstone, or open circuit.

(h) Insufficient solder quality.

Figure 2.4: Examples of common manufacturing defects.

Chapter 3

Related Prior Work

3.1 Defect Classification and Test Coverage

PCB assemblers and suppliers of test equipment use a variety of metrics to determine the quality of their tests, making results incomparable. For this reason a standardized way of describing board test coverage that incorporates the effectiveness of a test is proposed, called the board test coverage (BTC) [2]. BTC is divided in device coverage (DC) and connection coverage (CC) and both are subdivided in Fundamental and Qualitative properties. A device is defined as any component that is placed on the board such as an electrical component, connector, or heat sink. Connections are defined as the solder connections between the devices and the PCB. Fundamental properties directly impact the proper operation of a board while qualitative properties may influence the functioning of the product during its lifetime.

Figure 3.1 visualizes BTC. For each device and connection individually all properties can be assigned a weight and should be ranked as ”untested”, ”partially tested”, or ”fully tested”. The list of properties are is the following.

- **Presence** Is there a device present on the board?
- **Correct** Is the correct device placed on the board?
- **Orientation** Is the polarity of the placed device correct?
- **Live** Is the device (grossly) functional?
- **Alignment** Is a device displaced by a distance or an angle?
- **Shorts** Are there unintended electrical connections?
- **Opens** Is a pin of a device electrically connected to its solder pad?
- **Joint Quality** Is the solder connection of high quality?

The coverage metric of the BTC method, the Board Score, consists of the PCOLA scores of each device and the SOQ scores for each connection. The board score is defined as $\mathrm{BS} = (\mathrm{BDS}, \mathrm{BCS})$, where BS is the board score, BDS is the Board Device Sore, and BCS is the Board Connection Coverage. Both the Board Device Score and the Board Connection Sore have a range of $[0,\ 100,000]$. This range is chosen since K. Hird et all [2] reasoned that if the coverage is expressed in a single percentage the impact of each element is to small. The maximum board score is therefore $(100,000,\ 100,000)$.

A device can have a single or more connections. For each device the PCOLA-properties are included once in the total score, while the SOQ properties are included for each of the terminals individually. This means that for equal weighting the SOQ-properties get dominant if the number of terminals on a device is high.

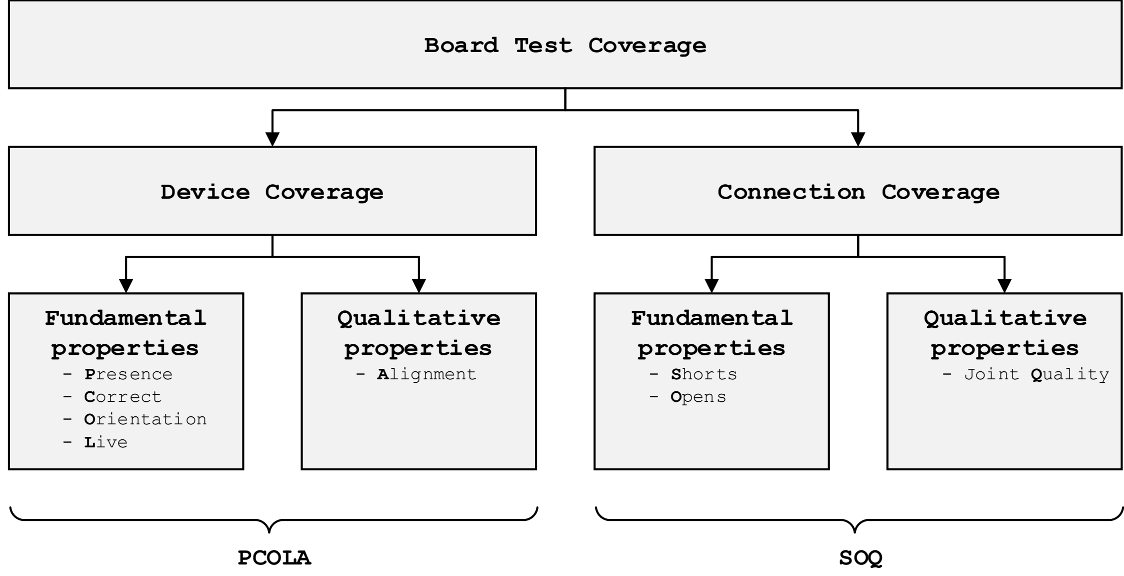


Figure 3.1: PCOLA-SOQ Board Test Coverage overview [2].

TEST M_1:	P = Full,	C = Partial,	O = Full,	L = Partial,	A = Untested
TEST M_2:	P = Full,	C = Full,	O = Full,	L = Untested,	A = Full
S_x(U23):	P = Full, (1)	C = Full, (1)	O = Full, (1)	L = Partial,(0.5)	A = Full (1)
W_x(U23):	0.25	0.25	0.15	0.25	0.1

RDS(U23) = 1·0.25 + 1·0.25 + 1·0.15 + 0.5·0.25 + 1·0.1 = 0.875

Figure 3.2: Example RDS of device U23 tested with two test methods.

For a single device d in the set of all devices $\mathbb{D}$ the raw device score can be calculated by summing up all the score of all the weighted PCOLA properties. Hence the raw device score $\mathrm{RDS}(d) = S_\mathrm{P}(d) \cdot W_\mathrm{P}(d) + S_\mathrm{C}(d) \cdot W_\mathrm{C}(d) + S_\mathrm{O}(d) \cdot W_\mathrm{O}(d) + S_\mathrm{L}(d) \cdot W_\mathrm{L}(d) + S_\mathrm{A}(d) \cdot W_\mathrm{A}(d)$, where $S_\mathrm{x}(d)$ is the score and $W_\mathrm{x}(d)$ the weight for some PCOLA-property x. The raw connection score RCS is calculated in the same fashion, but uses the SOQ-properties instead. If more than one test is conducted on a device or connection, the maximum value of each of the properties is taken, where ”untested”, ”partially tested” or ”fully tested” are defined as 0, 0.5, and 1 respectively.

An example of how the scores of multiple tests are combined is shown in Figure 3.2. It shows a device U23, which will be tested by two test methods; M_1 and M_2. The alignment is untestable with M_1, but as it is fully tested with M_2 it still gets the score ”fully tested” as its S_A. The RDS of U23 in this example. Now, the BDS and BCS can be calculated by again summing up the weighted RDS and RCS of all the devices and connections, i.e.:

$$\mathrm{BDS} = \sum_{d \in \mathbb{D}} \big[\mathrm{RDS}(d) \cdot W(d)\big], \quad \mathrm{BCS} = \sum_{c \in \mathbb{C}} \big[\mathrm{RCS}(c) \cdot W(c)\big]$$

3.2 Defect Detection Methods

The PCBA defects should be detected before products are shipped. There are multiple detections method to find these defects. In this section the most commonly used detection methods and their effectiveness are discussed.

3.2.1 Automated Optical Inspection

Automated optical inspection (AOI) [3] is a technique that relies on vision technology. This technique scans the product for visible defects such as missing components, wrongly oriented components, solder bridges etc. If a component is labeled, it might even be possible to check if the correct component is placed.

Problems arise with AOI when components or solder joints are hidden, which for example happens when parts like ball grid arrays (BGAs) are used that have its solder joints placed under the component itself, or when the device is completely hidden under for example a heat sink. In these cases there is no way to optically inspect the defects.

3.2.2 Automated X-Ray Inspection

Automated X-ray inspection (AXI) [3] is comparable to AOI with the difference that it relies on electromagnetic radiation instead of visible light. Because of this, AXI focuses on the solder connections instead of the packaging. By inspecting these connections it can verify if components are present, aligned correctly, if there are shorts or opens, and if the quality if the connection is sufficient. As opposed to AOI it can even do this with BGAs and other packages with visibly hidden connections. However, since it focuses on the connections rather than the package it cannot determine if the correct part is placed.

3.2.3 Functional Test

Functional testing (FT) is an electrical test method that is, as opposed to the other test methods, a black-box test method and focuses on testing if the device-under-test (DUT) functionally meets its requirements. The DUT is connected to test equipment and is then tested by a series of tests written by an engineer who has a thorough understanding of the product. Functional test generation is therefore much harder to accomplish than the other test methods in this chapter. Furthermore, assigning PCOLA-SOQ scores to a FT is hard since only the in and outputs of a system are observed with a limited number of test cases.

A problem with FT is that even though a product is correctly working at test time, it can still include mechanical poor connections or even include wrong components. As long as these defects do not cause a fault at test time, the defect will not be found. However, the defect could degrade the product over time, or hit a faulty state that was not covered in the test procedure, causing problems when the product is already shipped.

Other problems with FT are concerning the cost of products failing an FT test. In order to functionally test a product it has to be powered. This means that if an expensive product contains fatal defects -such as a short circuit in the power supply unit (PSU)- the complete product is lost. When a non-fatal defect is found, pin-pointing the defect is hard. The only information that could be derived from a product failing a FT test is that it is not functionally correct, not which component is causing the defect, meaning a skilled engineer is needed for further investigation.

3.2.4 In-Circuit Test

In-circuit test (ICT) [4] is an electrical test method that often uses a so called *bed-of-nails* test fixture to electrically connect the nets of a DUT to the ICT test equipment using spring-loaded pogo pins. ICT is a white-box testing method used to verify the assembled PCB complies with the schematic circuit of the product. In contrast to the other electrical tests in this overview, ICT has the advantage that it does not require the board to be powered to execute a test. This way most discrete components can be tested for presence, correctness, liveliness, opens, and shorts. In some cases, such as with diodes, the orientation can be tested as well.

Nets can be reached by probing on dedicated test pads, on the leads of THT components, or on vias. Probing the pads of SMT components is unreliable since the pressure of the pogo pin might temporarily fix an open. Probing on the top layer is possible but often not preferred since then electrical signals have to be routed through the moving top fixture. If the amount of reachable nets -nets which can be connected to a pogo pin- is high, ICT should result in an effective test. However, dense products with no room for test pads are not suitable for ICT testing.

Figure 3.3 shows a PCB with three components and four nets. In the figure there are four probes needed to reach each net. P2 can be placed on a dedicated test pad or on the right lead of C1 or left lead of L1. P3 can be placed on the right lead of L1 or on the via in between L1 and R2. The via can be probed from the bottom or the top, but as mentioned before probing form the bottom is preferred. Probing the last (rightmost) net is not possible since there are no test pads and probing the leads of R2 is impossible since it is an SMT component.

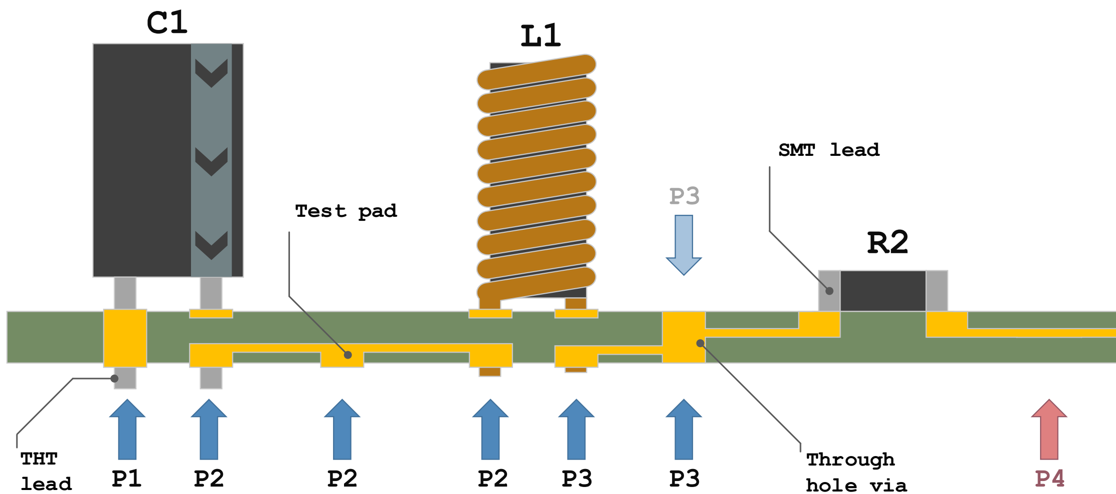


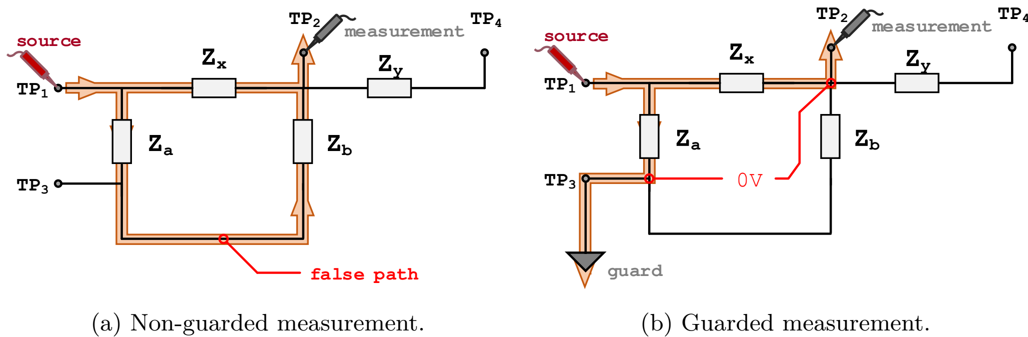
Figure 3.3: Probe access overview.

Since ICT tests the components on assembled PCBs, the components surrounding the component-under-test (CUT) might influence the test, if no precautions are taken. The biggest influence is caused by so called false paths of the CUT. A false path is a electrically conducting path on the PCB parallel to the CUT. Since these false paths are in parallel with the CUT they influence the measured impedance between two test points.

A technique used with ICT to mitigate the influence of these false paths is guarding. The guard is the reference voltage of the source and measurement circuits of the ICT test equipment. When a net in a false path is connected to this reference voltage no current will flow through the remaining part of the false path since now there is no voltage drop over the part between the guarded net and the net connected to measurement. Figure 3.4 shows an example of a measurement for which guarding should be used. Component Z_x is the CUT and can be reached by test pads TP_1 and TP_2. As shown in Figure 3.4(a) current will not only flow through Z_x, but unintentionally through

the false path formed by Z_a and Z_b as well. By guarding TP_3 as shown in Figure 3.4(b) there is no potential over Z_b anymore, stopping current to flow through it.

Due to the fact that the input and output impedances of measurement circuits are not ideal, the voltage drop over the guarded components is not exactly zero and therefore some current will flow through it. The *guard ratio* [5], which is the ratio of structure-under-test (SUT)-current to guard-current is therefore a metric of accuracy.



(a) Non-guarded measurement. (b) Guarded measurement.

Figure 3.4: Eliminating false paths through guarding.

Note that to perform these ICT tests the DUT has to be electrically floating. If the ground of the DUT would be the same reference as the test equipment testing of the components connected to ground would be impossible to test and a lot more paths would influence the tests.

Generating ICT tests

ICT tests are often composed by hand, and since ICT needs a test for each individual component on a PCB this is a long and tedious process. An experienced test engineer has to set up a test, assigning a source net, measurement net, voltage, frequency, and all the necessary guard points for each test. Automating this process could result in substantial cost reductions, but has proven to be a difficult task. In the paper New Directions in Loaded Board Testing [6] it is claimed that there is no known computer program or algorithm to configure or assign all the necessary guard points, including those which are necessary but not obvious.

As a solution to this problem could be to *learn* an impedance pattern of a known good board and test production boards against this pattern. This black-box testing method does not depend on guarding, since it effectively combines the values of an unknown group of components, including the components that would be in the false paths. An improved version of this principle, second-generation self-learn [7], extends this approach by trying to guess the type of component that is measured based on a step response. This should result in better diagnostics and fault localization. As described in Section 4.2.2 Prodrive has implemented these self-learning algorithms but found that this is not a viable solution.

Effectiveness of ICT

Since ICT relies on the use of physical probe contact to test PCBs, it is sometimes referred to as being an *intrusive* board testing method, while boundary scan (BS), and FT are *non-intrusive* [8]. This intrusiveness refers to the need of external access from as much nets as possible. This is a problem since the access needed by ICT has been diminishing significantly since integrated circuits with high-speed interconnects on dense boards are emerging.

Next to being intrusive, in-circuit testing is also considered to be expensive since it often relies on test fixtures. These fixtures can contain thousands of pogo pins which are often all connected by hand using long wires that cause unreliable connections and crosstalk.

Shorts Testing with ICT

The ICT methods discussed thus far mainly focus on device coverage (Section 3.1). However, as can be seen in Table 2.1 short circuits are a common defect as well. ICT testers are often capable of executing shorts tests [9].

There are two existing methods for shorts testing; the linear method and the binary approach. In the linear method all nets are connected to a single voltage source excluding one, which is connected to a measurement probe. If the current that flows from source to measurement is smaller than a predefined setpoint, there is no short from the single net to each other net. By iterating over all the nets, all shorts can be found.

In the binary method half of the nets are connected to source, while the other half is connected to measurement. If the current that flows form the source to the measurement group is smaller than the setpoint then there can be no short between nets in the one group and nets in the other group. However, shorts can still exits within the groups themselves. By subdividing each of the groups again in two groups, doing the measurement again and repeating this process until the groups can no longer be divided further, all shorts can be found in only $\log_2(n)$ tests, where n is the number of nets.

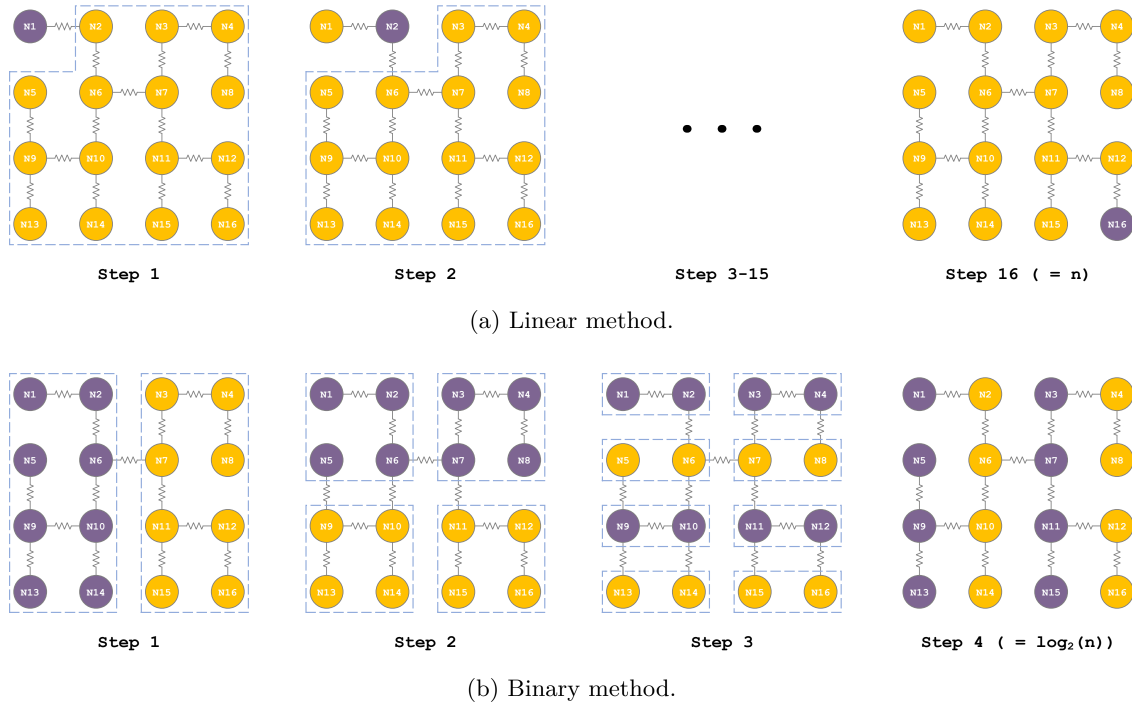
Figure 3.5 shows a example circuits to which both the linear testing (Figure 3.5(a)) testing and binary testing (Figure 3.5(b)) is applied. It shows a netlist consisting of 16 nets, 14 components, and no shorts. Vertices connected to the source are displayed yellow, vertices connected to a measurement probe are purple. Steps 3 till 15 of the linear method are not shown. Vertices which can still be shorted together after a test step is executed successfully are encircled.

As can be seen from the image some current could flow from source through the components on the board to the measurement probe(s). For example, in step 1 of the binary method there is one component that has one lead connected to a source probe while to other lead is connected to a measurement probe. In step 2 and 3 this holds for three components. If there are too many components in a test step that do this, then the current through all of these components might be bigger than the setpoint. This is known as a phantom short.

Anthony Suto claims that these phantom shorts slow down the binary search routine dramatically [9]. A phantom short will trigger the search routine of the binary method. As it divides the groups the number of parallel components in each group declines, and thus the impedance increases again, making the apparent short vanish. Therefore it is proposed to use the linear method as a prefilter for the binary method by allowing the tester to eliminate nodes from consideration by creating a shorted-node list as input for the binary method.

3.2.5 Boundary Scan

Boundary scan (BS) [10] can be used to test interconnects of devices supporting IEEE std. 1149.1 [11]. Using the test equipment embedded in the silicon of these devices each pin can be set to a logic '1' or '0' and can be read back. This way stimuli can be applied to each pin and the response captured on any connected IEEE 1149.1 device. Solder bridges would for example cause adjacent leads, and pins of devices connected to that lead, to switch according to the stimuli of the lead it is shorted to. Missing components in a path would mean the stimulus would not be perceptible by the other devices connected to the interconnect. Components in the path of interconnects can thereby be tested to be present. The boundary-scan enabled devices itself can be tested for presents, correctness, orientation, liveliness, shorts, and opens.



(a) Linear method.

(b) Binary method.

Figure 3.5: Linear and binary shorts testing method example for $n = 16$ nets.

Figure 3.6 shows an example of how shorts and opens can be found in between two IEEE 1149.1 devices U1 and U2. The circuit contains two defects; a short between Net 2 and Net 3, and a open in Net 4. There is no defect concerning Net N1, hence the stimulus applied by U1 on this net are received by U2 without changes. The response received on of Net N2, N3, and N4 are not equal to the stimuli applied by U1, hence there have to be defects on these nets.

A large number of boundary scan test generation algorithms exist [12]. Most of these algorithms cover single-net shorts and multi-net short defects as shown in the example above. However, some algorithms are more extensive then others, and some create more concise test patterns for faster execution.

Boundary scan has the advantage of not needing external access to every net. As mentioned in Section 3.2.4 the effectiveness of ICT is descending due to the fact that probe access is diminishing. Boundary scan does not suffer from this diminishing access since a single test access port (TAP) can be used for multiple devices.

Boundary scan however does need *internal* probe access; the nets do not need access from external test equipment but from one ore more IEEE 1149.1 devices. This means that boundary scan is not a viable option if large parts of the design do not include such devices. Another problem with this strategy is that it is not in every case possible to detect that a wrong component is placed. For example, a smaller resistor might logically still act the same but does allow for a bigger current which could degrade the product over time. Just like the functional test, the DUT has to be powered, meaning expensive components could get damaged in case of a defect.

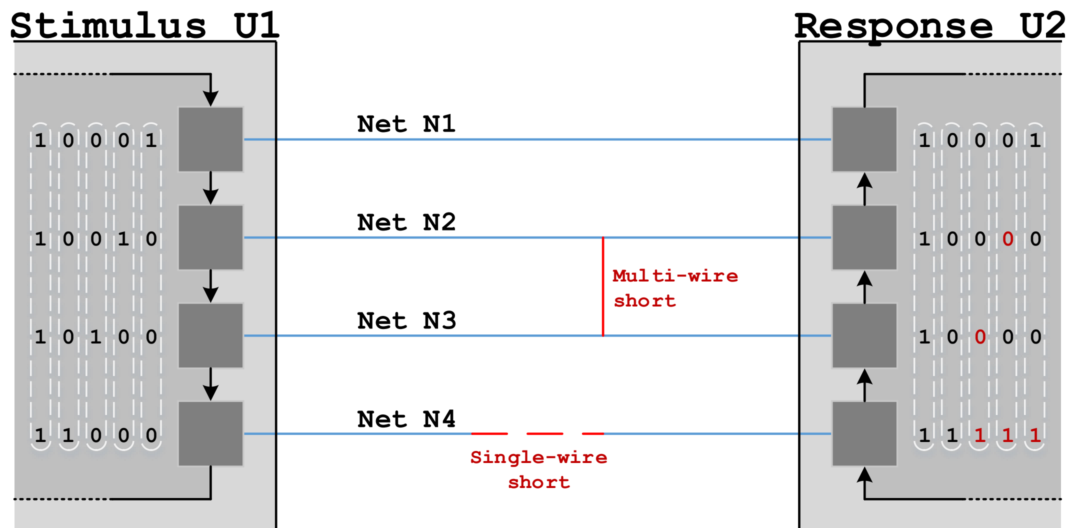


Figure 3.6: Boundary scan example.

Chapter 4

The Prodrive AET-ICT System

The automated electrical test (AET) system of Prodrive Technologies is an test system capable of performing analog, digital, power, high speed, and audio/video tests using a product specific probe nail bed. Test functionality can be added to using generic extension cards. This machine, shown in Figure 4.1, is designed to meet Prodrive Technolgies its desire for a more reliable and flexible test machine that does not need manual handling, has support for high speed signals, and has a short test development time.

The latest addition to the AET system is a ICT extension card. This extension card will allow the AET to be used as an in-circuit tester. In this chapter an overview of the AET and the new ICT capabilities are given.

Figure 4.1: The inline Automated Electrical Test system.

4.1 AET System Architecture

The electrical test system (ETS) is the heart of the AET. It is a 19 inch rack containing ETS slots, a backplane and Ethernet switch, power supply and fan tray, for holding, interconnecting, powering, and cooling the hot-swappable stimuli cards. These product-independent ETS stimuli cards supply all the test functionality to the system. Using connectors the ETS rack connects directly to the product-specific wireless test fixture.

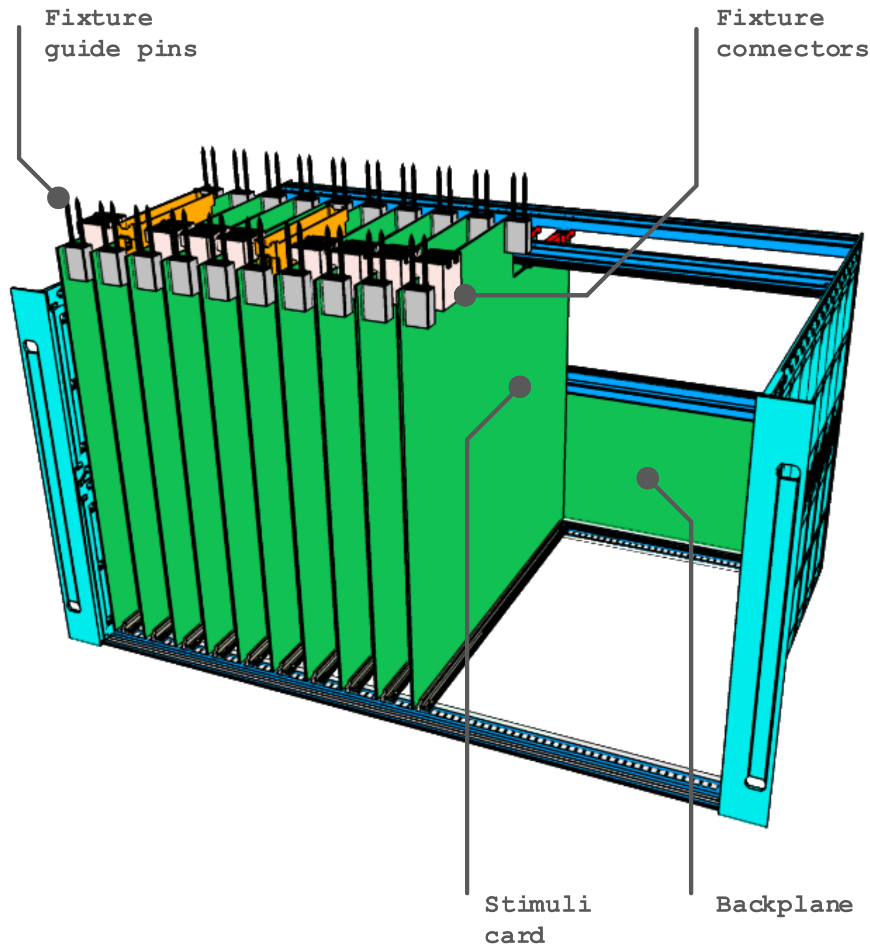


Figure 4.2: The Electrical Test System.

Wireless fixtures [13] were introduced to address the limitations associated with wired fixtures by replacing the nest of signal wires by a single translator PCB. As mentioned in Section 3.2.4 existing in-circuit testers require these unreliable and expensive bed-of-nails test fixtures. Wireless fixtures make the signal integrity better while also offering better reliability due to the absence of cables. Wireless fixtures also exhibit better crosstalk, reflection, and noise characteristics, which are particularly beneficial in low-voltage applications. The AET uses these wireless test fixtures, shown in Figure 4.3. The ETS cards directly connect to the *translator* PCB using the fixture connectors. The translator PCB is a bare PCB that routes signals from extension cards in the ETS rack to the appropriate pogo pin. The pogo pin on its turn connects to a net on the DUT.

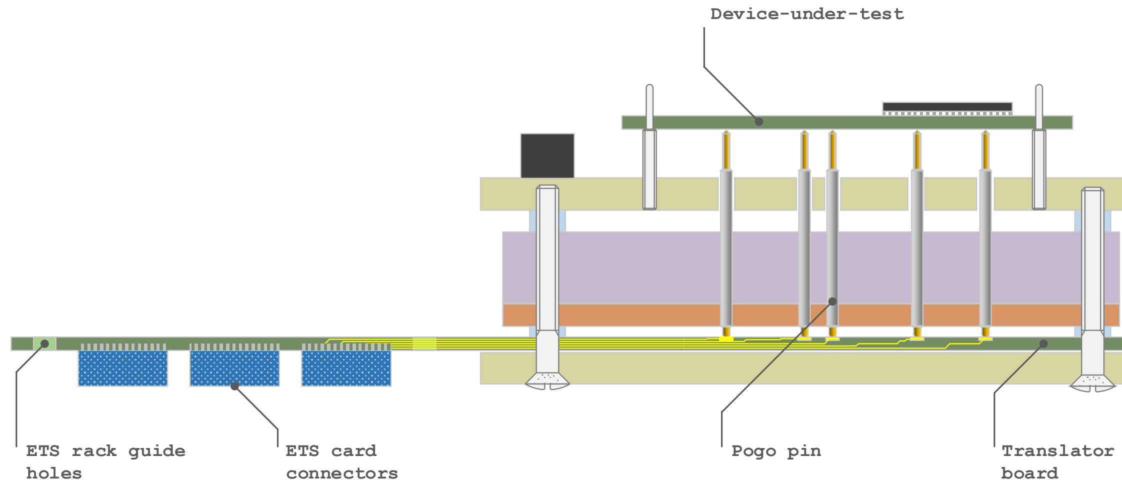


Figure 4.3: Schematic cross section of a wireless fixture.

4.2 ICT with the AET

4.2.1 ICT Hardware

To add in-circuit test capabilities to the AET system a new ICT-ETS card able to perform ICT tests is designed. If the AET is used for ICT, the ETS rack exclusively holds ICT test cards. The ETS rack can take up to 16 ETS ICT modules, each suppling 160 pins, making a total of maximum 2560 test pins. The card is designed in such a way that testing over multiple modules is possible. Each of these 160 pins on a ETS-ICT card can be connected to either a source, measurement, guard, ground, or discharge via a switching matrix.

An abstract overview of the ICT card is shown in Figure 4.4. Every ICT card holds the source and measurements circuits four times, driving 40 pins each. All ETS cards in the rack have the same ground; ETS-GND. However, to perform ICT measurements the sources and the measurement circuits have to be isolated from this ETS-GND. The isolated ground of the source and measurement circuits is known as the guard, see Section 3.2.4.

The idea of the ICT hardware is to keep it simple and reliable. Most existing ICT equipment uses a switching matrix of big relays in order to route the signals of the source circuits to one of the output pins. However, since these relays contain moving parts these matrices are usually slow and get unreliable over time. To overcome this problem the ETS-ICT card mainly uses analog switches. The relays in other ICT equipment do have the advantage of a very low on-resistance. The T-switch, that connects a test pin to the ICT circuitry, needs to have a low on-resistance as well, since it is always included in the source-to-measurement path and will therefore influence the accuracy of the measurements. Because of this for the T-switch a SMT reed relay is chosen instead of an analog switch.

Another consequence of the decision to keep the hardware simple is that only three-wire measurements can be done; i.e. source, measurement, and guard. This means 4 wire Kelvin measurements, and not 4,5, or even 6-wire measurements using sense wires are not possible.

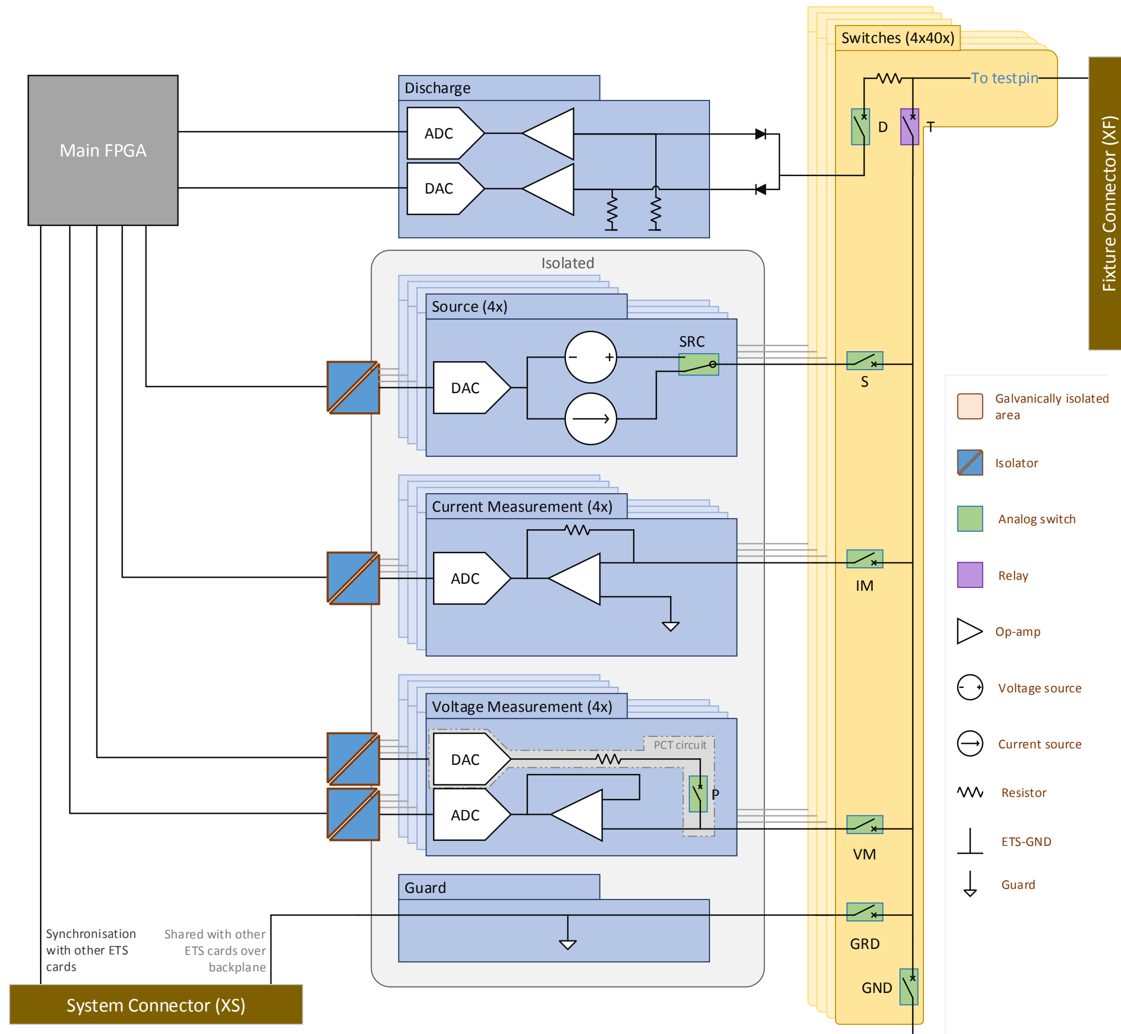


Figure 4.4: Schematic overview of the ETS-ICT extension card.

The source can be set to constant voltage or constant current, both alternating current (AC) or direct current (DC). The sources are isolated from ETS-GND and are referenced to the guard rail instead. In Table 4.1 an overview of the capabilities of a single source is shown.

Table 4.1: Source circuit specifications.

Item	Min	Typ	Max	Unit
Voltage source output voltage	-4	-	4	[V]
Voltage source output current limit	-	30	-	[mA]
Voltage source frequency	0	-	150	[kHz]
Current source output current	-20	-	20	[mA]
Current source voltage limit	-5	-	5	[V]
Current source frequency	0	-	150	[kHz]

The measurement circuits are used to measure the stimuli applied by the source. A single measurement circuit consists of a voltage measurement circuit and a current measurement circuit. The

circuits are isolated from ETS-GND and are referenced to the guard rail instead. Table 4.2 shows the specifications of the measurement circuits.

Table 4.2: Measurement circuits specification.

Item	Min	Typ	Max	Unit
Voltage measurement input voltage	-4	-	4	[V]
Voltage measurement frequency	0	-	150	[kHz]
Current measurement input current	-30	-	30	[mA]
Current measurement frequency	0	-	150	[kHz]

The ETS-ICT card currently supports impedance and $\frac{dV}{dt}$ tests. With the former the impedance of resistors, capacitors, and inductors can be tested. Also low impedance components can be verified using this test method. The $\frac{dV}{dt}$ method is used to test large capacitors. If capacitors are larger than $1\mu F$ the source cannot supply the voltage needed to accurately perform the test.

The accuracy of the measurements are specified in Table 4.3. These accuracies are specified for a bare ICT card, without any parallel components or influences of a fixture.

Table 4.3: Measurement accuracy.

Item	Range	Max Deviation (%)	Method
Resistance error	$[10\Omega,\ 100\Omega]$	30	AC/DC impedance
	$[100\Omega,\ 150k\Omega]$	10	AC/DC impedance
Capacitance error	$[200pF,\ 1nF]$	30	AC impedance
	$[1nF,\ 1\mu F]$	10	AC impedance
	$> 1\mu F$	10	$\frac{dV}{dt}$
Inductance error	$[10\mu H,\ 100\mu H]$		AC impedance
	$[100\mu H,\ 1H]$	10	AC impedance

4.2.2 ICT Test Generation

Prodrive has implemented self-learn principles as described in Section 3.2.4 but found that it not always leads to a desired result. This approach appeared to have two major drawbacks.

Learning patterns from a known good board does not take into account tolerances of the learned components. If a resistor with some value is measured on a board, it is unclear how measuring ranges should be determined. Even when multiple boards are used for learning the impedances the test might not be stable since impedances might change when a different batch of the same components are used. For example, a product can hold a capacitor with a capacitance of $1\mu F \pm 20\%$. If that capacitor is learned as a $800nF$ capacitor since all PCBs used for learning contained capacitors form the same batch with their capacitance in the lower end of the spectrum, the test might fail when a new batch of the same capacitors with values close to the upper limit are used.

Another problem with this approach is that with the first-generation of self-learning principles each impedance effectively combines the values of an unknown group of components, and therefore determining the test effectiveness is hard [7]. The second-generation self-learning principles do improve on this, but do not take into account the problem of diminishing probe access as described in Section 3.2.4. Using the solution proposed by R. Matheson [7] it is not possible to learn what is in between test points if there can be any combination of components.

This method uses testing as an afterthought since a test can only be created when the product has already passed its prototype phase. It might happen that vital parts of a product cannot be

tested without a redesign. Even when an effective test is designed in this way, locating the defects is difficult and time consuming. The only information that can be extracted from a failure is that there is an incorrect impedance in between two points.

The AET firmware is currently not capable of executing shots test. The switching matrix of the ETS-ICT cards are currently set up to only allow the source and measurement circuits to be connected to one test pin. Therefore executing shorts test using neither the linear method nor the binary approach are currently possible with the AET. However, this can be fixed using a firmware update.

Chapter 5

Problem Definition

The main topic of the research is determining the viability of an application that automatically generates effective ICT tests for the AET system. As mentioned in Section 3.2.4 the effectiveness of ICT is decreasing due to the rise of ICs and SMT components. Section 4.2.2 subsequently describes the problems with the self-learning solutions proposed by the industry. The desired result of this research is a proof of concept tool that proposes tests based on the design files of a product rather than on a known good board to allow for design-for-test (DFT) and that is capable of generating effective tests for PCBs that do not have probe access on all nets.

The main and sub research questions are:

- **In what way can ICT tests with an as high as possible PCOLA-SOQ score be automatically proposed based on product design files?**

  1. How should useful information be extracted from the design, and other input files?
  2. How can a given library of test methods be mapped to the components in a network of a given product in a reasonable amount of time?
  3. How can the effectiveness of a proposed tests be optimized and guaranteed?

5.1 Objective

The purpose of the tool is to propose effective ICT tests for a given circuit design. Not only will this enable test engineers to offload big parts of generating production-ready tests, it will also be helpful as a DFT tool.

The tool will make a proposal based on the design files, hence the test coverage can be predicted before the components are ordered and the PCB is finalized. This means that changes to the design can still be incorporated to increase the testability.

The most important objective of this tool is however to shorten test-development time. ICT tests for the AET system are still composed by hand, which is time-consuming and therefore costly. The tests proposed by the tool still need to be checked and verified by a test engineer, but the bulk of the work will be taken care of.

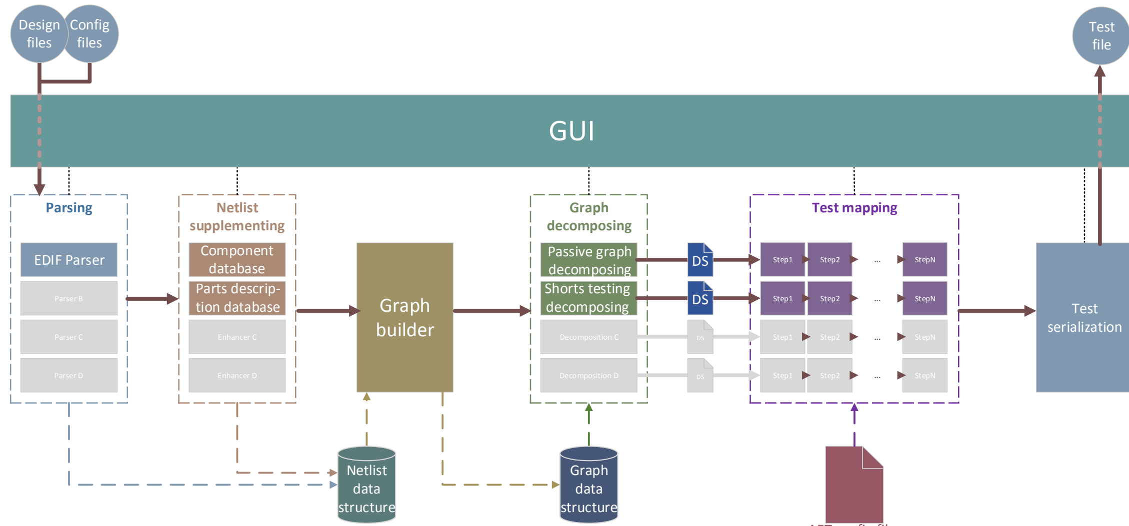


Figure 5.1: Schematic overview of the software tool.

5.2 Software Tool

The proof of concept software tool (written in C♯) will be a graphical user interface (GUI) tool consisting of six main parts, as shown in Figure 5.1.

1. The design files of the PCB and the configuration files of the AET need to be parsed. This is described in Section 6.2.1.
2. The design files do not contain all information needed for designing ICT tests. Section 6.2.2 describes which information is missing and how it is added to the netlist.
3. Since the netlist is not in a format that is easy to compute the netlist is converted into a graph. Chapter 7 describes this process.
4. The graph that is created can be really big and contains data that is not necessary for each kind of ICT test. For this reason the graph is pre-processed (*decomposed*) before the tests get actually mapped. Section 8.1 describes how this is done for mapping tests on passive components.
5. Section 8.2 describe how tests get mapped on the decomposed graph.
6. Finally the test has to be converted into a format that can be executed by the AET.

5.3 Project Scope

The goal of this research is to automatically generate effective ICT tests for the AET. Although a high PCOLA-SOQ score is desired (see section 3.1), it is impossible to get a score of 100.000 by just using ICT. The research will therefore focus on the properties that ICT excels at; Presence, Correctness, Orientation, Liveliness and Opens.

A remark should be made by Correctness. A test could be conducted to approximate a value to validate if the wrong tape is in the pick and place, but it is not in the scope of the research to test if a component has exactly the values as indicated by the supplier. If for example a 5% resistor is measured, for which the ICT card can perform a test with an accuracy of 10%, the test passes when the value is measured in the range of $\pm 15,5\%$ $(105\% \cdot 110\%)$.

All tests have the purpose to find defect introduced at the assembly of Prodrive itself. All incoming components, PCBs and other parts are assumed to be within the specifications of the supplier. Of course it is possible for a test to find defects caused by external suppliers, but this is considered incidental.

This research limits itself to the tool and its mapping functionality itself. The tool hands test engineers the tools to create a library of tests that will be mapped on the actual components in the netlist. This library is not created for this proof-of-concept tool. Only a small test library is created to test the mapping principles.

Chapter 6

System Inputs

In this chapter the files needed to design ICT test are described. As mentioned in Chapter 5 the ICT test will be generated by means of the design files of a product. Section 3.2.4 mentions that ICT is a white-box testing method that tests components individually. Therefore the design files and information about the devices is needed to be able to design an ICT test. Unfortunately the design files do not always contain the information that is required to design a test, meaning that information of multiple sources has to be combined to get all the required information. The minimum required information is:

1. **The reference designator (refdes)** of each component to distinguish components.
2. **The component type** of the components to decide what kind of measurement is needed.
3. **The value** (if component is passive) to check if the right component is placed.
4. **The tolerance of the value** to determine the measurement accuracy.
5. **The mounting technology** to determine if probing of the leads of a component is possible.
6. **The PCB layer** to determine if probing from the top is needed to reach the component.
7. **A description of the internal structure of the component** to determine between which leads should be measured.

6.1 Reference Designator Specification

Every device on a PCB has its own unique identifier. This identifier is called the refdes of the device. A refdes typically consists of one or two letters (the class designation letters, or refdes prefix) followed b a number. In most standards the prefix is a reference to the component type of the device. For example, if the prefix R is reserved for devices of the type resistor, then the devices R101 and R102 will be resistors on the PCB.

However, there are multiple conflicting standards [14] for these refdes prefixes. For example, according to IEC Publication 113-2 a transistor should have the prefix V, while according to IEEE-315-1975 a transistor should use Q. Furthermore, PCB designers can deviate from the standard or use a standard of their own. Therefore the tool has to accept an input file linking refdes prefixes to component types. In this document the IEEE-315-1975 standard will be used.

6.2 Board Properties

6.2.1 Netlist

Netlist are files containing descriptions of the connections of an electronic circuit. Each connection, or net, is the representation of a conducting trace, plane, or wire that interconnects at least two terminals of a component. Each component has its own reference designator, or refdes, which is a unique identifier.

Netlist can be either physical or logical. A physical netlist describes the actual location of each component and the nets that connect their terminals. A logical netlist is only a description of which terminals are connected together with which nets.

For this research only the logical netlists are used. A physical netlist is needed eventually to generate output that can be used by another tool to generate the test fixture for a product. However, this research focuses on generating the tests and therefore the physical netlist are not used.

In this document a netlist is formally defined as a tuple $\mathcal{N} = (C, N)$ where $\mathcal{N}$ is the netlist, C a set of components and N a set of nets. A component $c \in C = (\{t_1, ..., t_n\}, R)$ is defined as a set of its terminals t and a refdes R. A net $n \in N$ is defined as a set of terminals it connects together. Note that a terminal can only be part of a single net since electrically connected parts cannot be separated (see Equation 6.1) and that if all terminals are connected to a net the set of all terminals of C is the same as all terminals of N (see Equation 6.2).

$$\forall_t \left[t \in n_\alpha \wedge t \in n_\beta\right] \Rightarrow n_\alpha = n_\beta \tag{6.1}$$

$$\bigcup_{n \in N} n \subseteq \bigcup_{c \in C} \pi_1(c) \tag{6.2}$$

The electronic design interchange format (EDIF) is chosen as the input format for logical netlists since it is an open standard that is easy to parse. On Page 28 an example of an EDIF netlist generated by OrCAD Capture is shown. It is a netlist of a single resistor R101 that has a test pad connected to it on both sides, TP101 and TP102.

- On line numbers 13-19 a new type of component RES (resistor) is defined that has two terminals &1 and &2.
- On lines 20-25 the type test pad with only one terminal is defined.
- Lines 33-39 define a new resistor of type RES with part number 1100-0002-4701 (line 37) and a value of $4.7k\Omega$ (line 38).
- Lines 40-48 define the two test pads in the same manner.
- Lines 50-57 define two nets N101 and N102. The fist net is a connection between TP101 and the &1 terminal of R101. The other net connects TP102 to terminal &2 of R101.

6.2.2 Component Database

The component information stored in a netlist is not sufficient for generating ICT tests. As can be seen in the example EDIF netlist on Page 28 already provides some extra information, but it still not suffices. Furthermore the information stored inside a netlist may vary from format to format and is therefore not a reliable source. To overcome this problem a database system is used to retrieve missing information.

(a) Orion component example 1.

(b) Orion component example 2.

Figure 6.1: Two examples of information about components saved in Orion.

Most PCB assemblers, like Prodrive Technologies, do already have a central database system in which component information is stored. Prodrive's component database, which is called Orion, however only stores information needed for designing and manufacturing PCBs, not for ICT testing them.

Figure 6.1 shows two examples of components saved in Orion. Figure 6.1(a) shows a discrete SMT resistor. The component type is known, the value and the tolerance are known, and since it is a single resistor the internal structure is known. However, Figure 6.1(b) shows a resistor array. From reading the description it can be concluded that there are both $1k\Omega$ and $10k\Omega$ resistors in a single package, and that the tolerances differ between the components in the array as well. Furthermore, since their are multiple resistors in a single package the internal structure of the package is not known.

Only if the needed information can not be deducted from the information available in Orion, the component is saved in a dedicated ICT description database. Not everything is saved in the database since duplicating information causes the need to maintain the same data in multiple places. The new database will hold description of the package of the components (mounting technology, names of terminals) and an internal description that describes the *inner components* value, tolerance, and to which terminals of the package it is connected.

6.2.3 Maximum Save Voltage

While testing passive components on a PCB the voltage that can be maximally used is determined by the active components surrounding the SUT. This is further described in Section 8.1.4. The voltage that can be safely used varies by the logic family (e.g. TTL, CMOS, NMOS) of the active components used on the board. A global maximum safe voltage can be entered by a test engineer.

6.3 AET-ICT Description

The AET-ICT hardware described in the previous chapter is the first version of the ICT hardware. New versions with different capabilities or with an higher accuracy might follow. The tool should be able to, in some extend, cope with future versions. Of course the tool will not be prepared to be able to cope with 6-wire measurements just in case this will be ever added to the hardware, but an expansion of the number of test pins should be no problem. The specifications of Section 4.2.1 are compiled in a configuration file that is used as an input for the tool.

Parasitic impedances of the test equipment (co-) determine the accuracy of the system. In Section 3.2.4 this is described more thoroughly. For this reason an estimate of these impedances will help the tool determining the measurement ranges for each of the components-under-test.

```
1   (edif ED2
2    (edifVersion 2 0 0)
3    (edifLevel 0)
4    (keywordMap (keywordLevel 0))
5    (status
6     (written
7      (timeStamp 2017 01 02 08 56 46)
8      (program "CAPTURE.EXE" (Version "16.6.0.d001"))
9      (comment "Original data from OrCAD/CAPTURE schematic"))
10    ...)
11   (external OrCAD_LIB
12    ...
13    (cell RES
14     ...
15     (view NetlistView
16      (viewType netlist)
17      (interface
18       (port &1 (direction INOUT))
19       (port &2 (direction INOUT))))))
20    (cell PCB_ICT_D0100_REF
21     ...
22     (view NetlistView
23      (viewType netlist)
24      (interface
25       (port &1 (direction INOUT)))))
26  (library MAIN_LIB
27    ...
28    (cell ED2
29     (cellType generic)
30     (view NetlistView
31      ...
32      (contents
33       (instance R101
34        (viewRef NetlistView
35         (cellRef RES
36          (libraryRef OrCAD_LIB)))
37        (property Part_Number (string "1100-0002-4701"))
38        (property Value (string "4.7kR"))
39          ...)
40       (instance Q101
41        (viewRef NetlistView
42         (cellRef PCB_ICT_D0100
43          (libraryRef OrCAD_LIB)))
44        ...)
45       (instance Q102
46        (viewRef NetlistView
47         (cellRef PCB_ICT_D0100
48          (libraryRef OrCAD_LIB)))
49        ...)
50       (net N101
51        (joined
52         (portRef &1 (instanceRef R101))
53         (portRef &1 (instanceRef Q101))))
54       (net N102
55        (joined
56         (portRef &2 (instanceRef R101))
57         (portRef &1 (instanceRef Q102))))))))
58   (design ED2
59    (cellRef ED2
60     (libraryRef MAIN_LIB))))
```

An example of an EDIF netlist generated by OrCAD Capture.

Chapter 7

Netlist Handling

The netlist is not a format suitable for computing possible ICT tests. The netlist is just a list of components and a list of nets connecting the terminals of these nets together. Therefore the netlist needs to be converted to a format that does allow makes computing the designs easier. A representation that shows great resemblance and that is studied well is the graph.

Where the netlist is a tuple of components and nets, the graph is a tuple of of vertices and edges. In the netlist the terminals of components are connected together by the nets, and in a graph the vertices are connected together by edges. Therefore the representation that seems obvious would be to represent the nets as edges and terminals or components as vertices.

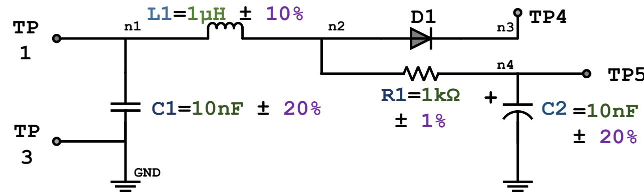
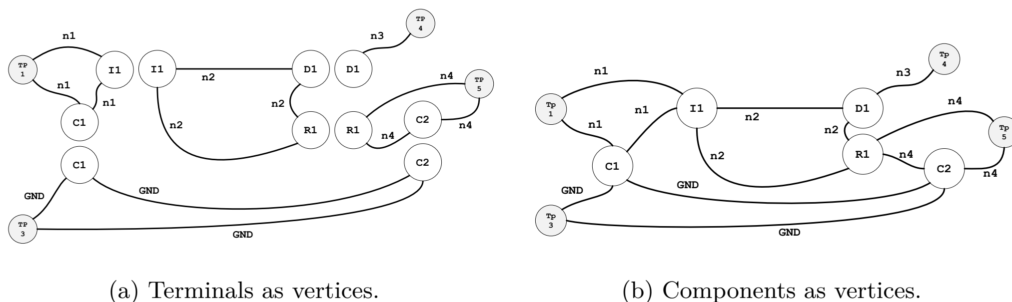


Figure 7.1: Example circuit. PCB.

Figure 7.2 show what this would look like for the example circuit of Figure 7.1. As apparent form these figures there are problems with these representations. First of all, both the graphs show a lot of cycles that are not actually present in the circuit. For example, both graphs contain the cycle $L1 \rightarrow D1 \rightarrow R1 \rightarrow L1$ that is not present in the netlist. Only when the labels of the edges connecting the vertices together are considered it becomes clear that net $n1$ is visited more than once.



(a) Terminals as vertices.

(b) Components as vertices.

Figure 7.2: Example circuit represented as graphs with nets as edges.

A closer look at the properties of vertices and edges reveals that nets are better suited to be represented by vertices, while most components can easily be represented by edges. Namely, in a netlist multiple terminals can be connected by a single net while an edge only a single point-to-point relation. Passive component (e.g. resistors, capacitors, inductors, fuses) and some active components (e.g. diodes, LEDs) only have two leads, while vertices can have an arbitrary degree.

However, since test pads are only one terminal devices these cannot be shown as edges. Therefore is is chosen to represent the vertices by colored vertices. A vertex that has probe access is represented as a green vertex, while a vertex without probe access is represented by a red vertex. The information about the mounting technology and the layer of the component is combined with the availability of a test pad to determine if a vertex has probe access.

The remaining information about the design that is extracted from the system inputs are stored in edge labels. The refdes, value, and tolerance are stored as a set in the edge label. If a certain component uses an array-package, then the internal components are extracted from this package and added to the graph as self-contained edges. The refdes that is saved in all of these edges is of course the same and therefore it can still be determined that these edges belong to the same package.

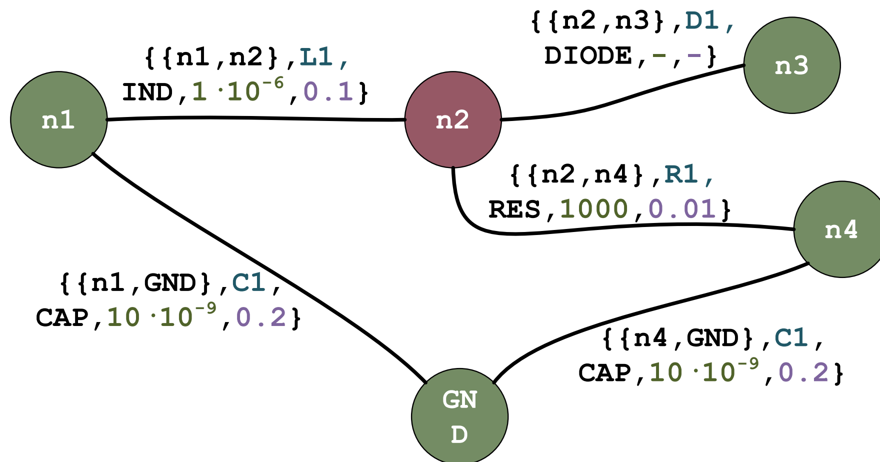


Figure 7.3: Example circuit. PCB.

Figure 7.3 shows what this would look like for the example circuit of Figure 7.1. It is clear that this representation does not have the problems of apparent cycles that don't exist. However, this representation has some other problems. Namely normal graphs do not allow for multiple edges to be in parallel, while in circuits parallel components are found in in most circuit designs. For this reason multi-graphs are used.

Another problem exists with components that cannot be represented as a single edge, such as ICs. Components with more than two leads do not resemble an edge. To solve this coloring is used again. All components that have more than two leads are represented by a black vertex, and all terminals of these components are represented by a colored edge. The edge color used for these terminals is orange.

Chapter 8

Test Generation for PCL-O of Passive Components

In-circuit tests excel at testing the presence, correctness and liveliness of passive components. If a passive component with only two leads is electrically tested for presence, it must be the case that the both connections of this device are not open, and are therefore covered as well. This chapter describes how tests can be mapped on the graph to cover the PCL-O properties for passive components.

8.1 Graph Decomposition

The graph that is created to represent the product (as described in Chapter 7) can potentially be enormous and contains information that is superfluous for passive PCL-O tests. Extracting only the information needed for these tests makes mapping tests faster and easier. Extracting this information from the graph is done in a process called decomposing. The information that needs to be extracted is:

- Information of the structure-under-test itself.
  - The device types of the components inside. (Resistor, capacitor, inductor, fuse, etc.)
  - The device class of the components inside. (Passive, active, electro-mechanical)
  - The value of the components inside. ($1k\Omega$, $1\mu F$, $1mH$)
- The surrounding circuit of interest.
  - The false paths. (See Section 3.2.4)
  - Which components are in the false paths.
  - The impedances of the components in the false paths.
- The highest voltage that can safely be applied to a net.
  - Which components need to be protected.
  - Which of these components can be reached from a given net.

As described in the previous chapter, the information about the SUT itself is stored in the edges of the graph representing that structure. Hence selecting an edge from the graph will decipher the needed information of the structure.

The information of the components surrounding the structure-under-test and the voltage that can be safely applied to the nets is harder to extract from the graph. Sections 8.1.1 to 8.1.3 describe algorithms used in the process of decomposing. In Section 8.1.4 these algorithms are combined to shown an overview of the complete decomposing process.

8.1.1 Connected Components and Biconnected Components

If two parts of a netlist are not connected by a net, nor by components, it is impossible that current will flow from one of these parts to the other. In the graph representing a netlist these groups would result in distinct subgraphs for which it holds that there does not exists a path between the subgraphs. This is defined as the connected components of a graph. The algorithm for finding the connected components [15] of an undirected graph is well known and runs in linear time.

However, if there does exist an electrical path from one net through certain components to another net, it does imply that there is also a return path to create a circuit. For example, in the circuit in figure 8.1 current could potentially flow from N_1 to N_5 if one of these nets is sourced while the other is measured. However, for an in-circuit test this scenario is very unlikely since the measurement is over six or seven components at once. In this example it would only be possible to guard $C1$ by applying the guard on net N_6 (see Section 3.2.4).

This means that for each structure in the graph the surrounding circuit of interest is probably smaller than the connected component it is in. In fact, only the components that are part of some false path with the same start and endpoint as the structure under test are of interest since these paths will influence the measurements if no precautions are taken. These paths are called the false paths.

A single path can be found in $\mathcal{O}(V + E)$ time, but the number of simple paths in a graph can be very large, e.g. $\mathcal{O}(n!)$ in the complete graph of order n [16]. This means that finding all the false paths around each structure is a complex problem and that naively using a k-th shortest simple path [17] algorithm such as the Hoffman algorithm [18] to find the false paths of each structure is not a viable solution on large graphs such as the full netlist graph.

However, a closer look at these false paths reveals some interesting properties. Namely that edges being in a false path of the SUT is both a reflexive and symmetric relation. Proofs of this can be found in Appendix A. This means that it should be possible to find the false paths of an edge once, and reuse the data for all the edges that are found in the process. Furthermore, note that false paths can only be simple paths; a net, which represents a piece of conducting material, cannot have multiple voltages at the same time and current cannot pass it more than once.

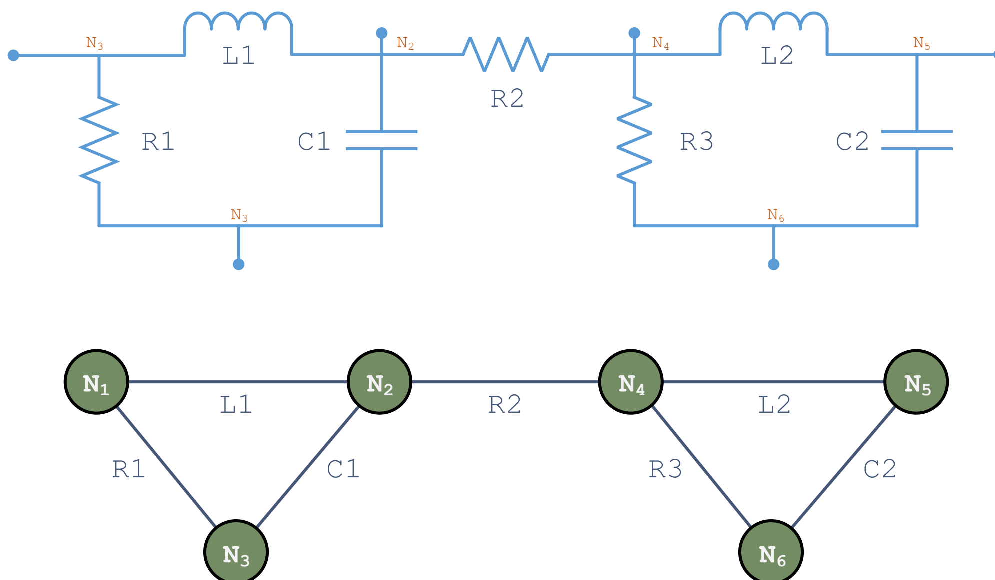


Figure 8.1: False paths in a connected component.

Biconnected components and articulation points [19] have very similar properties as these nets. It can even be proven that the set of edges consisting of the SUT and edges in its false path is equal to the biconnected component that holds the edge, see Theorem 1. So, in order to find all the false path of a structure, one can use a biconnected component algorithm such as the one proposed by John Hopcroft and Robert Tarjan [15], which runs in linear time.

Definition 1. *A path p is a false path of $e = \{s, m\}$ if and only if p is a s, m-path, denoted as $s \overset{p}{\rightsquigarrow} m$ and p is a simple path i.e. $\forall_v[v \in p : \#_v(p) = 1]]$.*

Definition 2. *The articulation points $\mathbb{A}(G)$ of a graph G are defined as vertices that increase the number of connected components of G if it would be removed from the graph. For any vertex $v \in V(G)$ it holds that $v \in \mathbb{A}(G) \Leftrightarrow \exists_{u,w}[u, w \in V(G) \wedge u \neq w : \forall_p[u \overset{p}{\rightsquigarrow} w : v \in p]]$*

Definition 3. *The biconnected components $\mathbb{B}(G)$ of a graph G are defined as the fragments held together by the articulation points $\mathbb{A}(G)$.*

Lemma 1. *For some graph G, let $v_1, v_2 \in V(G)$, where $v_1 \neq v_2$, then:*

$$\exists_p \left[v_1 \overset{p}{\rightsquigarrow} v_2\right] \Rightarrow \exists_q \left[v_1 \overset{q}{\rightsquigarrow} v_2 : \forall_{v \in c} \left[\#_v(q) = 1\right]\right]$$

Proof. Let v_1 and v_2 be two distinct vertices of graph G. Let $p = v_1 \rightsquigarrow v_2$ be the shortest path between v_1 and v_2, then p has to be a simple path. Namely, assume it is not simple. Then some vertex has to be in p at least twice: $p = v_1 \overset{p_1}{\rightsquigarrow} x \overset{p_2 \neq \emptyset}{\rightsquigarrow} x \overset{p_3}{\rightsquigarrow} v_2$. However, that means there exists a path $v_1 \overset{p_1}{\rightsquigarrow} x \overset{p_3}{\rightsquigarrow} v_2$ that is shorter than p, hence there is a contradiction.

By contradiction it can be concluded that the shortest v_1, v_2-path is a simple path, thus it can be concluded that if there exists a v_1, v_2-path, there also exists a simple path between v_1 and v_2. □

Theorem 1. *For some graph G, let $e \in E(G)$, where $\mathbb{P}_e$ is the sets of all false paths of e, then:*

$$\left\{e_1 \middle| e \in \mathcal{B} \in \mathbb{B}(G) \wedge e_1 \in \mathcal{B}\right\} = e \cup \mathbb{P}_e$$

Proof. Let $e = \{s, m\} \in E(G)$ be some SUT. Now, if a random traversal following path $p = (p_1, p_2, ..., p_n) \not\ni e$ is started from any of the vertices in e, lets say s, then anywhere in the path it either holds that an articulation point has been crossed (i.e. $\exists_v[v \in p : \forall_{q_1}[s \overset{q_1}{\rightsquigarrow} p_n : v \in q_1] \vee \forall_{q_2}[p_n \overset{q_2}{\rightsquigarrow} m : v \in q_m])$ meaning that it is not possible to return to both s and m without crossing the articulation point again.

- An articulation point has been crossed:
  - By the definition of biconnected components this means that the path has reached some other biconnected component.
  - Assume that p can still be part of a false path of e. A false path of e concatenated with e itself creates a simple cycle, meaning that there should are at least two distinct s, p_n-paths. This however contradicts with the fact that the articulation point v is on every s, p_n-path and therefore it can be concluded that p cannot be part of some false path of e.
- No articulation point has been crossed:
  - By the definition of biconnected components this means that the complete path lies within a single biconnected component.

– There exists a s, p_n-path, namely p. There also has to exist a p_{n-1}, m-path q since no articulations are crossed. By applying Lemma 1 it therefore holds that there also exists a simple s, p_n path p' and a simple p_{n-1}, m-path q'. It follows that there has to exist a simple path $\hat{p} = s \overset{p'}{\rightsquigarrow} p_n \overset{e_l}{\longrightarrow} p_{n-1} \overset{q'}{\rightsquigarrow} m$ containing e_l. However for $\hat{p}$ to be a false path of e, p' has to be distinct from q' otherwise $\hat{p}$ isn't simple. So, assume that $\hat{p}$ cannot be simple, then there has to exist a vertex $v \in p'$ that is on every possible p_n, m-path. This would be an articulation point, which contradicts with the fact that p reached p_n without crossing one. By contradiction it holds that there has to exist a simple path $\hat{p}$, from which it follows that $\hat{p}$ is a false path of e.

Hence it can be concluded that if a path p crosses an articulation point the path has reached a different biconnected component and it can no longer be a false path of e. As long as the path does not cross an articulation point it can be concluded that the path lies in a single biconnected component and that the edges taken in that path have to be part of some false path of e. Therefore it can be concluded that the set of edges inside of the same biconnected components as e is equal to e together with the set of edges that are part of a false path of e. □

8.1.2 Graph collapsing

Since the netlist is represented as a multigraph there can be multiple edges in parallel, just like there can be multiple components in parallel on a PCB. However, as mentioned in section 3.2.4 parallel components cannot be isolated from each other during ICT testing. Furthermore in Chapter 5 it is mentioned that the goal is to test components without test access as compound structures.

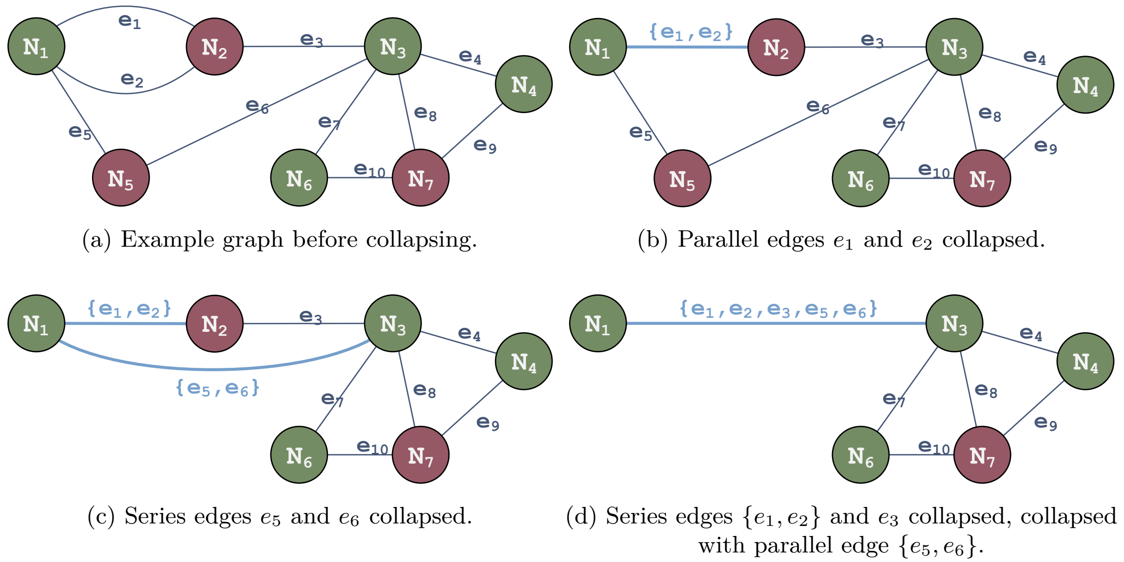
In this section the process of converting these parallel components and components without probe access into single structures is described. This (reversible) process is called collapsing. The reverse process is called expanding. The goal of the collapsing process is to reduce the graph from a multigraph containing unreachable components to a simple graph containing as much reachable structures as possible. The circuit of Figure 8.2(a) is used as an example.

The first step in reducing the graph is to convert it from a multigraph G to a simple graph G'. In order to do this all vertices from G can be added to the new graph G'. Then each edge $e = \{v_1, v_2\} \in E(G)$ will be added from G to G', using a procedure that collapses edges if there already existed an edge between vertices v_1 and v_2. Figure 8.2(b) shows an example of G' after the parallel edges are removed.

Now that the graph does no longer contain any parallel edges the collapsing of components can start. In this process edges in series with at least one unreachable net are collapsed into a single edge. Note that only edges connected to vertices with degree 2 can be collapsed. If this is the case then the two edges connected to these vertices are collapsed, and added to the graph. In Figure 8.2(c) the two edges e_5 and e_6 connected to non-probable vertex N_5 are collapsed.

However, graph G' should stay a simple graph, meaning that while collapsing edges it should be checked if there already exists an edge between the vertices of the new collapsed edge. Figure 8.2(d) shows an example of this situation; collapsed edge $\{e_1, e_2\}$ and edge e_3 are in series connected by non-probable net N_2 and should be collapsed to a new edge $\{e_1, e_2, e_3\}$. But if this net is collapsed there is already edge $\{e_5, e_6\}$ between N_1 and N_3, so these two edges have to be collapsed again to $\{e_1, e_2, e_3, e_5, e_6\}$ before adding it to G'.

In figure 8.2(d) it also becomes clear why only vertices with degree 2 can be collapsed. Vertex N_7 is does not have probe access, but since it is connected to N_3, N_4 and N_6 it cannot be collapsed between two vertices.



(a) Example graph before collapsing.

(b) Parallel edges e_1 and e_2 collapsed.

(c) Series edges e_5 and e_6 collapsed.

(d) Series edges $\{e_1, e_2\}$ and e_3 collapsed, collapsed with parallel edge $\{e_5, e_6\}$.

Figure 8.2: Example of collapsing process.

8.1.3 Minimal-Guards-Set Finding

If not every false path is guarded, these paths will probably influence the measurement. However, if too much nets are guarded in a false path components are shorted between two guarded nets. This shorting will not harm the components since there will be no current flowing through the components, but it will withhold the component of contributing to the guard impedance. As mentioned in Section 3.2.4 some current will flow through the guarded components due to imperfections in the source and measurement circuits. Furthermore, if the current that drained through the guard will be bigger than the source circuits can deliver, testing becomes impossible. Using as few guards as possible is therefore desired.

To solve this problem an algorithm is designed that chooses just the right amount of guards, called the minimal-guards-set algorithm. Given a SUT and the biconnected component it is contained in, it finds all the options of minimal sets that do guard every false path. It works by first determining the false paths, removing the vertices without probe access, and sorting the paths on length. Then it chooses a possible guard for the first path, recursively performs this on underlying paths, and chooses the next available guard for the path and recurses again. It repeats this process exhaustively until all possible options are found.

The algorithm will be demonstrated using the biconnected component shown in Figure 8.3. Note that the graph in this figure is not a valid biconnected component since vertex N_4 does not have probe access and has degree 2, meaning that this vertex would be collapsed. However, this does not matter to show how the algorithm behaves and allows for a concise example.

For this proof-of-concept it is chosen to use a undirected variant of the Hoffman Pavley algorithm [18] to determine the paths parallel to the SUT. The Hoffman Pavely algorithm is intended for directed graphs, but since the biconnected components are undirected graphs the algorithm is adapted to handle every edge as being bidirectional. As mentioned in Section 8.1.2 this can be a slow algorithm, but the difference is that this time it runs on a much smaller graph for which it holds that every edge in the graph is part of a false path of the SUT.

In order to make sure that components do not get shorted by guards, the paths with the least guarding options should be considered first. For this reason the vertices with probe access are

removed from the paths and sorted based on length.

Even though the paths are sorted on length the algorithm still needs to take into account that vertices of longer paths might also be part of a shorter path. This can for example happen when a path with few and a path with many guard options eventually merge into the same path. The algorithm has to make sure that none of the selected vertices appear in already processed paths.

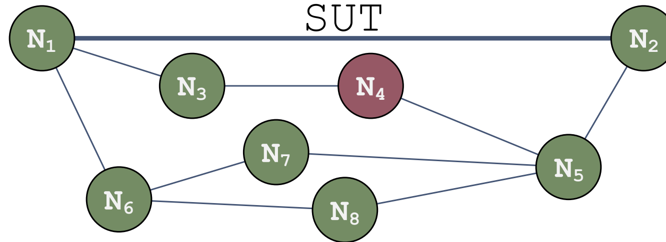


Figure 8.3: Example biconnected component for showing the minimal-guard-set algorithm.

In the example circuit shown in Figure 8.3 the SUT is the edge between N_1 and N_2. The steps taken by this algorithm are shown in Figure 8.4 on Page 37.

- The paths parallel to the SUT are searched using the Hoffman Pavley algorithm, of which the results are shown in cell A1 of Figure 8.4. As shown, three paths are found, named 1,2, and 3. These paths still contain vertex N_4, which does not have probe access, so in the next step shown in cell A2 this vertex is removed and the paths are sorted on length again.

- The exhaustive search for guard-sets can be started. In cell B1 is shown that N_3 is chosen as the vertex to guard the path with the least guarding options; path 2. Since the other guarding option N_5 would cause a part of path 2 to be shorted, this vertex is marked as *unavailable*. The algorithm jumps to the underlying path 1 which has two guard options left, N_6 and N_8, of which the first is chosen as guard and the other is marked as unavailable (shown in cell B2). Now path 3 contains the already chosen vertex N_6, so this path is skipped. The first result $\{N_3, N_6\}$ is found.

- Since path 3 was done, the process jumps back to the path above it and chooses the next available guard option which is N_8 and marks the other vertex as unavailable. The algorithm goes to the underlying path again to find that the only available choice would be N_7, getting another result $\{N_3, N_8, N_7\}$.

- Again there is no other choice for path 3 so it goes up to path 1 which also has no choices left so it goes back up to path 2. Path 2 has the option of N_5 left as a guard and marks N_3 as unavailable (cell D1). When it jumps to the underlaying path 1 it finds that it is already guarded (cell D2) so continues with the underlying path 3. This path is already guarded as well (cell D3), meaning that the result is only $\{N_5\}$.

- The algorithm goes all the way back up to find that there are no more options to consider and is done.

The algorithm found $\{N_3, N_6\}$, $\{N_3, N_8, N_7\}$, and $\{N_5\}$ as results for the example circuit in Figure 8.3. It can be verified that by choosing one of these three results there does not exist a false path of the SUT that does not cross a vertex in that result. It can also be verified that none of the components in the false paths are shorted between guards.

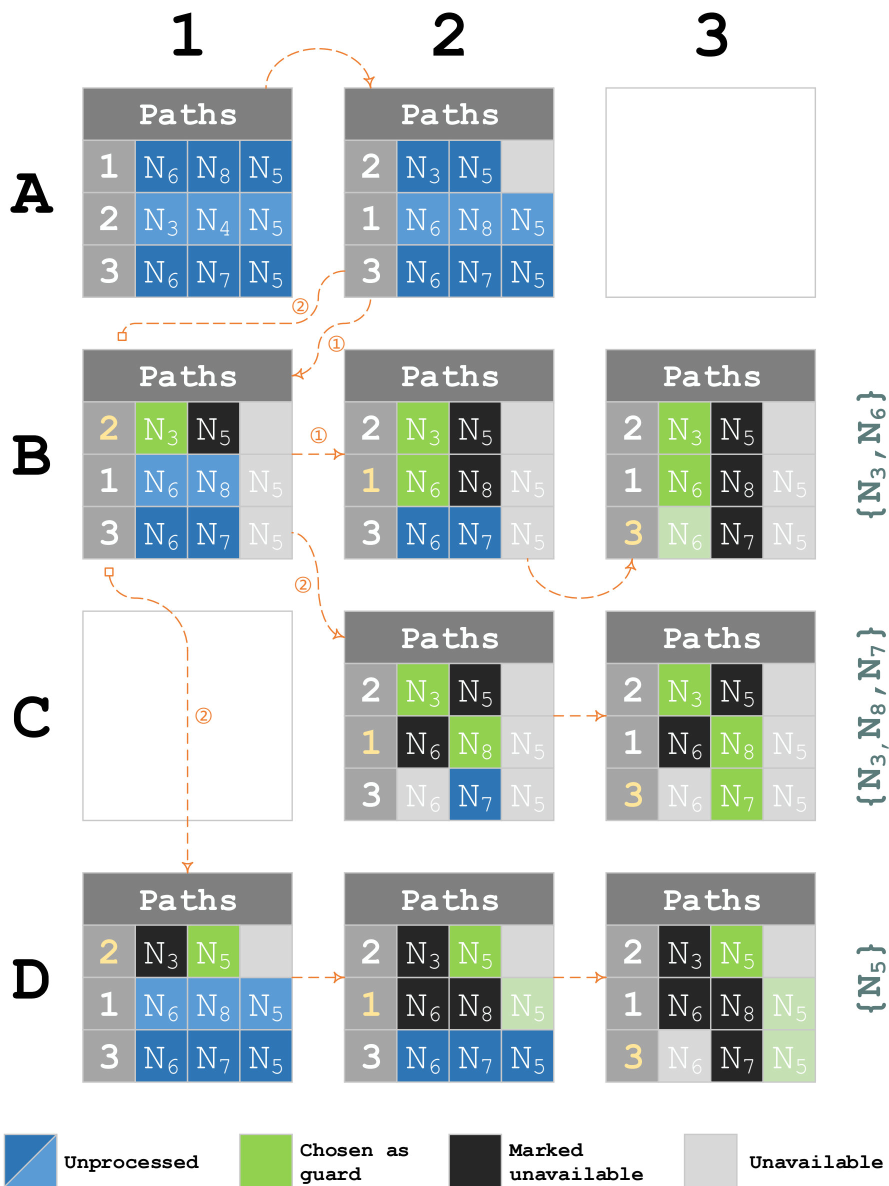


Figure 8.4: Example of collapsing process.

8.1.4 Passive Decomposition

The passive decomposition algorithm pre-processes the graphs before test mapping can take place. The goal is breaking the full graph into small pieces while conserving the required information as described in the introduction of this section.

The tests that will be mapped are all passive tests. The leads of active components typically are connected to PN-junctions inside the package, and when this is not the case this should have been indicated in the component description, as described in Section 6.2.2. If the stimulus voltage is made low enough in amplitude to avoid accidentally turning on these integrated PN-junctions, the influence will be minimal [20]. This means that it if the test voltage is low enough, the active components can be assumed to be absent and thus removed from the graph. Sometimes however the test equipment is not precise enough to measure the current that flows through high impedance components, so a higher voltage is desirable. For that reason the nets that are connected to a active component get marked so.

All remaining edges in the graph are passive. For these edges a dictionary of impedances is created. This dictionary maps edges to an impedance. Since the impedances of capacitors and inductors are frequency dependent, the impedance is set up as a function of the frequency e.g. an inductor $L1$ with value $10\mu H$ gets impedance $Z_{L1}(\omega) = \omega \cdot 10 \cdot 10^{-6}$.

The collapsing process is applied twice to the original graph, once before finding the biconnected components, and once after. This is needed because the biconnected components algorithm would find articulation points within strings of series connected edges that have no probe access and marking these as different biconnected components, effectively losing test opportunities. Collapsing should happen after finding the biconnected components again since if might be the case that the an articulation point is a vertex with a degree bigger than 2, while after abstracting the biconnected components as a block the degree is only 2. Both of these cases are treated in the example below.

Every time two edges are collapsed together an event is fired. This events are overheard by an observer that records the collapsing events. This way the observer can constantly supplement the impedance dictionary with new edges and their impedances.

In between the two collapsing processes the connected components and the biconnected components of the graph are found, see Section 8.1.1. This is done with the the linear time algorithm proposed by John Hopcroft and Robert Tarjan [15].

Once these biconnected components are found they are extracted from the graph and turned into self-contained graph called blocks. It is important to preserve the information at which points the blocks were connected to each other. The blocks will be the entities to which test will be mapped as if there does not exist any other circuitry. However, via an articulation point voltages can still be applied to nets outside of the block, meaning that active components can still be damaged if to high voltages are applied.

Finally for all of the edges (*structures*) in each block the minimal guard sets are found.

Example:
Figure 8.5 shows the full graph of a product that will be used as an example for passive decomposition. It contains both passive components (dark blue edges) and active components (orange edges), nets with probe access (green vertices) and nets without probe access (red vertices), and two integrated circuits U_1 and U_2 shown as gray vertices. The passive decomposition algorithm is applied to it, consisting of the following steps.

- Active components are removed from the graph while maintaining the information of which vertex is connected to an active component. This turns the full graph in a passive graph.

This is shown in Figure 8.6, where components connected to an active component are marked with a yellow border.

- The passive graph is collapsed for the first time to create structures that are inseparable by the next step of the algorithm. An example of why this step has to be executed before finding the biconnected components is shown in Figure 8.7. If edge e_7 and e_{20} would not be collapsed into a single structure, net N_4 would be an articulation point, resulting in 2 non-testable blocks since they would have only access to a single net.
- The connected components and biconnected components of the graph are found. In Figure 8.8 the in total three connected components c_1, c_2, and c_3 are found. For each of these components the biconnected components are found as well. This is shown in Figure 8.9. Connected component c_1 contains biconnected component $b_{1.1}$, c_2 contains [$b_{2.1}$, $b_{2.2}$, and $b_{2.3}$], and c_3 contains [$b_{3.1}$, $b_{3.2}$, and $b_{3.3}$].
- The biconnected components are saved as self contained blocks which are collapsed again as showed in Figure Figure 8.10, where the orange labels represent articulation points via which an active component can be reached. Block 3 and 7, which are the collapsed $b_{2.1}$ and $b_{2.3}$ respectively show why collapsing needs to be done a second time.
- The finding of the guard sets is not shown in the example. An elaborate example of how this is done can be found in Section 8.1.3.

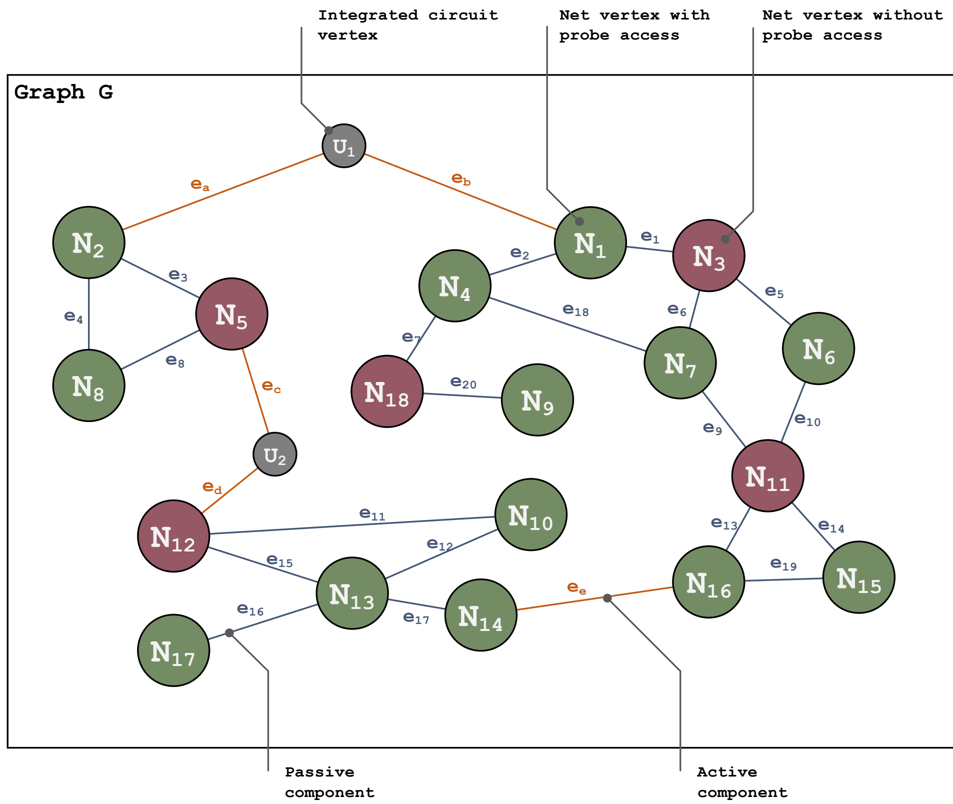


Figure 8.5: The example circuit.

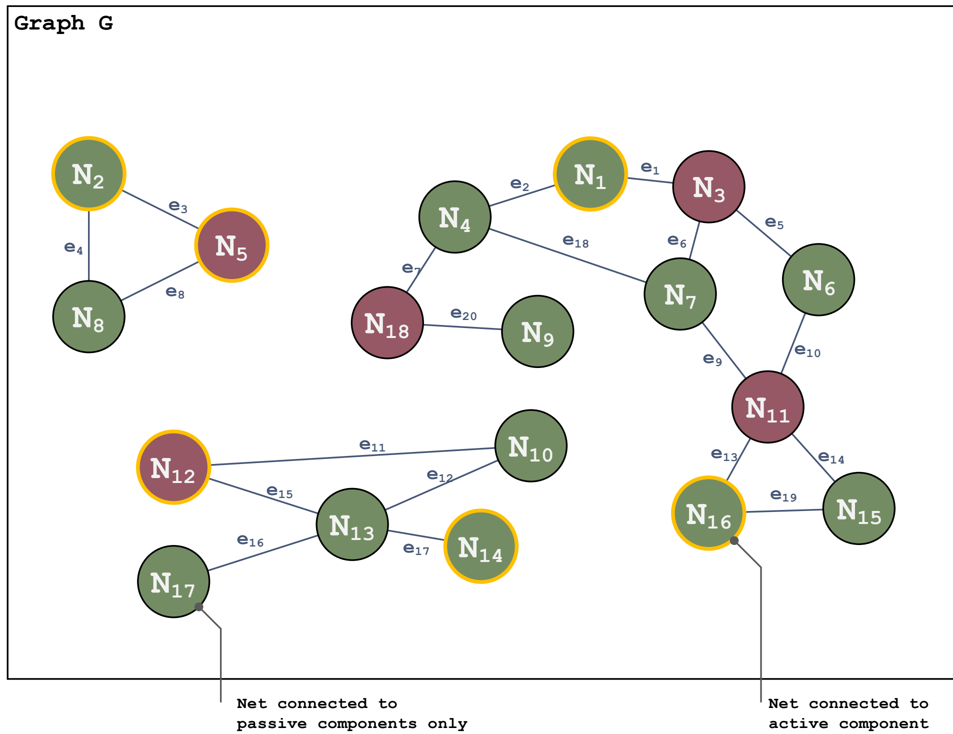


Figure 8.6: Passive graph.

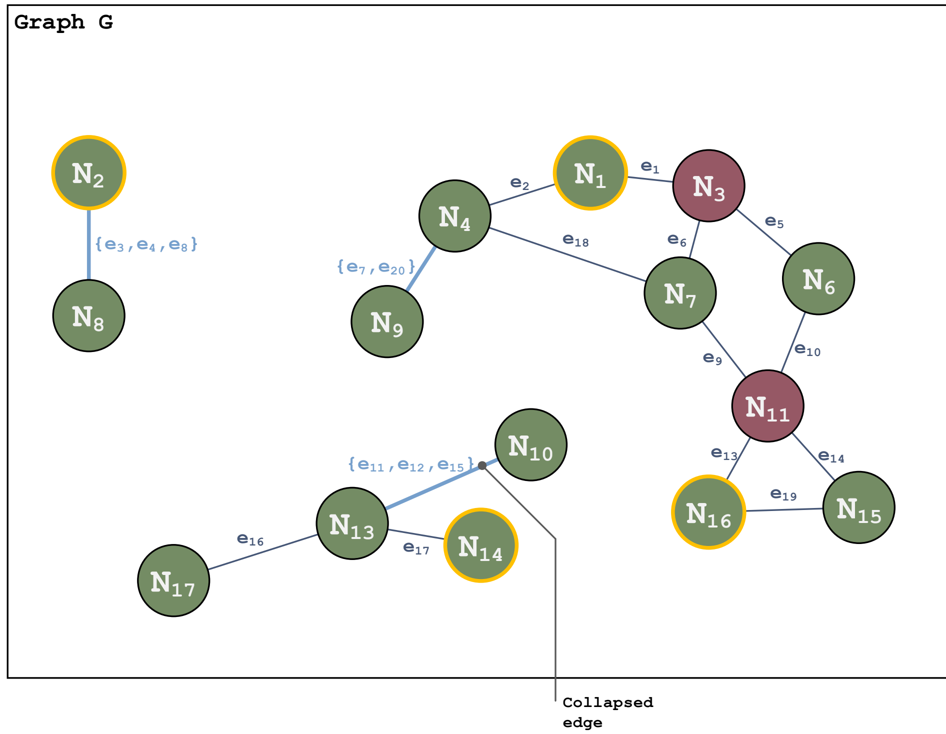


Figure 8.7: Collapsed passive graph.

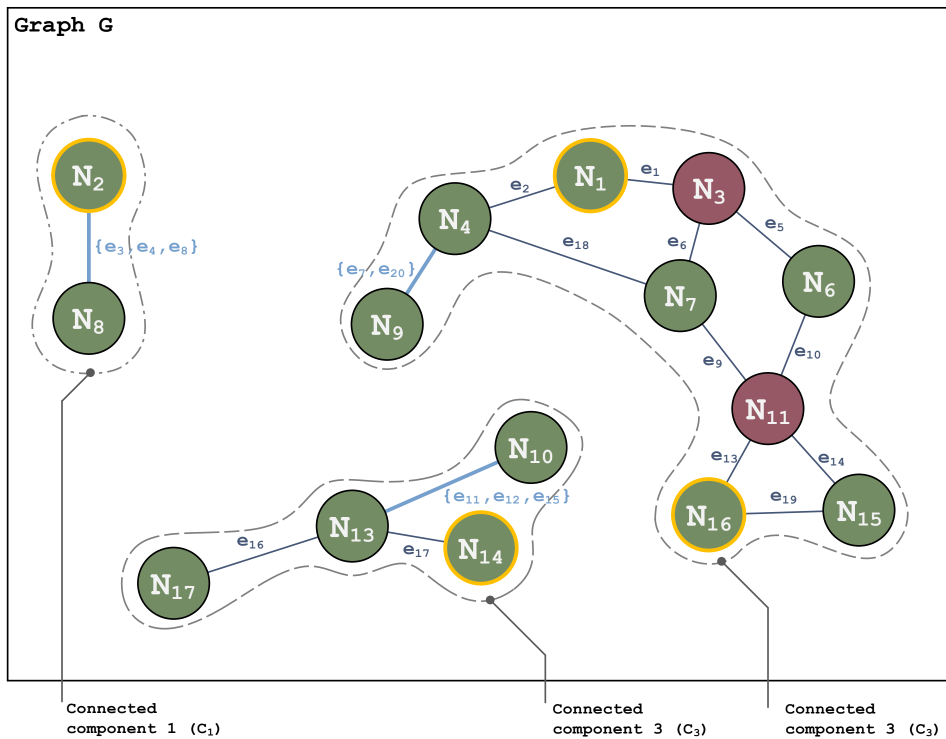


Figure 8.8: Connected components.

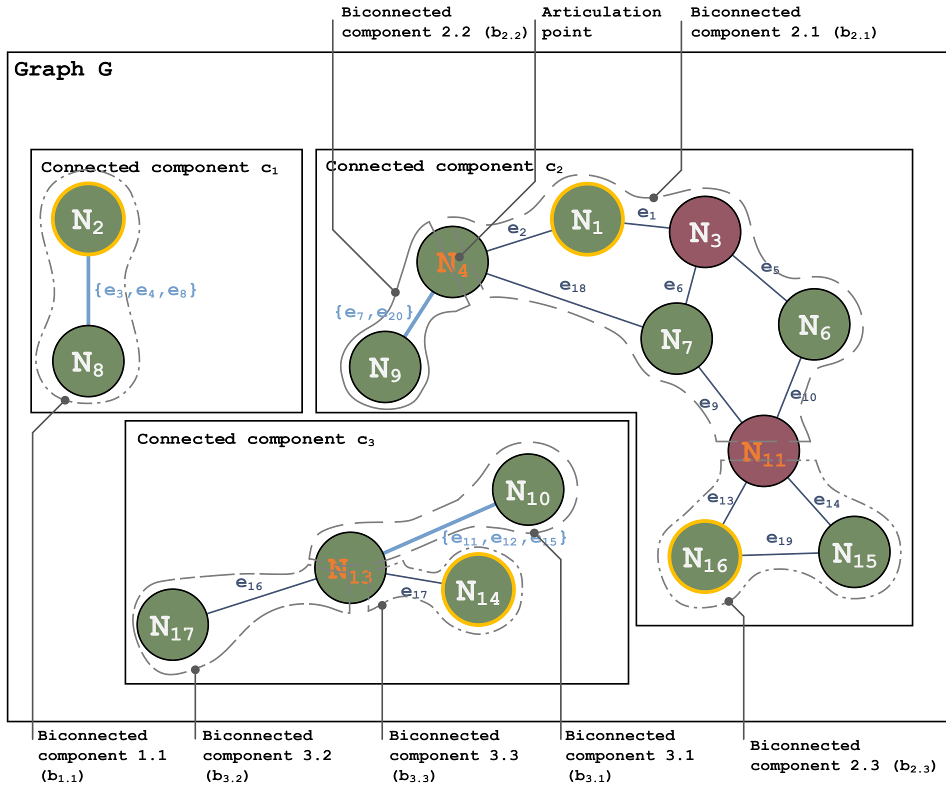


Figure 8.9: Biconnected components.

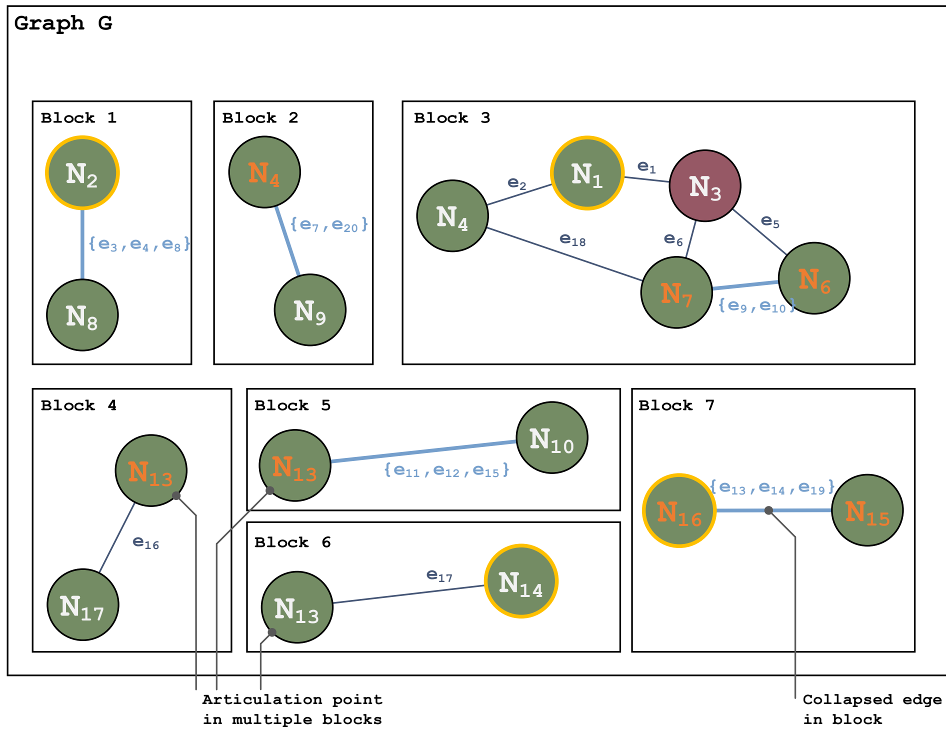


Figure 8.10: Collapsed blocks.

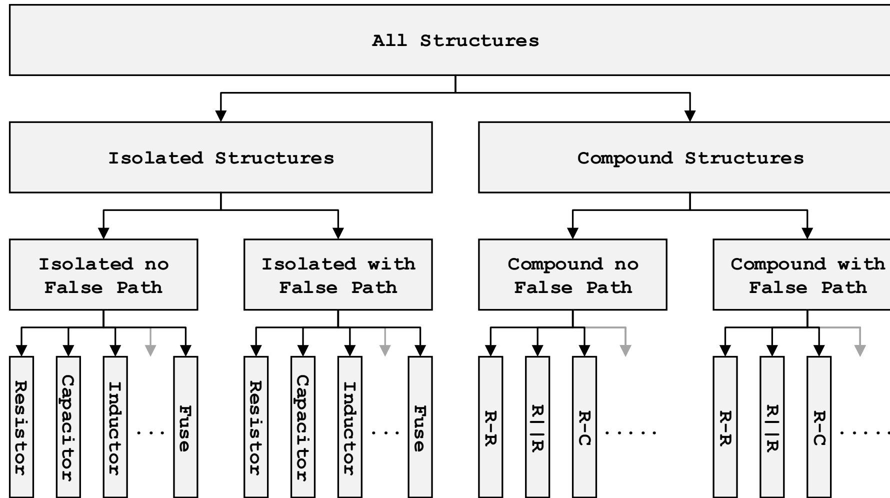


Figure 8.11: Collapsed blocks.

8.2 Test Mapping

After the graph is decomposed as described in Section 8.1 and the blocks resulting from this decomposition are ready, tests should be mapped to each of the structures in these blocks. Even though all blocks are potentially different, the blocks can be split up into four distinct groups. If a block contains only a single structure, than that structure cannot have a false path. However, if the block does have more than a single structure, then the structures in that block must be in each others false paths (see Section 8.1.1).

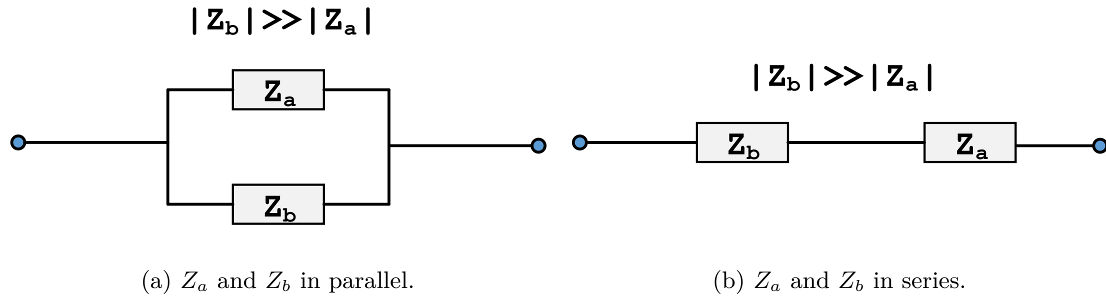
Another distinction that can be made is if the structure contains multiple collapsed edges or not. If the structure contains only a single edge, the structure is a isolated component. If the structure represents multiple edges, then that structure represents a composition of components.

This means the four distinct groups are: (1) Isolated no false path (INFP), (2) isolated with false path (IWFP), (3) compound no false path (CNFP), and (4) compound with false path (CWFP). The groups are shown in the second last layer of Figure 8.11.

A library of tests can be created that will hold a test for each of the structures at the bottom layer of Figure 8.11 just as how some existing ICT testers would do this [21]. For the isolated structures this should be no problem at all. Once a test is specified for each component type, all cases are covered. However, for the compound structures this is not the case, theoretically the group of distinct compound structures that could be found on a PCB can be infinitely big, so it is impossible to cover each case. Therefore, to maximize the test coverage a generic test method is needed that tries to map a test on a structure if there is no test found in the test library.

8.2.1 Generic Test Method

The generic test method is a method that tries to map an as effective as possible test on the structure with as few test steps as possible regardless of what is inside that structure. However, since there can be multiple components inside a structure that can in any kind of configuration on the PCB this is not an easy task.



(a) Z_a and Z_b in parallel.

(b) Z_a and Z_b in series.

Figure 8.12: Z_a and Z_b in parallel and series where $|Z_a| << |Z_b|$.

Test Effectiveness

The value and therefore the impedance of every component has a tolerance the current that flows through a structure may vary. This means that components might be *invisible* inside of a structure. For example, some components cannot be verified to be present since their contribution of the total current flowing is to small to distinguish.

Consider the two simples cases shown in Figure 8.12. In Figure 8.12(a) two impedances are connected in parallel, with $|Z_b|$ much bigger than $|Z_a|$. In Figure 8.12(b) the same impedances are in series. In both configurations neither Z_a nor Z_b can be isolated.

If the impedances are in parallel, the voltage over both impedances is the same, but most current will flow through the path with the least resistance. In the case of Figure 8.12(a) this would be through Z_a i.e. impedance Z_a is *dominant* (see Section 8.2.1). If a test is created and current flows from source to measurement, the measurement can only verify if impedance Z_a is present.

If the impedances are in series, current has to pass through both Z_a and Z_b. This means that if current starts flowing both components have to be present. However, since impedance Z_b is much larger than impedance Z_a, nearly all voltage will be over Z_b. This means that if a certain impedance in measured over the structure that only component Z_b can be verified to be live and having the correct value.

It does not always have to be the case that components are strictly in series or parallel. Using the three fundamental laws of network analysis; Ohm's law, Kirchhoff's voltage law (KVL), and Kirchhoff's current law (KCL) [22] the voltages over and currents through each components can still be determined. Using these methods it can therefore be determined if a component is dominated.

Frequency

In these examples of the previous section was assumed that Z_b was always bigger than Z_a. However, capacitors and inductors have a frequency dependent impedance, meaning that a component might be dominant on one frequency while almost invisible on another. These properties should be exploited to get a test that is effective for as much components in the structure as possible.

This will be done by iterating over the frequency domain of the test equipment, which is specified in the configuration files of the AET as mentioned in Section 6.3. The size of the increments is supplied by the user. This will generate a list of the visibility of each component within the structure for a given frequency.

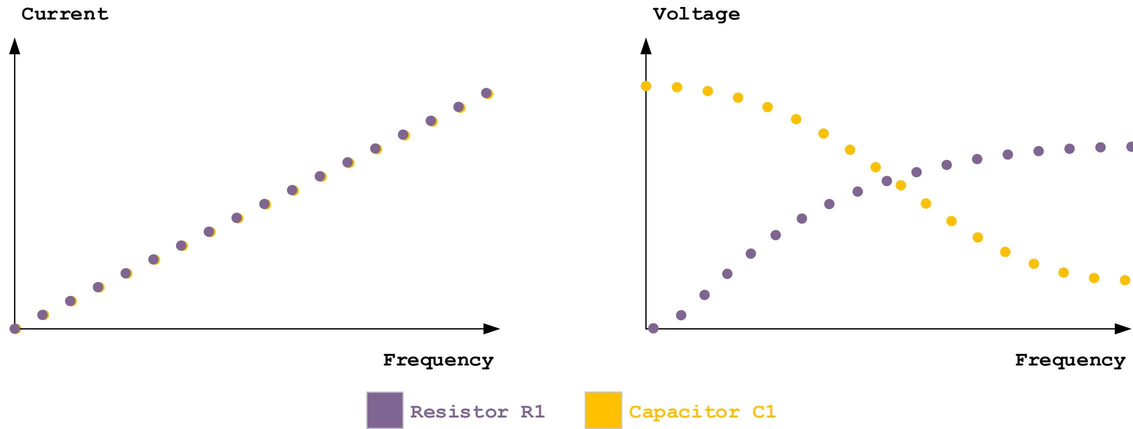


Figure 8.13: Example current and voltage data points of iterating over the frequency domain.

Figure 8.13 is an example of what this could look like. There are two components in series inside the structure; a resistor$R1$ and a capacitor $C2$. Since these components are in series the current through both components is always equal. The impedance of the $C1$ gets lower as the frequency rises, meaning that the voltage over $C1$ drops wile the voltage over$R1$ rises.

Dominance

A component is defined to be dominated if the current flowing through, or the voltage over the component is is smaller than the margin of error of the total structure. However, as mentioned in the previous section, the impedance of components are frequency dependent, meaning that the impedance of the structure itself is frequency dependent as well. This means that to determine if a component is dominated the tolerance of the structure has to be calculated for the chosen frequency. This means that not only the current through and the voltage over each component has to be calculated, but also its upper and lower limit for each of the components.

For example, the circuit of Figure 8.14 contains three components; $R1$, $C1$, and $L1$. To calculate the lower bound of the current through $R1$, it is clear that the resistance of $R1$ should be taken as high as possible $(+1\%)$. It is however not possible to say that the current through the other components should be high and therefore the value of $L1$ should be small (-10%) and the value of $C1$ should be high $(+20\%)$ to make their impedances low. Namely, since the impedance of $C1$ limits the current through $R1$ as well, the value of $C1$ should be as high as possible to get to the lower bound. And even when a way to maximize the current through $R1$ is found, this still does not mean that the current ratio of $R1$ to SUT current is optimized. Then for minimum and maximum voltage over a component a similar problem needs to be solved again.

Instead of this it is chosen to take the tolerance of the total structure to be equal to the tolerance of the maximum tolerance of the components inside that structure. For the example circuit above the tolerance of this structure would be defined as $\pm 20\%$. This definition is therefore not the true tolerance of the structure, but to the upper limit, i.e. if a component is not dominated using this definition, than it certainly is not dominated when calculated using the elaborate method. This makes this definition frequency independent, which means a great reduction the amount of calculations. Of course this also means that in some tests the test-effectiveness score is lower than strictly could be granted. However, the next section will show that the test effectiveness not necessarily has to suffer from this.

Test-Step Minimization

A test for a single structure should consist of as few test steps possible. But looking at the example data points of Figure 8.13 there might be numerous frequencies that result in a effective test. However, in Section 3.1 it is shown that the raw device score does not have to increase by testing a component more than once. Moreover, the test sequence would get to long, harming the lead time of assembly. On the other hand, test that contribute to a more effective test should not be skipped.

Therefore, the BTC can be used to determine if a test adds to the effectiveness of the overall test. For example, assume that four tests are found for a structure containing two components; $t_1(rds_1, rds_2) = [\emptyset,\ \emptyset]$, $t_2(rds_1, rds_2) = [\{po\},\ \emptyset]$, $t_3(rds_1, rds_2) = [\{po\},\ \{pclo\}]$, and $t_4(rds_1, rds_2) = [\{pclo\}), \emptyset]$. Now, since t_1 does not contribute at all it is obvious that this test should note be executed. Altough t_2 does have a score, it is a proper subset of t_3 and should therefore not be executed as well. Even though t_4 does not help with testing component 2, it should still be executed since its score for component 1 is better then the score of t_3 for the same component. This means that executing both t_3 and t_4 will given an optimal test effectiveness.

Now, this is an example of just 4 tests, but in practice this number will be higher. For example, if a test effectiveness score is determined for the AET that has a frequency domain of $(0,\ 150,000)$ with a increment step of 500Hz, there are 300 tests to consider. This also means that if some test has been granted a lower effectiveness score due to the definition of dominance in the previous section, there will likely exist a test on some other frequency at which the component is not dominated.

Modified nodal analysis (MNA)

To calculate the currents and voltages in the block an algorithm known as MNA [23] is used. Nodal analysis uses the Kirchoff laws to determine the voltages on each net and the current through the components.

To use MNA the graph has to be converted into a matrix specifying the connections and impedances of the components. Voltage sources of $0V$, which act as a short circuit, can be added to force MNA to calculate the current through it. Since the total current flowing through the system and the guard ratio are important, these sources are added between the source and the SUT and in both guard paths.

Another thing that can be added to the circuit are the parasitic impedances of the source, measurement and guard circuits. As mentioned in Section 3.2.4 these circuits are not ideal and will influence the accuracy of the measurement.

In Figure 8.16 the circuit is shown how this would be done for the generic test algorithm. The example circuit is provided with the parasitic impedances as described in Section 6.3. The voltage

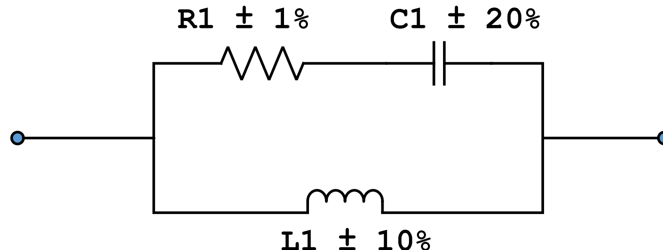


Figure 8.14: Example circuit for calculating dominance.

source is added to the circuit, shown in white. The $0V$-*ammeters* are added to the circuit as well, shown in gray.

The Generic Test Algorithm

The properties described in the previous sections will be brought together to create the generic test mapping algorithm. Given a SUT and the block it is contained in, the algorithm tries to find an as effective test as possible with the least test steps possible while considering the limitations of the test equipment.

First the algorithm takes the block and expands (the opposite process of collapsing as described in Section 8.1.2) the SUT to expose the internals of the structure. This is needed since the algorithm needs to know the current through each of the individual components of the SUT. The other structures in the block, the false paths, are not expanded.

Now one of the two vertices connected by the SUT is chosen as the source probe, the other as the measurement probe. From the results of the minimal-guard-set algorithm a guard set is picked. Figure 8.15(a) shown as example circuit that is expanded and for which the source, measurement, and guard probes are chosen as shown in 8.15(b). The vertex chosen as source is yellow, the measurement vertex is purple, and the only guard option is colored dark gray.

Then next step is to iterate over the frequency spectrum of the test equipment, in this case the AET, and calculating how much current will flow through the components. The currents and voltages are normalized as a percentage of the total current through and voltage over the SUT. This way, using the definition of dominance above, it can be determined if a component is visible for a given frequency. Using this information the effectiveness of a test on the given frequency is determined.

However, this does not mean that the test equipment is actually capable of executing the test. It might be that the currents that will flow are to small to measure, or to large to supply. By iterating over the voltage range of the test equipment it can be determined if there exists a voltage at which this test can actually be executed. The maximum voltage of the test equipment is however not the only limiting factor. As mentioned in Section 8.1.4 possible active components connected to the net should be taken into account as well. If an active component is connected to the net (direct or via an articulation point), the maximum active component voltage as specified in Section 6.3 is used as maximum.

If it can be executed it still needs to be determined if this test makes the overall test more effective. This is done by the process test-step minimization as described in Section 8.2.1. If the test does help making the test more effective it is added to the list of to be executed tests. If the test is in all areas more effective than a test already in the list, the existing test is swapped out for this one.

Figure 8.17 shows how this could look like for the example circuit of Figure 8.15(a). Since the components in the SUT are in series the current always has to pass both of the components, meaning that the current through both components is always 100% of the total SUT-current. For low frequencies the voltage over e_{11} is dominated by the voltage over e_{12}. However, there exist multiple frequencies for which none of the components are dominated. For one of the frequencies the search for an appropriate voltage is shown in the rightmost plot. If the voltage is to low then the SUT-current is to small to measure, but for voltages too high the source circuits cannot supply the current anymore.

The algorithm swaps the source and measurement probes and repeats the process. If there are multiple guarding options the algorithm picks the next option available, and repeats the process again.

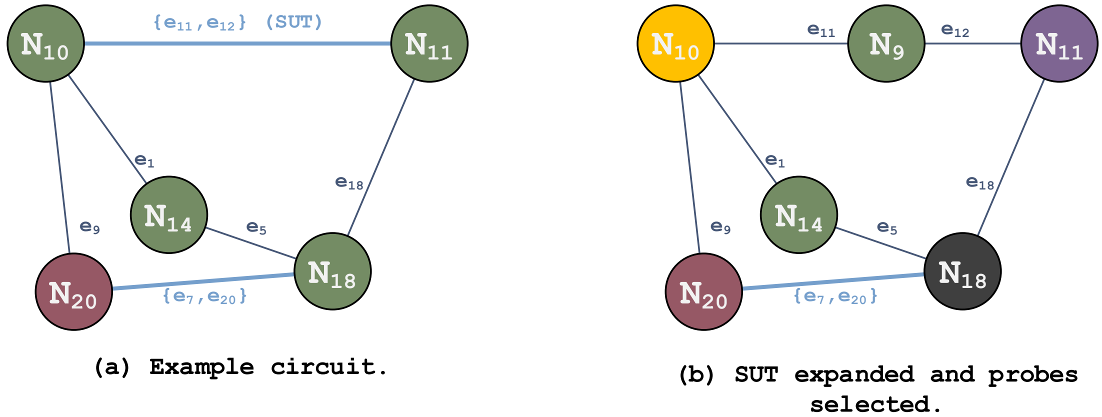


Figure 8.15: Example circuit for the generic test algorithm.

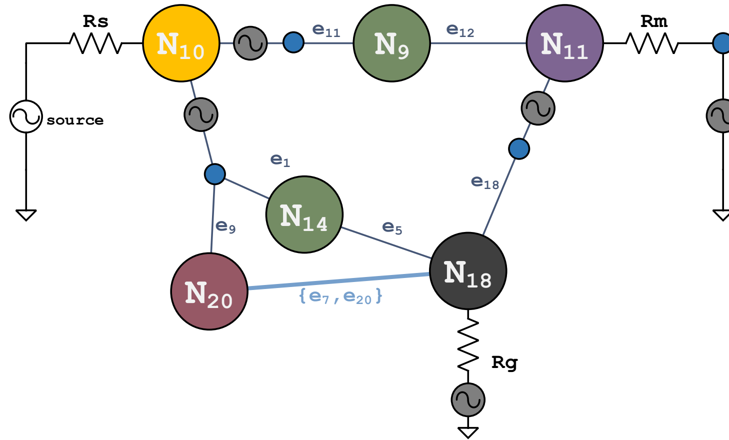


Figure 8.16: The example circuit adapted for use with MNA.

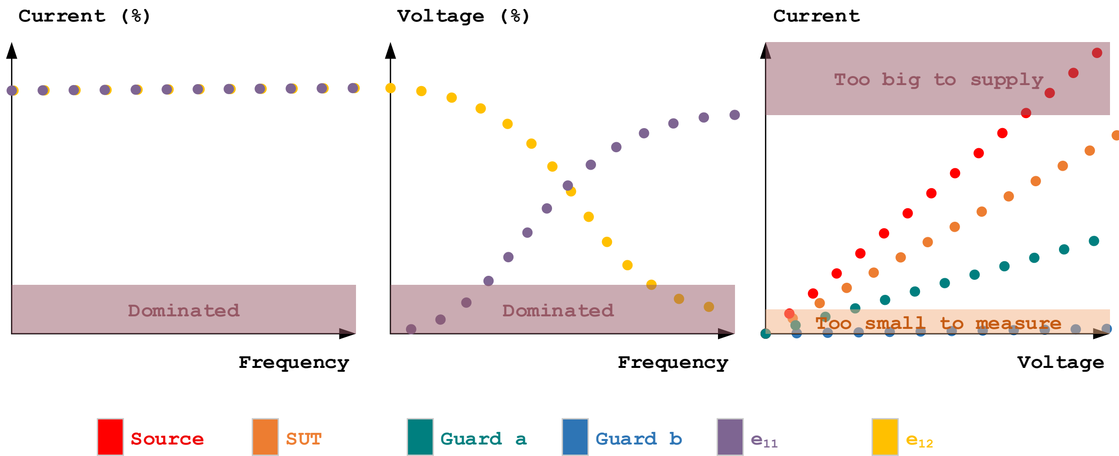


Figure 8.17: The normalized currents through and voltages over e_{11} and e_{12}.

8.2.2 Library Mapping

To give test engineers more control over how a test will be mapped than using the generic test method, tests can also be designed by test engineers and added to the tool by means of a plug-in. Such a plug-in consists of three major parts; (1) if the plug-in can handle false paths, (2) a so called *mold* which specifies to which properties a structure must comply, (3) a test mapping procedure, and (4) a test effectiveness report. Only if a structure complies with the mold the mapping algorithm will continue to try to map the test onto the structure.

Molds

A test designed by a test engineer typically focuses on a single component type. Some test methods are only suitable if the value of the component lies within some range. Therefore the mold for a single component exists of of a specification of a component type, and a value range. Molds for isolated components are called simple molds.

Figure 8.18 shows an example of a simple mold. The mold specifies that the component has to be of type capacitor (C) and that its value has to be at least $1\mu F$. The upper bound of the value is not defined. Structure 1 does not comply with this mold since its value is to low and is not a capacitor. Structure 2 does not comply either since it does have a value that is within the specified range, but the type of component is wrong. Finally structure 3 does comply since both its type is correct and its value within the specified range.

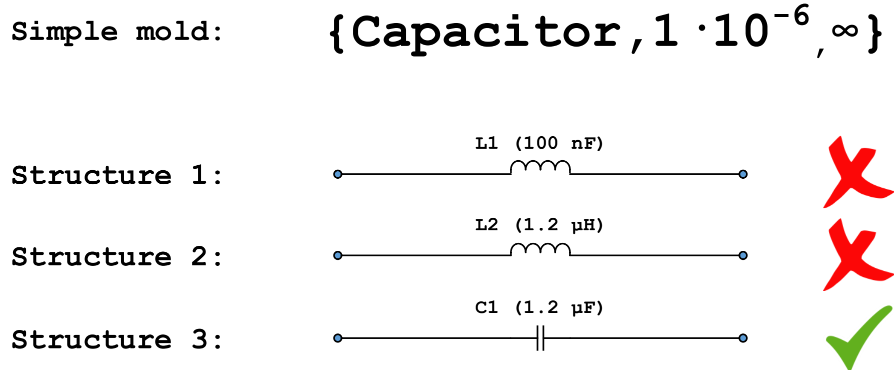


Figure 8.18: Example of a simple mold.

Mapping of isolated structures is relatively easy compared to mapping of compound structures. By definition there can only be one passive component in the structure, whereas structures contain multiple components all with their own values. Therefore compound molds have to be able to specify not only the components, but also their configuration. Compound molds are a composition of multiple simple molds that are small graphs just like the structures themselves.

The compound molds have labeled edges and two labeled vertices. The label of an edge $l(e)$ is equal to the simple mold of an isolated structure. The two labeled vertices define where the source and measurement probes will be placed. The definition of the edge label however slightly differ with the definition of a simple mold. Where the simple mold can consist of just one component, the labels of the edges in the compound mold are a representation of one ore more components in parallel.

Figure 8.19 shown an example of a compound mold. The mold consists of two edges in series; an edge representing capacitors between $100nF$ and $1\mu F$ connected to the source probe, and another edge representing resistors between 100Ω and 1000Ω connected to the measurement probe.

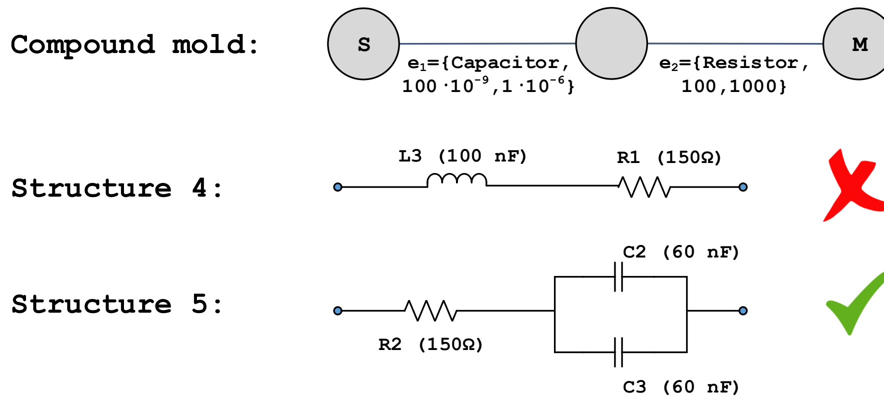


Figure 8.19: Example of a compound mold.

Structure 4 obviously does not comply with this mold since it has an inductor in it. Structure 5 has two capacitors that separately do not fall within the range. However, combined the two capacitors have a value of $120nF$ which is within the range, just like the single resistor $R2$. Note that the direction in which a structure is found does not matter, the structure still complies with the mold.

To find if a structure complies with a mold, it needs to be determined if the two graphs are isomorphic [24]. Graph isomorphism is a adjacency preserving bijection between the vertex sets of two graphs. The preserving of adjacency means that if two vertices are adjacent then it must hold that the the mapping of the vertices are adjacent as well. That the sets are bijective means that the relation is both injective and surjective.

Mold Isomorphism Algorithm The graph isomorphism problem is a NP problem. However, there exist two labeled edges in the mold which have to be mapped onto one of the two vertices connected by the SUT. Furthermore all edges are labeled. This greatly reduces the complexity of the problem. Let $t(e)$ be defined as the component type of the label of edge e. Let $r(e)$ be the range of an edge e if that edge is in a mold or the value of edge e if it is part of a SUT. The compound mold M and the graph of the SUT G are defined to be isomorphic if and only if there exists a bijection $f : V(G) \mapsto V(M)$ such that $e_g = \{u, w\} \in E(G) \Leftrightarrow e_m = \{f(u), f(w)\} \in E(M)$ and $t(e_g) = t(e_m)$ and $r(e_g) \cup r(e_m) \neq \emptyset$. Hence a normal isomorphism added with the restriction that the component types of edges have to be the same and that the values of the edges of the SUT has to be in the range of the edge of the mold.

The algorithm to check if the two graphs are isomorphic works as follows.

- First a simple check is done. If there are less edges in the SUT then there are in the mold the graphs cannot be isomorphic. In the collapsed edge components of the same type can be in parallel, but by definition this is is not possible for the compound mold. If the SUT contains an equal amount or more edges, the algorithm continues.
- A new undirected multigraph is created based on the SUT. In this graph there can only be parallel edges if the edges are of the same component type. This is to match the form of the mold. Parallel edges of the same type are merged while preserving the value of the two edges separately, e.g. for two capacitors C_1 and C_1 the value is $C_1 + C_2$ while for two resistors R_1 and R_2 the value would be $\frac{R_1 \cdot R_2}{R_1 + R_2}$.
- Then again a check is done. Since the relation is bijective the vertex sets $V(G)$ and $V(H)$ have to be of equal size. If the size of the vertex sets is equal the algorithm continues.

- If there exists a vertex $v \in V(G)$ and an edge $e_1 = \{v, \alpha\} \in E(G)$ then by definition of the isomorphism $\exists_{e_2}[e_2 = \{f(v), \beta\} \in E(M) : t(e_g) = t(e_m)]$. From adjacency preserving and bijection it follows that $\delta(v) = \delta(f(v))$. Therefore it has to hold that the multisets $\left\{c_1 \middle| c_1 = t\big(\{u, \alpha\}\big)\right\}$ and $\left\{c_2 \middle| c_2 = t\big(\{f(u), f(\beta)\}\big)\right\}$ are equal. The algorithm uses this property and groups vertices of G and M based on these component-type-multisets (ctm).
- Vertices of G can only be mapped to vertices of M that have the same ctm. This means that the same ctm have to exist for both G and M and that the number of vertices in these ctm are equal. If this is the case the algorithm continues.
- Let $e_{sut} = \{p, q\}$ be the SUT-edge, of v, and let v_s and v_m be the vertices in the mold labeled as source and measurement. Now either (1) $\text{ctm}(p) = \text{ctm}(v_s) \wedge \text{ctm}(q) = \text{ctm}(v_m)$, or (2) $\text{ctm}(p) = \text{ctm}(v_m) \wedge \text{ctm}(q) = \text{ctm}(v_s)$, or both. If this is not true then the structure does not comply with the mold. If only (1) is true it starts the procedure below with the mapping $\{p \mapsto v_s, q \mapsto v_m\}$ and tries to finish the mapping using the recursive process below. If only (2) is true it starts with initial mapping $f(q) = v_s, f(p) = v_m$ and starts the recursive process below. If both are true then tries the first mapping first and continues with the second mapping if the first mapping was unsuccessful.
- Select the source vertex of the mold. B = initial mapping. Go to 1.
  1. For each unselected edge $e_u = \{v_s, \alpha\} \in E(M)$ connected to the selected vertex v_s, perform step 2.
  2. Now find an edge $e_s = \{f(v_s), \beta\} \in E(G)$ for which it holds that $\text{ctm}(\alpha) = \text{ctm}(\beta)$, and $\text{t}(e_u) = \text{t}(e_s)$, and $\beta \in B \Rightarrow \alpha = f(\beta)$. If this is not possible this will not yield a valid mapping, therefore return.
     Otherwise, go to 3.
  3. For all valid options of e_s add $\alpha \mapsto \beta$ to B, select β.
     Recursively go to 1.

Algorithm Example Figure 8.20 shows a compound structure, the SUT, and a compound mold that will be used in an example for the mold isomorphism algorithm. For the edges the vertices that it connects together, the component type (in red) and the value (in blue) are shown. Other data that is saved in the edges is omitted for this example.

Figure 8.20(a) shows a SUT, an edge consisting of seven collapsed edges. It will be verified if the SUT complies with the mold shown in Figure 8.20(b). To be able to distinguish the vertices the unlabeled vertices are named x and y. The mold contains six edges, meaning that the number of edges in the structure is higher, and thus the algorithm will continue.

In Figure 8.20(c) the graph of the SUT if it would be fully expanded. However as described in the algorithm, parallel edges of the same component type are held together. Figure 8.20(d) shows what this would look like for the SUT. As can be seen the edges e_{10} and e_{11} keep collapsed in an edge e_{14}. Now the number of edges in the mold and in the SUT-graph are equal, hence the algorithm continues.

Next the vertices are grouped by their ctm. The vertex with label S in the mold has one R, one C and one L edge connected to it, hence its cmt is $\{R, C, L\}$. The vertex named x has two R and two C edges connected to it, hence its ctm is $\{R, R, C, C\}$. This is done for the SUT-graph as well.

Figure 8.21 shows the recursive process. Since $\text{ctm}(N_5) = \text{ctm}(S) \wedge \text{ctm}(N_1) = \text{ctm}(M)$ it start with the initial mapping $\{N_5 \mapsto S, N_1 \mapsto M\}$. As shown in cell A1 edge e_1, which is connected to S, is selected. The edge e_9 of the SUT-graph is found that meets the requirements of both component type and value, therefore $N_3 \mapsto x$ is added to B. However, as shown in cell A2 the process cannot continue any further since N_1 already is in B and $N_1 \not\mapsto S$. The recursive process returns twice and ends without finding a mapping.

However, since $\text{ctm}(N_1) = \text{ctm}(S) \wedge \text{ctm}(N_5) = \text{ctm}(M)$ holds as well, the process can be executed with the initial mapping swapped. This time each edge taken in the mold can be successfully mimicked. For example in cell C1 when e_3 is selected in the mold. The edge e_3 enforces a component of type C with value between 1 and 10. If e_3 is followed from the current vertex y it ends up in vertex M with ctm $\{R, C, L\}$. This can be mimicked by selecting e_7 of the SUT-graph, which is of type C with value 2. Edge e_7 ends up in vertex N_5 which has ctm $\{R, C, L\}$. Vertex N_5 already is in B, but since $N_3 \mapsto M \in B$ the last requirement holds as well.

Test Mapping

Once a structure is found that complies with the a mold, then the test mapping procedure of the plug-in will be applied to the structure and the test effectiveness will be calculated. The structure and its block are passed to the plug-in. In this procedure the test frequency and voltage are chosen, possibly supplemented with a setup time and test time. For this the engineers can use parts of the generic test method described in Section 8.2.1. For example, to calculate the current through false paths the MNA functionality can be used. As mentioned in Section 5.3 this part of the tool is not part of this research.

8.2.3 Passive Devices Test Mapping

The passive devices test mapping algorithm combines library mapping with the generic test method. The process is rather straightforward.

- First all structures in all blocks are divided into four groups; isolated structures without false paths, isolated structures with false paths, compound structures without false paths, and compound structures with false paths. For the plug-ins the same is done; plug-ins for isolated structures that cannot handle false paths, plug-ins for isolated structures that can handle false paths etc.
- Now for each group the algorithm iterates over all the structures and tries to find all tests for which structure complies with the mold of that test. The test mapping procedure is started and the effectiveness is calculated. Next it is checked if the settings of the test that is proposed by the plug-in are within the specified ranges of the test equipment. Then, only the tests that meet the specified ranges and contribute to a more effective test are selected, similar to what is described in Section 8.2.1.
- If no test is found the generic test method is used for that structure. Of course the test engineer can overwrite the setting and always use both the library tests and the generic test.

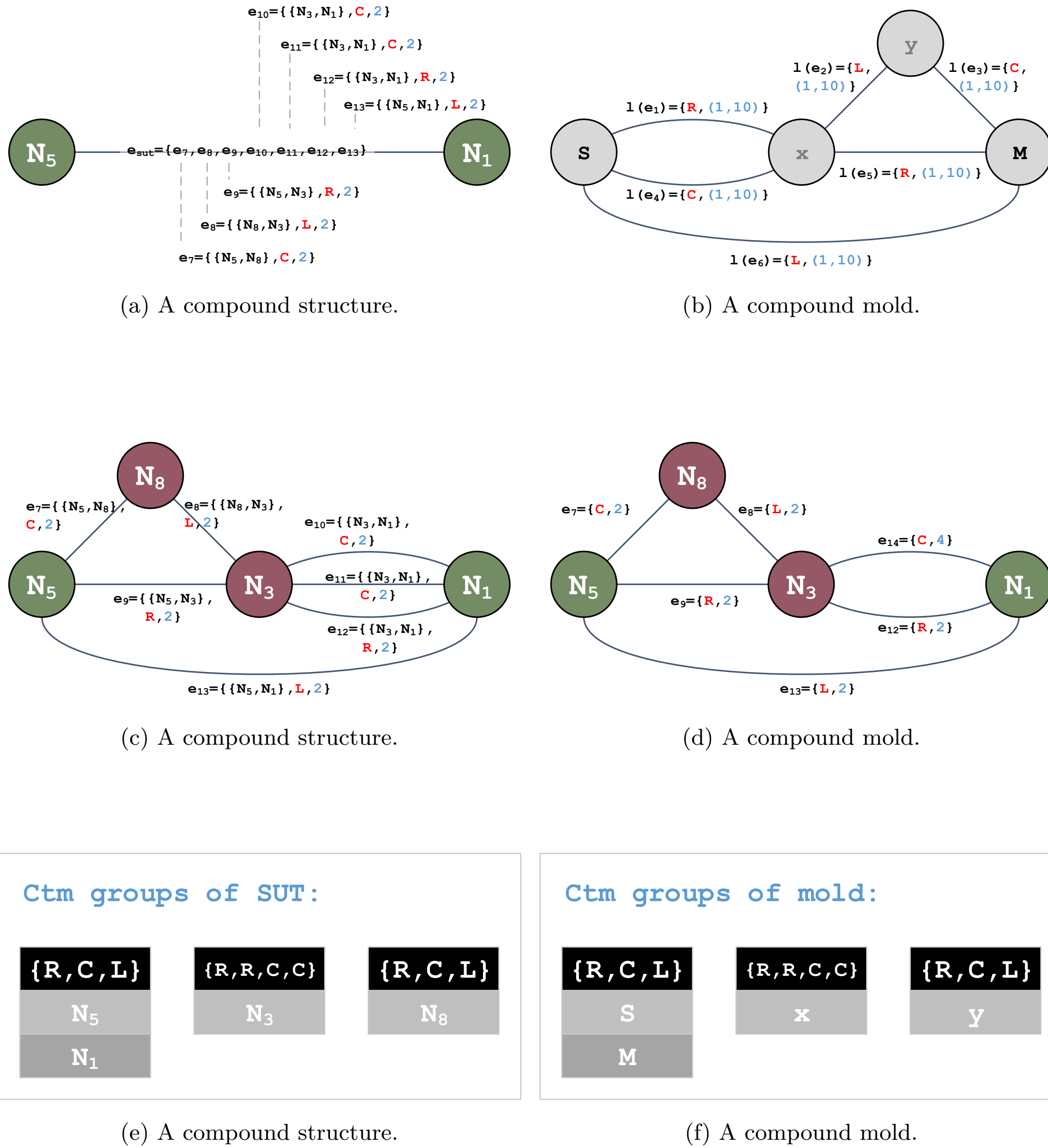


Figure 8.20: A compound mold as structure to which the mold isomorphism algorithm will be applied.

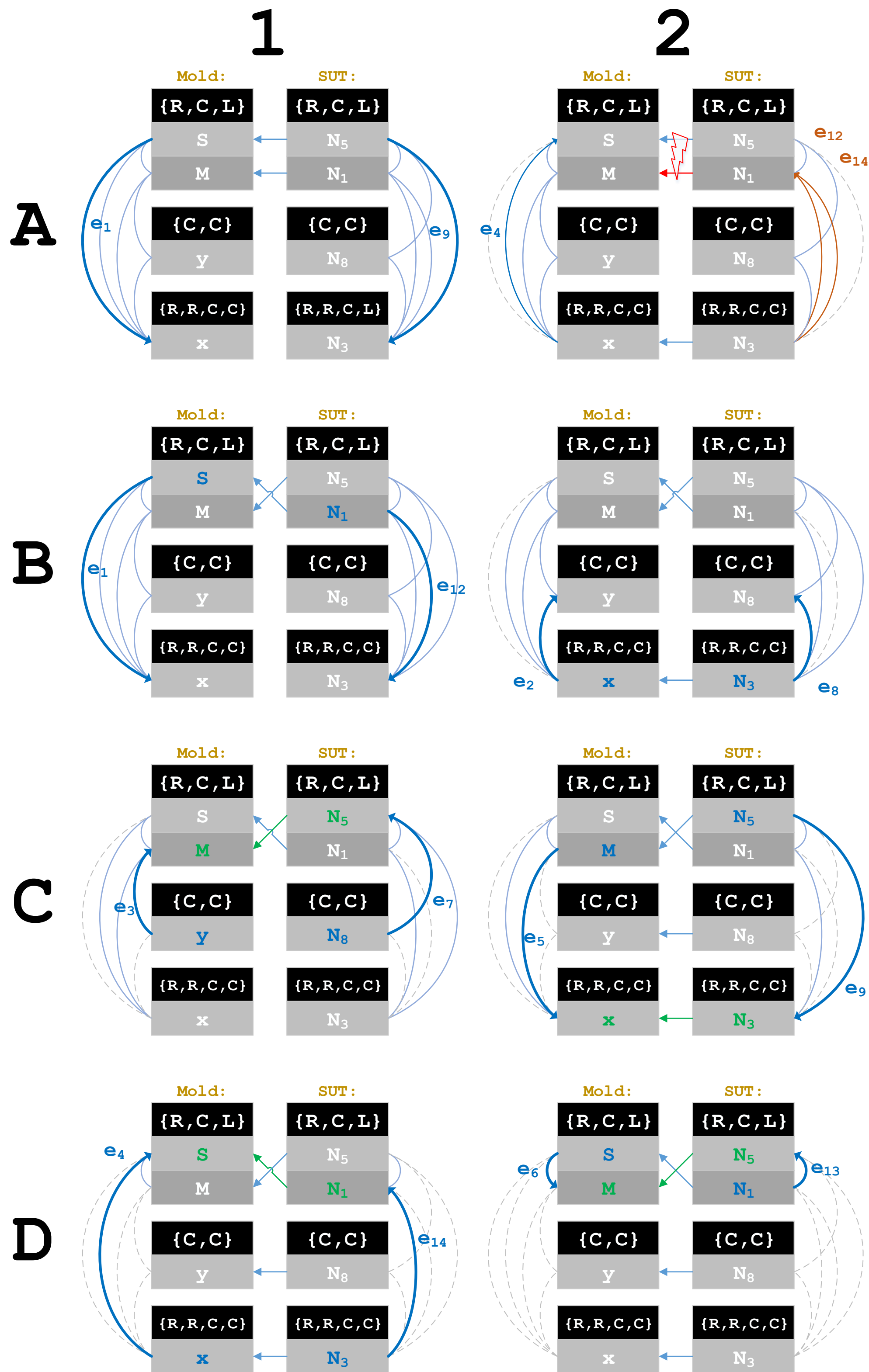


Figure 8.21: Example of a compound mold.

Chapter 9

Graph Decomposition for Boundary Scan

To show the versatility of the modular architecture of the tool and the opportunities the tool has to offer, a small proof-of-concept decomposer that can be used for boundary scan is created. This decomposer does not create the test patterns used to find the assembly defects as described in Section 3.2.5. Instead it focuses on finding the interconnects on the board by analyzing the graph.

The proof-of-concept limits itself to finding direct connections between IEEE 1149.1 devices, and connections containing a single discrete component. If a interconnect is connected to a pull-up or pull-down resistor this information is extracted from the graph as well. This information is needed to determine if a net is default '1' or default '0'. The approach the decomposer takes is as follows.

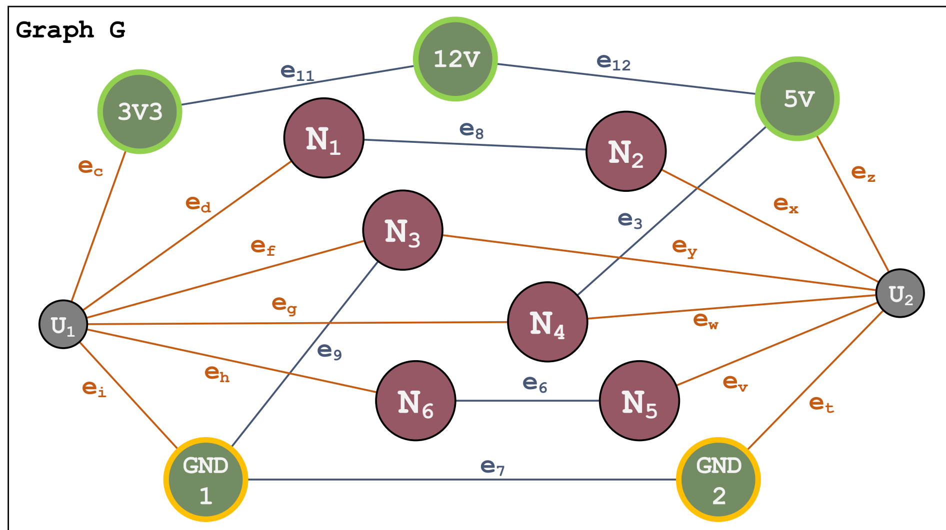
Some information that is needed to find the interconnects is currently not available in the graph. Currently all active components have a component type, but this does not distinguish the IEEE 1149.1 devices from ICs not supporting this standard. Furthermore all nets have a name that might suggest that it is a power or a ground net, but this is not conclusive. The test engineer has to manually select the IEEE 1149.1 devices from a list of all ICs and the power and ground nets form a list of all nets.

Once the test engineer has provided the needed information the tool will find the vertices that are representing the power and ground nets. The algorithm marks the neighboring nets that are connected via a resistor and subsequently removes the power and source vertices from the graph. Note that capacitors cannot be used for pull-up or pull-down. These vertices should be removed since these vertices can form a path between two IEEE 1149.1 devices but these pins cannot be toggled.

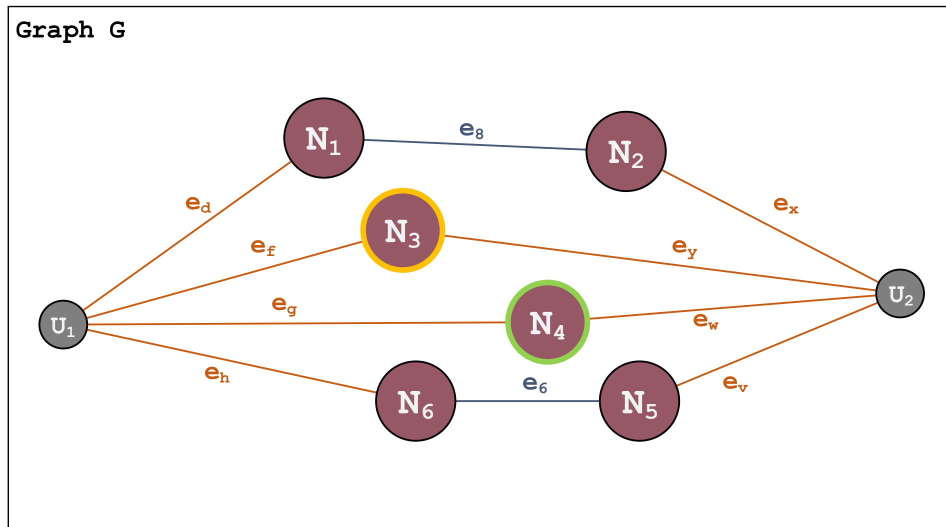
This also helps finding the other paths easier. Namely, all vertices that are exclusively connected to IEEE 1149.1 devices form a direct connecting between two or more devices for which it is sure that these are not connected to any other components.

Finally the last step in this proof-of-concept decomposer is finding single devices that connect two boundary scan devices together. If for both the vertices that are connected by the passive component it holds that all its other neighboring vertices are boundary scan devices, then that component forms a path between the connected devices.

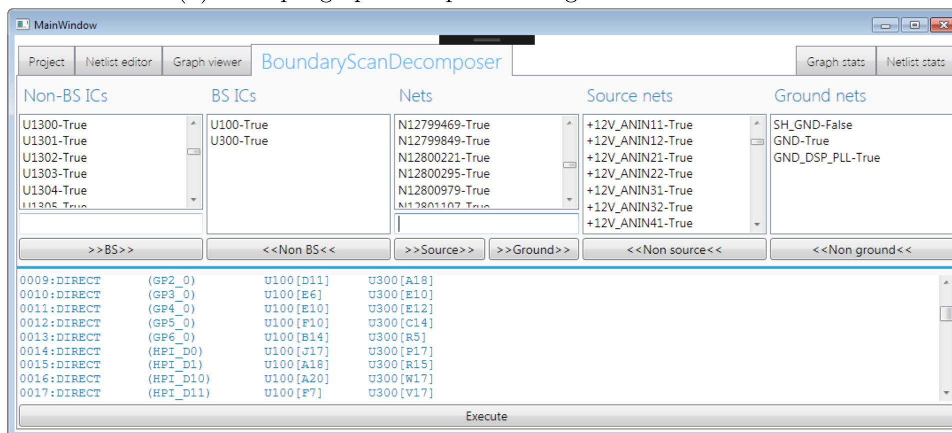
A small example is shown in Figure 9.1. Figure 9.1(a) shows an example circuit in which the vertices that are either power or ground are marked by a test engineer. Figure 9.1(b) shows the same circuit with the neighboring nets marked and the power and ground vertices removed. Figure 9.1(c) shows a screenshot of the tool in boundary scan mode.



(a) Example graph with power and ground nets marked.



(b) Example graph with power and ground nets removed.



(c) Screenshot of the tool in while executing boundary scan decomposition.

Figure 9.1: Boundary scan decomposer example.

Chapter 10

Experimental Results

The algorithms described in previous chapters are applied to seven different PCBs designed by Prodrive. In this chapter the results of these experiments are discussed. First the setup used for creating these results are described. This is followed by the results of transforming these designs into graphs. Finally the results of the algorithms are described.

10.1 Experimental Setup

A broad range of boards is chosen from the Prodrive product catalog to use for the experiments. Among the selected boards are both consumer electronics, industrial boards. Some boards are designed for ICT, but most are never intended to be tested with ICT. The boards that are used are:

- **QA** [6996-1303-0104] board is a PCB used for qualification of the AET-ICT hardware. With 218 components (17 active, 201 passive) and 268 nets this board is one of the smaller boards in this overview. Since this board is used for the qualification of the ICT hardware all nets except for one are reachable.
- The **ETMA** [6500-1100-5218] is a small board of a consumer electronics device. It contains 171 components of which 129 are passive. With 57 nets roughly half of the nets can be reached by a dedicated test pad. Another 20 nets can be reached by probing through hole components.
- The **MIUI** [6166-1000-2708] is an input board for a device that converts mains to the required output voltages of an X-ray generator. This input board is the smallest of all the boards compared with only 138 components and 69 nets. This board exclusively uses through hole components, meaning that all of the nets are reachable.
- The **MIUC** [6166-1000-1912] is the control board of the same device as the MIUI. This board is much larger with 1344 (of which 1006 are passive) components, connected by 847 nets. On this board 630 of the in total 766 nets that are connected to a passive components are reachable.
- The **EQDM** [6001-1243-2602] board provides sensor inputs and motor outputs for actuators. This is an interesting board as it, in contrast to the other boards in this overview, is designed for ICT. With 3028 components and 1977 nets this is the largest board in this overview. Of the 1744 nets connected to at least one passive component 1371 can be reached by a test pad, and another 99 by the leads of THT components.

- The **PASB** [6001-1232-7610] is a power supply board for a 260V, 50A DC power supply. This, containing 318 components, board mainly contains passive components but clearly was not designed to be used with ICT; only 67 nets of the 173 connected to passive components are reachable.
- The **PACB** [6001-1214-4203] is the control board of the same product as the PASB. The board contains 1234 components of which 1045 are passive. Of the 695 nets connected to a passive component, 564 are reachable.

Since it is not in the scope of this project, only a small number of tests are added to the test library. This is done to be able to test the mold matching algorithms, not to create actual ICT tests. All these tests are dummy tests that would not yield test that are usable. For example, the *single resistor test* is set to use a fixed frequency and output voltage. The tests that are in the proof-of-concept library are:

- **A single resistor test**. This test uses a simple mold that accepts structures of component type resistor, with the minimum and maximum value set to the minimum and maximum values that can be measured by the test equipment, as described in Section 6.3. For the AET these ranges are given in Section 4.1. This test does not allow false paths.
- **A single small capacitance test**. This test uses a simple mold that accepts a capacitor within the range of capacitance that can be measured by the test equipment using an impedance test. This test does not allow false paths.
- **A single big capacitance test**. This test uses a simple mold that accepts values higher than the maximum value that can be measured using an impedance test. It does this by using a $\frac{dv}{dt}$ test. This test does not allow false paths.
- A **parallel capacitor resistor test**. This test contains a mold of a resistor and capacitor in parallel. Both components need to have a value in the range specified by the test equipment. This test does allow false paths.
- A **series capacitor resistor test**. This test contains a mold of a resistor and capacitor in series. Both components need to have a value in the range specified by the test equipment. This test does allow false paths.
- A **Parallel capacitors test**. This test has a compound mold that accepts structures that exist of multiple capacitors in parallel. The minimum value is, just as with the single big capacitance test, equal to the maximum value the AET can measure as an impedance.

All timing results are from experiments executed on the tool using the selected board netlists. The tool was running in debug mode, on a PC with a Core i7-6820HQ CPU at 2.7 GHz, accompanied with 8 GB of DDR4 RAM running at 2133 MHz. Timings are determined using the $C\sharp$ `System.Diagnostics.Stopwatch` class. These numbers should only be considered as a reference, not as a exact results.

10.2 System Input Handling

In Table 10.1 an overview is given of the system input handling per board. The table shows how much of the (both internal and external) component information missing in the netlist could be fetched from the existing Orion database and the new internal description database. The number of edges and vertices of the graphs created from these inputs is shown per board as well. Note that ICs and transistors are represented by a vertex and an edge for every terminal of the component. Therefore the number of vertices and edges do not have to match the number of nets and components in a netlist.

Table 10.1: System input handling.

	External description (Orion)			**Internal Description**		**Graph**	
	Values	*Mounting Tech*	*Tolerance*	*Implicit*	*ICT Database*	*Vertices*	*Edges*
QA	19	68	55	67	5	276	244
ETMA	26	88	64	83	2	141	245
MIUI	13	19	10	18	0	71	145
MIUC	70	133	67	119	1	954	1967
EQDM	74	198	128	188	12	2194	5739
PASB	61	78	51	70	11	205	382
PACB	57	114	61	104	3	852	2073

Of this process a benchmark is executed. The results of this benchmarks is shown in Table 10.2. The timing of the ICT database and the Orion database are greatly fluctuating due to the dependence of the IP connections.

Table 10.2: System input handling timing in milliseconds.

	Parsing	**Fetch Orion**	**Gen. Implicit**	**Fetch Internal**	**Create Graph**
QA	37	4413	11	1049	22
ETMA	268	8715	14	1110	23
MIUI	229	1179	14	470	20
MIUC	1316	2683	15	565	45
EQDM	5730	3322	11	977	56
PASB	363	7710	19	985	19
PACB	1482	8046	14	722	26

10.3 Passive Decomposition

In this section the results of the the conducted experiments regarding the algorithms introduced in Chapter 8 are shown.

10.3.1 Connected Components, Blocks, and Collapsing

For all boards the connected and biconnected components are determined and shown as bar graphs. In these bar graphs the number of edges that remain after collapsing are shown in blue. The amount of collapsed edges are shown in orange.

As shown in Figure 10.1, 93 connected components are found in the graph of the QA-board. Most of these connected components exist of only one or two components (edges). There is one connected component of 69 edges after collapsing (84 before). This immediately shows the utility of finding the biconnected components. Namely, there are 148 biconnected components, meaning that the big connected component is divided into 55 separate blocks.

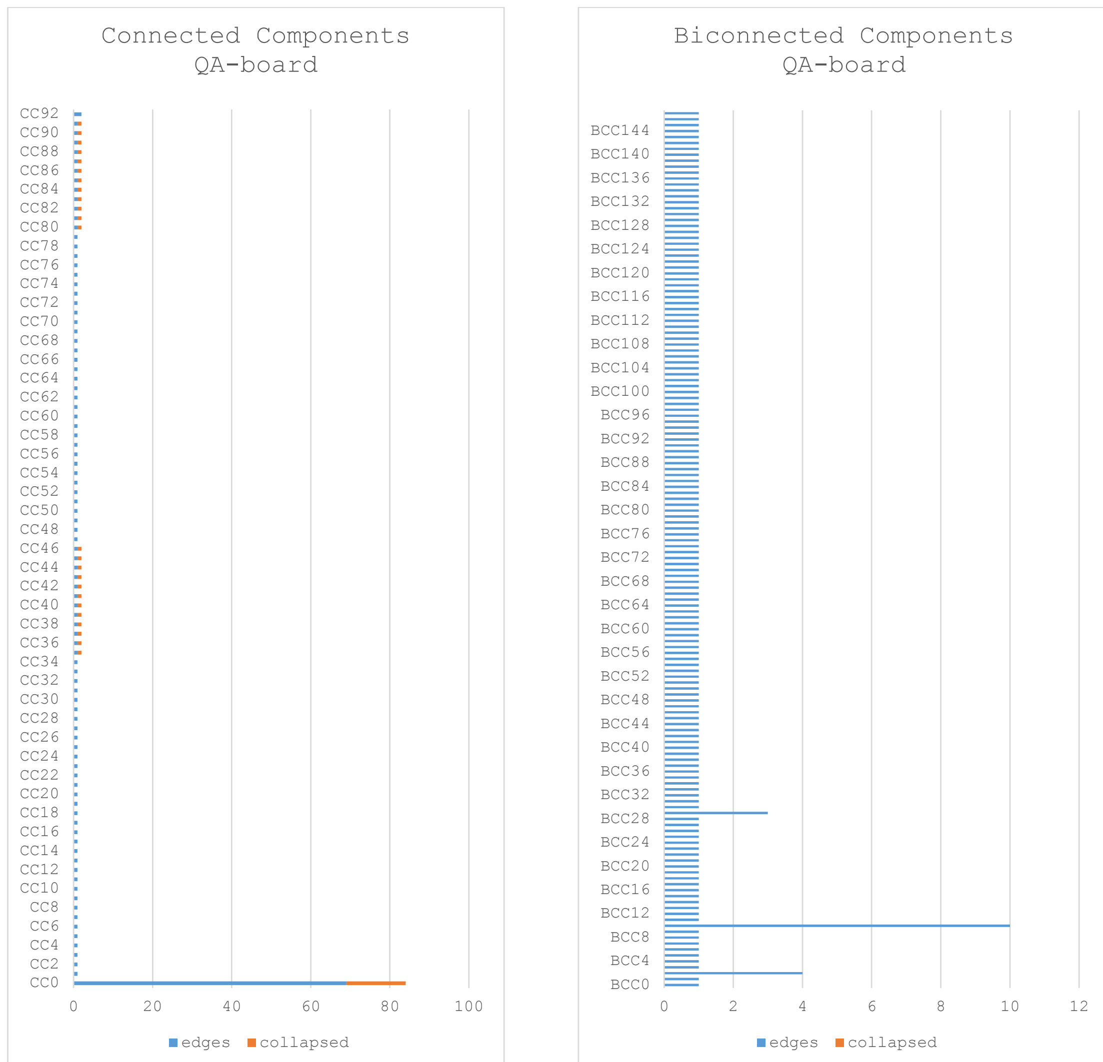


Figure 10.1: Connected components and blocks of the QA board before and after collapsing.

The other three smaller boards (ETMA, MIUI, and PASB) show very similar results. All three boards contain a connected component that hold a substantial part of the design. However, these boards show a greater number of collapsed edges. This is due to the fact that these boards are designed provide real functionality, in contrast to the QA-board which is only used for verification of the test equipment. The three boards, of which the number of connected components and blocks are shown in Figure 10.2, can be powered and therefore contain PSUs, which often contain multiple capacitors in parallel.

The PASB-board is the fist board in this overview that shows that edges can be collapsed after the biconnected components are found. Before block 4 (BCC4) of this board was found was collapsed it contained two articulation points without probe access.

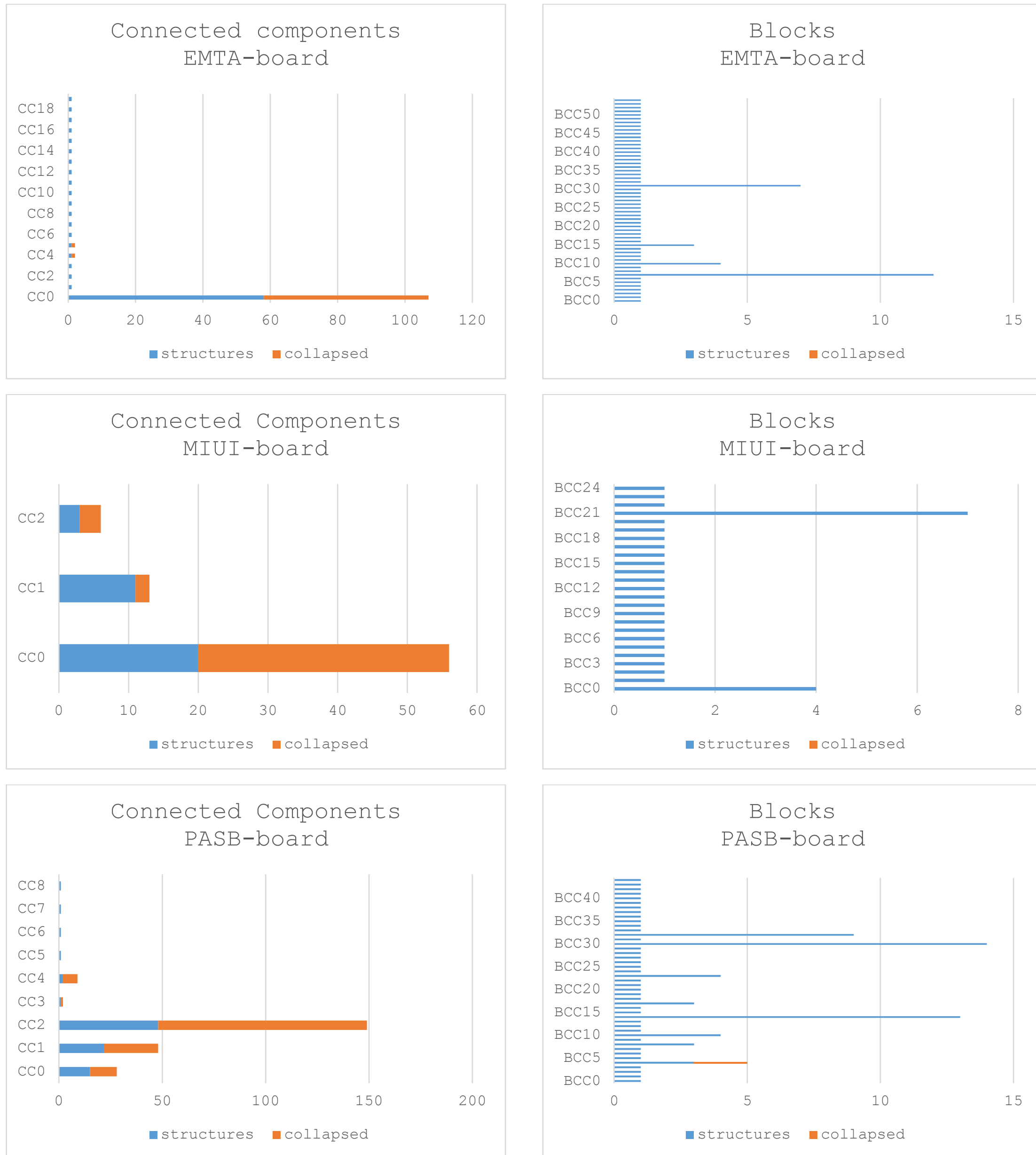


Figure 10.2: Connected components and blocks of the ETMA, MIUI, and PASB board before and after collapsing.

Table 10.3: Results for the connected components, blocks and structures for each board

	Connected Components			**Blocks**			**Structures**			
	#Components	*Biggest*	*Collapsed*	*#Blocks*	*Biggest*	*Collapsed*	*INFP*	*IWFP*	*CNFP*	*CWFP*
QA	93	69	39	147	10	0	118	13	27	4
ETMA	20	58	51	55	12	0	43	13	8	13
MIUI	3	20	41	25	7	0	9	8	13	10
MIUC	69	346	291	370	63	4	288	262	43	109
EQDM	223	1675	740	663	183	2	765	767	100	276
PASB	9	48	148	45	13	2	20	21	17	32
PACB	72	605	298	312	112	20	230	347	48	114

The three remaining boards have to many biconnected components to visualize. However, the results of these boards are included in Table 10.3 which shows an overview of the structures in the blocks of each of the boards. The table shows the amount of connected components that are found, the size of the biggest connected component (measured in number of structures), and the total of collapsed edges of each connected component combined. Furthermore it shows the number of blocks that are found, the size of the biggest block (also measured in the number of edges) and the total number of edges that are collapsed after finding the biconnected components. Finally the number of structures per group, described in Section 8.2, found inside of the blocks is shown. Table 10.4 gives an overview of the timing of the process of finding the connected components, collapsing and finding the blocks of each of the boards.

Table 10.4: Passive decomposing timing.

	Passive Decomposing Timing
QA	442 Ms
ETMA	52 Ms
MIUI	44 Ms
MIUC	853 Ms
EQDM	2705 Ms
PASB	48 Ms
PACB	688 Ms

10.3.2 Minimal Guards

The minimal-guard-set algorithm finds all possible minimal-guard-sets for each of the structures with a false path. In Figure 10.3 the numbers of all guard options are shown. On the horizontal axis the number of simple paths that are found around the SUT are shown. On the vertical axis the number of guards that are used to guard all of these paths are shown.

EQDM board has a total of 1043 structures with false paths (see Table 10.3).The algorithm has found a total of 4260 guard options for these structures. All of these except for four outliers are plotted in Figure 10.3. However, since there are only 1043 structures a lot of solutions have the same number of paths to guard. If then the algorithm finds two different minimal guard sets that use an equal number of guards, the solutions are on the same point in the graph.

Table 10.5 gives an overview of the amount of minimal guard sets that are found in total for each of the boards and the amount of time the algorithm required to find all these sets. Interesting in this result is the great difference in amount of sets that can be found for the MIUC and EQDM boards. Both are larger boards, but the algorithm finds the guard sets much faster on the MIUC board. This can be explained by the complexity of the EQDM board. The biggest number of simple paths that has to be checked is 38 for the MIUC board, while there are structures on the EQDM board for which 2871 paths need to be checked.

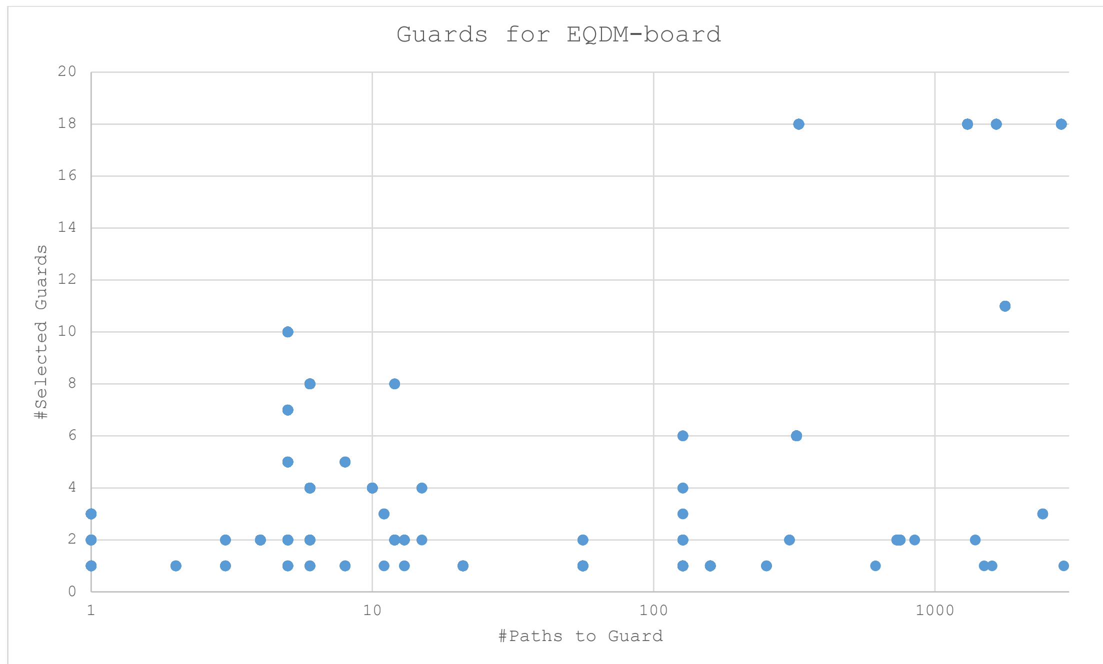


Figure 10.3: All guard options for the EQDM board.

Table 10.5: Amount of guard sets per board.

	#Guard Sets	Time [s]	Sets per Second
QA	45	1	45
ETMA	31	1	31
MIUI	31	1	31
MIUC	6807	13	532
EQDM	4260	623	6
PASB	66	5	13
PACB	3310	132	25

10.3.3 Mold matching

The mold matching algorithm has been benchmarked by using the tool to map the dummy tests of the experimental setup to all seven boards. The number of components that are mapped by a certain test are shown in the left ”Mapping distribution” diagrams, while the number of passive component per type are shown in the right ”Component distribution” diagrams. This benchmark focuses on matching the molds only. It does not consider all guarding options for the generic method.

In the figures below the mapping distribution of the seven boards are shown. Figure 10.4 shows the mapping distribution and the component distribution of the QA board side by side. As can be seen the number components with a test is equal to the number of passive components in the design. For the MIUI board the same holds, there is access to all of the components, hence a test could be mapped to all of the components. For all other boards there are components to which there is ”no access”.

In Table 10.6 the time needed to create the mapping can be found for all the boards. From this is can be seen that the number of mapping decrease if the complexity of a board rises.

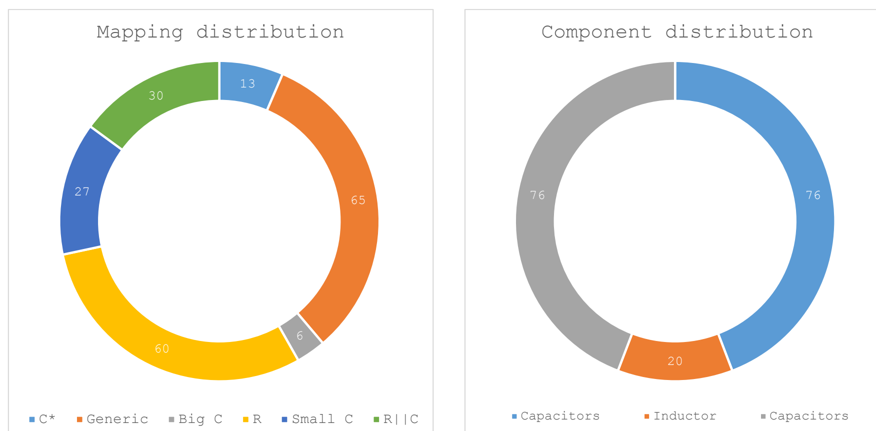


Figure 10.4: Mapping distribution of the QA board.

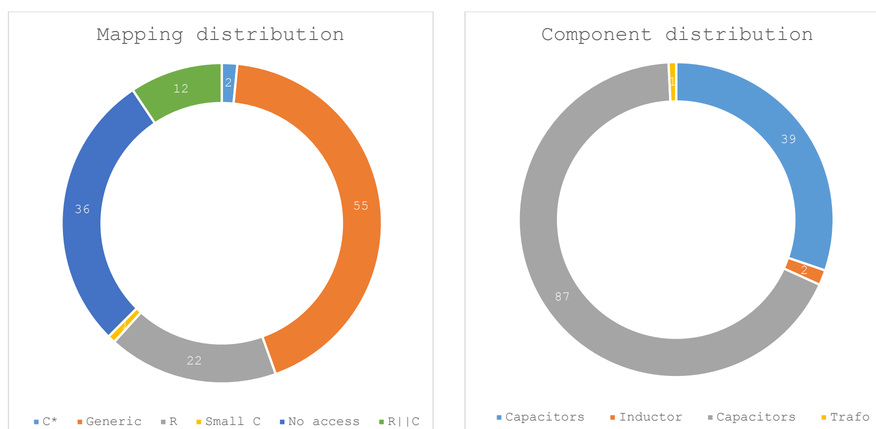


Figure 10.5: Mapping distribution of the ETMA board.

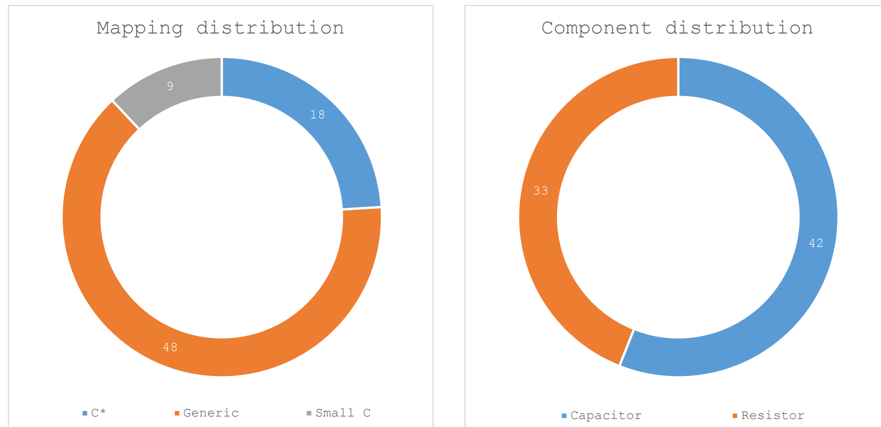


Figure 10.6: Mapping distribution of the MIUI board.

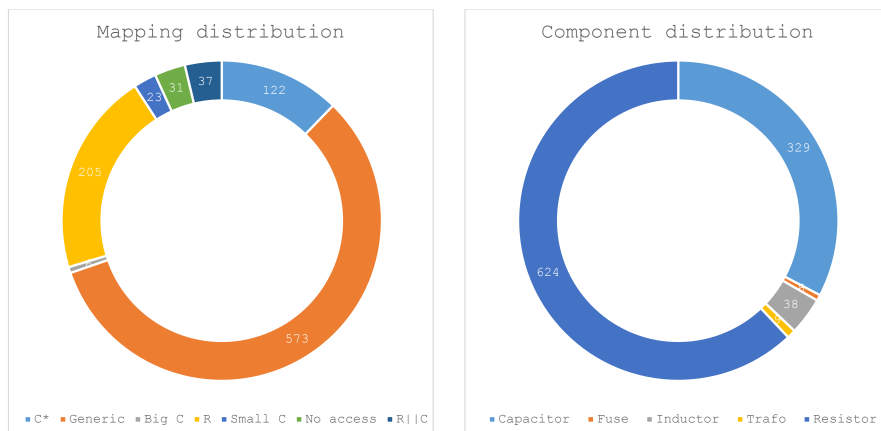


Figure 10.7: Mapping distribution of the MIUC board.

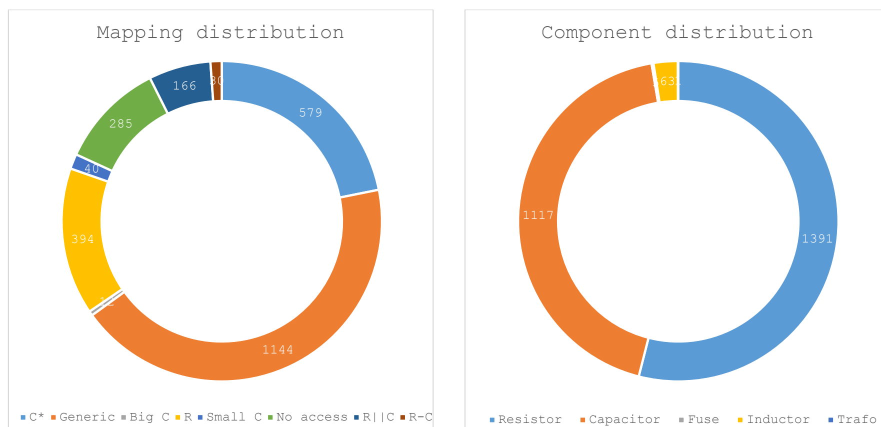


Figure 10.8: Mapping distribution of the EQDM board.

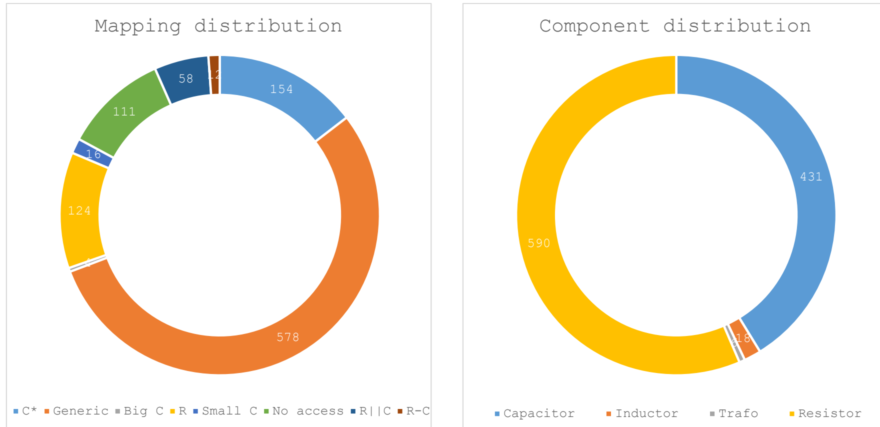


Figure 10.10: Mapping distribution of the PACB board.

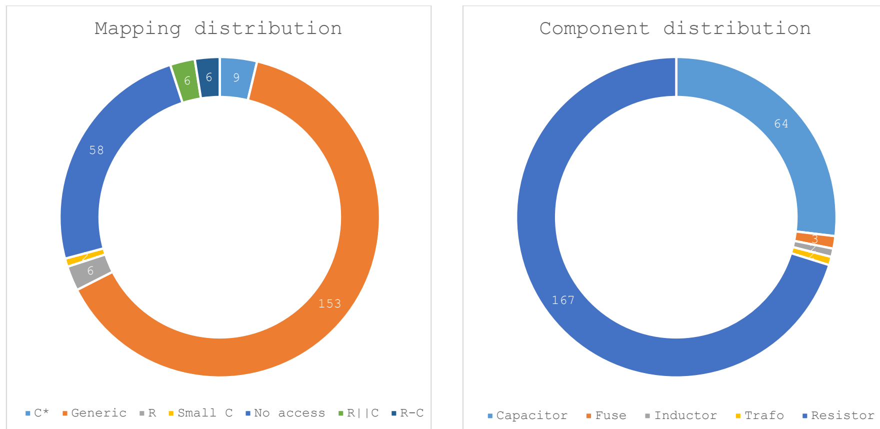


Figure 10.9: Mapping distribution of the PASB board.

Table 10.6: Amount of guard sets per board.

	#Guard Sets	Time [Ms]	Mapping per Ms
QA	161	45	3.58
ETMA	76	28	2.71
MIUI	77	37	2.08
MIUC	702	3406	0.21
EQDM	8194	8606	0.95
PASB	91	216	0.42
PACB	741	3226	0.23

Chapter 11

Discussion, Future Work, and Conclusion

11.1 Discussion

Overall the results of the previous chapter look very promising. Even though there currently are no real world tests executed that are designed by this tool, the algorithms to decompose the graph into smaller pieces and to map a library of test on these pieces seem to work.

For example, the EQDM board, the largest board is this comparison, has its complexity greatly reduced by finding the collapsed blocks of its graph. Where the full graph representation of this board holds 5739 edges, the biggest block in that graph holds only 183 edges. This is a reduction of 30 times the size of the original graph, while the decomposition of the graph took only 2705 Ms.

The minimal-guard-set finding algorithm clearly is more compute intensive. For the same EQDM board it takes around 10 minutes to generate all the possible options. Still 10 minutes is very acceptable since it will be faster than a test engineer would be able to do it.

Finally the mapping of the test library looks promising as well. In very little time the tests are mapped on the structures in the graph. Of course a more extensive library would take longer to map, but even with this tiny dummy library ofter more than half of the components can get a test mapped to it.

Some board however also proof to be not suitable to be tested with ICT. The mapping of the ETMA and PASB board show a very high amount of structures without access.

11.1.1 Future Work

While this thesis has demonstrated the potential of using graph theory for generating ICT tests, there are some more opportunities to utilize the graphs even further. Some opportunities that need further research are presented in this section.

Shorts Testing

The shorts testing methods as proposed by A. Suto are not real white-box testing methods. Even though precautions are taken for nets that are connected by fuses or other low resistance

connections, both methods do not predict the actual current that will flow through components connecting the groups together. The proposed solution [9] is to exclude one of the two nets connected to these components form the test to make sure that these low impedance components are always in series with another higher impedance components during the test. However, this does not overcome the problem of the phantom shorts, which is caused by multiple components in parallel between the two groups.

The graph structure allows for easy processing which components, or since not every net has probe access which paths of components, are in between the source and measurement groups. Using the MNA algorithm that is used in the generic test method, the steady-state currents that will flow from the source probes to the measurement probes can be predicted. Using these predictions the distribution of the groups can be altered to prevent phantom shorts.

Furthermore the proposed solution uses the linear method as a prefilter for the binary method. However, since phantom shorts can already be eliminated by predicting the currents that will flow, every short that is measured has to be a real short. Therefore it is unnecessary to immediately start the short finding routine. Instead the normal routine can be ended as normal, gathering extra information about the alleged short. This way the shorts testing method will be a prefilter for itself. This will eliminate the need for the linear method completely, speeding up the process even further.

Testing of Active Components

Large ICs are not suitable for ICT testing, discrete active components however are. For testing ICs boundary scan is a better solution. Using a small proof-of-concept it is shown that the graph can be used as a tool for automatically finding the interconnects between IEEE 1149.1 devices. This should be further explored.

For the discrete active components further research has to be conducted. Since active components themselves need a higher voltage to execute tests, the surrounding devices might get damaged. However, if for example a diode is surrounded by only passive components testing this device is easy. Namely, a diode allows current to flow in one direction only. Hence no matter what components surround the diode, the measured impedance will always be smaller if the diode is forward biased. By applying AC voltage across the diode the presence and orientation can be determined by looking at the peak voltages of the response.

Test Delay and Test Time

Currently the generic test method does not include a calculation for the delay and test time. Sometimes a test needs some time to stabilize before a reliable measurement can be made. The cases that will be covered will, in practice, always be the non-trivial cases since otherwise there would be a library test created for it. Therefore the tests generated by the generic test method can benefit from a reliable proposed delay and test time.

The biggest influence on the delay time is usually the amount of capacitance connected to a net. The capacitors connected to a net can be found in the graph. Subsequently the the charging curve of the capacitors can be predicted or minimized by preferring test on a higher frequency since the influence of capacitors get lower once the frequency rises.

Test Confidence Score

Currently the tests are selected merely on their PCOLA-SOQ score. However, if two tests have an equal effectiveness it is likely that one is still preferable over the other. Not because the test would be more effective, but because is is a more stable test.

Currently the test are selected in a first-come, first-served manner. If two tests are equally as effective, the one that was found first is chosen. However, while these test are all checked to be theoretically possible, the test that is selected might be very close to the measurement ranges of the test equipment. Therefore, another metric, the test confidence store, should be introduced to distinguish which of the tests with the same effectiveness is the most likely to be a stable test in production.

11.1.2 Conclusion

The graph representation of a board design has proven to be a versatile tool that allows for relative easy computation of the the false paths and the impedances between certain nets. By decomposing the graph into its biconnected components the computation complexity is log greatly lowered, meaning that more computation can be devoted to for example the compute heavy algorithm of finding all the possible guarding options.

By combining a netlist of the DUT, a description of the test equipment, a component database and an all new ICT database that hold the internal descriptions of complex components this graph can be built.

Test engineers can create tests that will be mapped be on both isolated and compound structures found in the biconnected components (blocks) of the graph. To accomplish this the engineer designing the test has to supply the test with a so called mold that acts as a filter to determine if the test can be mapped. Using this mold the tool can quickly determine if the test is suitable for a given SUT.

By utilizing the PCOLA-SOQ properties the effectiveness of a proposed test can be optimize and guaranteed. Test of which the effectiveness is dominated by the rest of the tests are automatically removed. In addition the PCOLA-SOQ score is also a well defined metric that allows the effectiveness to be compared to and even combined with the effectiveness of other test equipment.

Bibliography

[1] G. Leinbach, "Using Production Defect Data to Improve an SMT Assembly Process," *SMTA International*, September 2000. [Online]. Available: https://www.smta.org/files/SMTAI00-Leinbach.pdf 4

[2] K. Hird, K. P. Parker, and B. Follis, "Test coverage: what does it mean when a board test passes?" in *Proceedings. International Test Conference*, 2002, pp. 1066–1074, doi: 10.1109/TEST.2002.1041863. 7, 8

[3] G. Leinbach and S. Oresjo, "The Why, Where, What, How, and When of Automated X-ray Inspection," *SMTA International*, September 2001. [Online]. Available: http://www.keysight.com/upload/cmc_upload/All/Why_Where_What_How_When.pdf 9

[4] A. Buckroyd, *In–Circuit Testing.* Butterworth-Heinemann, December 2013, isbn: 1483112071. 10

[5] S. Scheiber, *Building a Successful Board-Test Strategy*, ser. Test and Measurement Series. Elsevier Science, 2001, isbn: 9780080476124. 11

[6] J. K. Berger, "New Directions in Loaded Board Testing," in *IEEE Automatic Testing Conference*, September 1989, pp. 212–216, doi: 10.1109/AUTEST.1989.81123. 11

[7] R. Matheson, "Second-Generation PCB self-Learn," *Computer-Aided Engineering Journal*, vol. 4, no. 5, pp. 209–212, October 1987, doi: 10.1049/cae:19870051. 11, 19

[8] A. W. Ley, "Defect Coverage for Non-intrusive Board Tests," 2009, ASSET InterTech whitepaper. [Online]. Available: http://www.smtnet.com/library/files/upload/defect_coverage_for_NBT.pdf 11

[9] A. Suto, "Faster Shorts Testing," *Evaluation Engineering*, August 2007. [Online]. Available: https://www.evaluationengineering.com/faster-shorts-testing 12, 68

[10] A. Hassan, J. Rajski, and V. Agarwal, "Testing and Diagnosis of Interconnects Using Boundary Scan Architecture," *IEEE Test Conference, 1988*, September 1988, doi: 10.1109/TEST.1988.207790. 12

[11] "IEEE Standard for Test Access Port and Boundary-Scan Architecture," *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)*, pp. 1–444, May 2013, doi: 10.1109/IEEESTD.2013.6515989. 12

[12] J. T. de Sousa and P. Y. K. Cheung, ser. Frontiers in Electronic Testing. Springer US, 2001, doi: 10.1007/b100750. 13

[13] M. J. Smith and N. Adams, "The Selection and Economics of Wireless Test Fixtures," in *Evaluation Engineering*, 2006. [Online]. Available: https://www.evaluationengineering.com/the-selection-and-economics-of-wireless-test-fixtures 16

[14] "IEEE Standard American National Standard Canadian Standard Graphic Symbols for Electrical and Electronics Diagrams (Including Reference Designation Letters)," *IEEE Std 315-1975 (Reaffirmed 1993)*, pp. 241–244, 1993, doi: 10.1109/IEEESTD.1993.93397. 24

[15] J. Hopcroft and R. Tarjan, "Efficient Algorithms for Graph Manipulation," *Communications of the ACM*, vol. 16, no. 6, pp. 372–378, June 1973, doi: 10.1109/TEST.1990.114074. 32, 33, 38

[16] R. Sedgewick, *Algorithms in C, Part 5: Graph Algorithms*, 3rd ed. Addison-Wesley Professional, 2001, isbn: 9780768685329. 32

[17] A. W. Brander and M. C. Sinclair, *A Comparative Study of k-Shortest Path Algorithms*. Springer London, 1996, pp. 370–379, doi: 10.1007/978-1-4471-1007-1_25. 32

[18] W. Hoffman and R. Pavley, "A method for the solution of the nth best path problem," *J. ACM*, vol. 6, no. 4, pp. 506–514, Oct. 1959, doi: 10.1145/320998.321004. 32, 35

[19] F. Harary, *Graph Theory*, ser. Addison-Wesley series in mathematics. Addison-Wesley Publishing Company, 1969. [Online]. Available: www.dtic.mil/dtic/tr/fulltext/u2/705364.pdf 33

[20] A. J. Suto, "Principles of Analog In-Circuit Testing," *Evaluation Engineering*, no. 51, pp. 18–21, December 2012. [Online]. Available: https://www.evaluationengineering.com/principles-of-analog-in-circuit-testing 38

[21] GenRad, Inc., "Introduction To In-Circuit Testing," pp. 81–85, January 1984, Concord, Massachusetts, USA. [Online]. Available: http://www.ietlabs.com/pdf/Handbooks/Introduction%20to%20In-Circuit%20Testing.pdf 43

[22] G. Rizzoni, *Principles and Applications of Electrical Engineering*. McGraw Hill, 2000. [Online]. Available: http://library.aceondo.net/ebooks/Technical_Education/Engineering_Principles_and_Applications_of_Electrical_Engineering_.pdf 44

[23] C.-W. Ho, A. Ruehli, and P. Brennan, "The Modified Nodal Approach to Network Analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, Jun 1975, doi: 10.1109/TCS.1975.1084079. 46

[24] P. Czimmermann, "The Graph Isomorphism Problem and its Applications," 2003. [Online]. Available: http://kifri.fri.uniza.sk/ojs/index.php/JICMS/article/download/385/146 50

Appendix A

Math Properties of False Paths

Lemma 2 (symmetric)**.** *For some graph G, let $\mathbb{P}_e$ be the false paths of some edge e, then:*
$\forall_{e_1,e_2}\left[e_1, e_2 \in E\left(G\right) : e_1 \in \mathbb{P}_{e_2} \Leftrightarrow e_2 \in \mathbb{P}_{e_1}\right]$

Proof. Let $e_1 = \{a, b\}$ and $e_2 = \{s, m\}$ be two edges in graph G. If $e_1 \in \mathbb{P}_{e_2}$ then by definition of a false path there has to exist a simple path $p = s \overset{p_1}{\rightsquigarrow} a \overset{e_1}{\longrightarrow} b \overset{p_2}{\rightsquigarrow} m$. The concatenation of p with e_2 gives a simple cycle $c = s \overset{p_1}{\rightsquigarrow} a \overset{e_1}{\longrightarrow} b \overset{p_2}{\rightsquigarrow} m \overset{e_2}{\longrightarrow} s$ containing both e_1 and e_2. Removing e_1 from this cycles results in a valid and existing simple path $b \overset{p_2}{\rightsquigarrow} m \overset{e_2}{\longrightarrow} s \overset{p_1}{\rightsquigarrow} a$ which by definition is a false path of e_1.

Hence it has to be true that if e_1 is in a false path of e_2, e_2 also has to be in a false path of e_1. The same reasoning can be applied to prove that if e_2 is in a false path of e_1, then e_1 also has to be in a false path of e_2. Therefore it has to hold that being in a false path is a symmetric relation. □

Lemma 3 (transitive)**.** *For some graph G, let $\mathbb{P}_e$ be the false paths of some edge e, then:*
$\forall_{e_1,e_2,e_3}\left[e_1, e_2, e_3 \in E\left(G\right) : e_1 \in \mathbb{P}_{e_2} \wedge e_2 \in \mathbb{P}_{e_3} \Rightarrow e_1 \in \mathbb{P}_{e_3}\right]$

Proof. Let $e_1 = \{s, m\}$, $e_2 = \{a, b\}$, and $e_3 = \{c, d\}$ be three edges in graph G. Now, assume that $e_2 \in \mathbb{P}_{e_1}$ and that $e_3 \in \mathbb{P}_{e_2}$. This means that by the definition of false path there exists two simple paths $s \overset{\alpha_1}{\rightsquigarrow} a \overset{e_2}{\longrightarrow} b \overset{\alpha_2}{\rightsquigarrow} m$ and $a \overset{\beta_1}{\rightsquigarrow} c \overset{e_3}{\longrightarrow} d \overset{\beta_2}{\rightsquigarrow} b$. From this it follows that there exists two other paths $s \overset{\alpha_1}{\rightsquigarrow} a \overset{\beta_1}{\rightsquigarrow} c$ and $d \overset{\beta_2}{\rightsquigarrow} b \overset{\alpha_2}{\rightsquigarrow} m$. Now, by applying Lemma 1 it follows that there has to exist a simple s, c-path p_1 and simple d, m-path p_2. Concatenating p_1,e_3, and p_2 results in a valid and existing simple path $p_3 = s \overset{p_5}{\rightsquigarrow} c \overset{e_3}{\longrightarrow} d \overset{\alpha_1}{\rightsquigarrow} m$.

By the definition of a false path it can be concluded that p_3 is a false path of p_1. Hence being in a false path is a transitive relation. □