Development of Test Automation Framework for Printed Circuit Board Assembly

B Aravind Balaji¹, S Sasikumar¹ and K Ramesh²

¹Electronics and Communication Engineering, Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India
²Computer Science Engineering, Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India

Corresponding author: balaji.aravi@gmail.com

Abstract. A Test Automation Framework is a bunch of rules or coding guidelines for test-process handling that is followed during coding. These are simple strategies that produce helpful results such as improved code re-usability, higher portability, diminished script support cost, reduced interdependency, higher interchangeability, and so on. This paper presents the development of a test automation framework for Printed Circuit Board Assembly (PCBA) to test the functionality and fault identification of the electronic circuit in the PCBA. It provides an integrated array of test and measurement Instruments which is automated using SCPI (Standard Command for programming Instrument) based commands to control test and measurement processes, making it simpler to configure and query. These test instruments imitate the environment in which the PCBA is to be deployed, which helps to observe its characteristics to determine whether it meets the expectation for which it was designed. Any possible variations in its characteristics due to fault are observed during testing and those faults are identified automatically using the data-driven fault analysis method.

1. Introduction

Printed Circuit Board Assembly (PCBA) is an electronic circuit that is to be tested during the manufacturing process to crosscheck whether it meets the functional requirements. Selection of rite testing ideology reduces the manufacturing cost since testing plays a role of 30% in overall manufacturing cost. Different testing Strategies are in practice to test the electronic circuits [3]. This paper focuses on an improved approach to functional testing, overcoming its inability over fault diagnosis. Typically test and measurement process plays an inevitable role in the quality of functional testing. Automation of such processes provides huge Test coverage and limiting the factor of “time to market”. Functional testing mimics the nature of the environment where the PCBA is to be deployed and evaluates overall module function and electrical characteristics. Various characteristics/properties are under consideration to replicate the environment. It is achieved using an array of test and measurement instruments using which the desired generation and acquisition of such properties are carried out. To provide an efficient automation process this paper provides a comprehensive exposure to Test Automation Framework for PCBA functional testing and fault identification. A framework is an overall Skeleton of an automation process that helps for code organization, process handling, decision making, fault analysis, and much more.

Conventional test management frameworks verify the functionality of electronic assembly using test
signals generated from different instruments to check whether they are working or defective [1], [2], [4], [6] and not focused on fault analysis for faulty components identification. Basically analog circuits are nonlinear and are more difficult to diagnose fault when compared to digital circuits. This paper focuses on fault analysis in analog circuits. Fault analysis is carried out by different methods in analog circuits to identify parametric (soft) faults and catastrophic (hard) faults [15-20]. Generally, there is limited access over the circuit node in functional testing compared to in-circuit testing which reduces the fault locating capability. Hence to improve the testing efficiency and identify fault location, node accessibility is increased by using “bed of nail” fixture and matrix-based switching to access these nodes [6], [7].

Instrument control lacks two properties namely coding compatibility and exchangeability. The different instrument control method is in practice [8] that can eliminate this drawback. Traditional control methods are carried out by transferring commands in form of ASCII characters between the test application and the instrument to be controlled [5]. These commands were not consistent such that each instrument vendor had an appropriate command set, leading to difficulty in programming while using similar instruments of unlike vendors. As a result, the entire process is reprogrammed as per the vendor's command set during the upgrading process. To eliminate this difficulty a Text-based command so-called SCPI is used as a standard command commonly used by instrument vendors making script compatibility much easy [9-11].

Similarly, connection bus between instruments and test applications is supported using various protocols including RS232, USB, Ethernet, GPIB, and much more [6], [8], [10], [11], [12]. Interchange of connection bus between these protocols gives rise to re-scripting during refurbishing. Virtual Instrument Software Architecture (VISA) is a software architecture that consists of a set of APIs that is used nearly for all bus connections, which makes the control of instruments no longer, rely on particular bus connection and can transplant among various buses [8], [10], [12].

From the Literature review different test management frameworks for functional testing, and their drawbacks in lack of fault detecting capability were reviewed, and various control methods over instrument control and fault analysis methods were discussed. With the above reference, this paper presents an exposure on the development of a test automation and management framework for PCBA to test its functional behavior along with fault location capability; integrating an array of test and measurement instruments which is automated using the script-based standard command set so-called SCPI [21]; VISA based APIs to interconnect different bus communication within the same platform [21]. Finally, generate a fault report. Fault analysis is carried out using the machine learning technique where the best fit model is developed by predicting the relationship between the feature extracted and the predicted fault.

2. Design Flow

The figure 1 Illustrates the design process, starting from the test application which observes, controls, and monitors the whole process; communicates with an array of instruments via different bus connections with the help of VISA APIs; Controls the test fixtures to accesses the test nodes in Unit Under Test (UUT); validates the instrument response; analysis and locates the faults; finally generates a fault report. The functionality of the UUT is tested by replicating the environment in which the unit is to be deployed and verify the behavior. This is achieved using an array of instruments that mimic the environment setup. Certain parameters are to be extracted from the UUT for validations. Test cases are developed which extracts these parametric details using test and measurement equipment/instruments. These instruments are controlled by the test application by sending script-based instructions (SCPI: often called “Skippy”) to configure and query the instrument's response.

SCPI instructions use a hierarchical structure framed in form of a tree system [22]. Each instruction consists of a header followed by one or more configuration parameters. The header/identifier is the
root node; these instructions are encapsulated/framed by the test application and send in form of string to the instruments.

Example 1: CONF:VOLT:AC:RANG 750V, Example 2: SENS:VOLT:AC:RANG?

Instruction tree (Example 1)

CONFigure:
  └── VOLTage:
        └── AC:RANGe {<range>|DEF|MIN|MAX}

Instruction tree (Example 2)

SENSe:
  └── VOLTage:
        └── AC:RANGe? [DEF|MIN|MAX];

Where:
CONFigure, SENSe - Root identifier
VOLTage - sub identifier.
AC - sub identifier.
Colon: - Isolates back-to-back identifiers.
Curly Braces {} - Parameters.
Angle brackets < > - value.
Vertical bar | - Splits parameters.
Semicolon ; - Isolates header within same root Identifier.
Question mark ? - request Query.

The examples given above show the instruction to configure the AC range in voltage mode to 750V and Query the AC range in voltage mode.

Figure 1. Design flow diagram.

These string instructions are transferred between the test application and set of instruments supporting various Bus connections (Ethernet, USB, GPIB) via VISA APIs modules in form of sessions, each
instrument form a session. Write module and read modules are used to configure and query the instrument response.

![String Indexing](image)

**Figure 2.** Parsing SCPI command.

The instruments are probed to the circuit’s test nodes of the UUT, providing the stimulus inputs to the UUT. Also reads the response as per the configuration and query invoked by the test application and revert it in form of string-based instructions. The response of each instrument are parsed and parametric values are extracted for example figure 2 above shows the response reverted from an AC source as per the query raised by the test application to measure the AC source i.e 2.2808E+02 is the response string that is to be extracted as 2.2808 x 10^2 which implies as 228.8VAC. The returned array of a string is spitted or subset using the index method and parametric value is extracted. These values are validated by the test application by comparing them with a reference value to determine if the response is within the acceptable range. Any deviation in the response invokes fault analysis.

Test application accesses circuits test nodes of the UUT using test fixtures which are controlled using a matrix of switches to switch between test nodes. Any deviation in the UUT response invokes the fault analysis test case which gets access to different test nodes by switching the desired instrument probe between different test nodes with help of switches. The response dataset from the instrument is reverted to the test application. These datasets are provided to a learning model which gets the input data set as a test dataset and compares it with the existing data set in the fault dictionary to classify and locate the faults in the circuit. Finally, the cumulative fault report is generated as per the feedback from the model. Meanwhile, the response of UUT at normal conditions is logged automatically during consecutive test cases throughout the process.

### 3. TEST AUTOMATION FRAMEWORK

A Framework is a basic structure of an application architecture that manages the process flow, following a set of strategies while coding. It is a cluster of processes that supports test automation. The entire test process is subdivided into sub-codes making it easy to reuse the code for different applications during refurbishing. Figure 3 shows the design diagram of the test automation framework, the whole framework runs under the test application so-called LabVIEW. It’s a virtual automation platform that uses a block-based coding technique making it simpler to code by the “drag and drop” concept [14], [23]. The framework is segregated into four consecutive layers starting with the data extraction layer, as the name implies it is used for dataset extraction from the response of the UUT as per the stimulus. VISA sessions are used for connecting an array of instruments such as Electronic load, voltage source, multifunction meter, Scope, signal generator, etc. These instruments provide the required source and payload to the UUT to replicate the environment where it gets deployed. Multi VISA instances are invoked using parallel computing with the help of a distinctive channel for each instrument. VISA Driver supports the write module and reads module [24], to send configuration instructions and read queries about UUT response. Every instance initiates with an open modules/session which initials a communication link among the test application and instrument. The open modules are executed outside the test case loop keeping the instance active throughout the process. Each instance is activated with the resource parameters including distinctive address, Port number and Bus interface (Ethernet, USB, GPIB) of the instrument connected. Once the communication link is established, write/read modules are used to command the instrument to send the UUT response.
Once the instrument acknowledges, the response is pushed to receive buffer for post-processing and the next test case is initiated with the help of the test case controller. The case controller determines which case is to be executed as per the data fetched by the default register and a shift register. The default registers fetch the default case during the first iteration of the test case followed by is the shift register which takes control from the second iteration and the same process repeats. N+1 test cases are executed one after another in sequential processing. Close module/session is provided at the end to terminate each instruments instance.

The second layer is the validation layer which fetches the data from the data extraction layer to post-process the data. The receive buffer gets the data from the data extraction layer and pushes it to the Range finder. The data is compared with the reference limits using the windowing method to validate whether it is of the specified range. If the data is in desired range the data is pushed to the report generation layer; shift register is fetched with next test case to be executed which is forwarded to case controller.

In other cases, if the data is not in the desired range, the fault invoker gets invoked by the range finder. Fault invoker initials the interlock system which flushes all the instruments and registers value to initial/default condition halting the test process. Similarly, the Shift register is fetched with the fault test case which is pushed to the case controller initiating the fault case. Various test nodes of the UUT are probe using instruments along with matrix switching to extract the data set. The third layer is the Fault analysis layer that gets the dataset from the validation layer, these data are so-called test data set, and are fetched to ML APIs. The ML application program is invoked using the open module with resource parameters to configuring the application in a specific version. Node modules are used to get the data set vectors and build a classification model.

| X1 | X2 | X3 | X4 | Label |
|----|----|----|----|-------|
| 1  | 1  | 1  | 1  | \textit{f}_1 |
| 1  | 1  | 2  | 1  | \textit{f}_2 |
| 2  | 2  | 3  | 1  | \textit{f}_3 |
| 3  | 3  | 3  | 2  | \textit{f}_4 |
| 4  | 2  | 4  | 2  | \textit{f}_5 |
| 4  | 3  | 4  | 2  | \textit{f}_6 |
Figure 3. Test Automation framework.

The model is used to identify the faults in the UUT and locate the fault. Generally, analog circuit faults can be analyzed using different methods, out of which the Data-driven method is implemented using the Machine learning (ML) technique in this framework [15]. Initially, the historical dataset of possible fault conditions is collected before using circuit simulation and a fault dictionary is created using the dataset's in an appropriate pattern. Fault dictionary is a form of tabular column that consists of two variables namely the independent and dependent variables in subsequent columns and samples in each row respectively. The Feature vectors are called independent variable $X_i$ ($i$ is the number of features) and the target or labels are called dependent variable $f_j$ ($j$ is the number of faults). Where $X_i$ is the features extracted from the faults and $f_j$ is the predicted faults. The table 1 illustrates an example table of a fault dictionary with six samples and four features of the current value of an analog circuit.
power source varied in accordance to faults [16]. This data set is considered as training data set using which a machine learning model is trained and developed to analyze the test data and deduce a function that represents the association between the feature vectors \( X_i \) and label/target \( f_j \), using which multi-classification of faults is done. Finally, close modules are used to terminate the ML session after classification. The final layer is the report generation layer. It is used to log the data from the range finder (during the normal process) and multi classifier model (during fault analysis). Table formatter is used to log the data in the appropriate rows and columns. A reference mechanism is used to segregate the test samples using the serial numbers and manage file location. Other configurations such as auto-save and window minimize are carried out by the report generation layer.

4. TEST AUTOMATION ALGORITHM

The above steps are the proposed algorithm for the test automation framework of fault analysis and location of analog circuits.

Step1: Execute test case to extraction features of UUT response using an array of test instruments from M sessions.

Step2: Validate the actual response with the expected response.

Step4: If the actual response is in range shift to the next test case, repeat from step1 for \( N+1 \) test cases.

Step5: If the actual response deviates, shift to fault case resetting to initial Condition.

Step6: Extract \( X_i \) feature vectors from \( T_k \) test nodes.

Step7: Predict \( f_j \) target in relation with feature vector \( X_i \) using fault Dictionary.

Step8: Export Fault report.

5. TEST AUTOMATION RESULTS

Thus the functional test framework as discussed is implemented successfully and tested its performance connecting a UUT to the array of instruments. figure 4(a) shows the runtime response of the hardware connected to the UUT. The arrays of instruments are probe to the UUT via test fixture which makes access to the test nodes using spring-based pogo pins attached at the bottom of the test fixture.

![Figure 4. Test Automation framework.](image)

The UUT is provided with the voltage source and its output response is noted at different payloads,
Fault analysis, and the interlock system is cross-checked using a fault sample. The figure 4(b) shows the report generation of normal operation in which the Actual response is within the range of the Expected response.

6. CONCLUSION

The Test Automation Framework for PCBA to test the Analog circuit was created in contrast to conventional functional testing equipment, providing a better compatible and exchangeable framework along with a fault analysis technique that reduces the “Time to Market” and achieves a higher “test coverage area” that can be used for different test application to test the functionality of Analog circuits.

REFERENCES

[1] Jan Hrbaceka, Radek Hrbacek, and Jakub Lewinsky 2019 Automatic Mechatronic Test Stand Development for Embedded Electronics Using NI LabVIEW. International Conference Mechatroncs, springer 113-18
[2] Reinhardt S, Butschkow C, Geissler S, Dimaichner A, Olbrich F, Lane C. E, and Hüttel, A. K 2019 Lab::Measurement—A portable and extensible framework for controlling lab equipment and conducting measurements Computer Physics Communications 234 216–22
[3] Sabapathi, S 2018 The Future of PCB Diagnostics and Trouble-shooting IEEE AUTOTESTCON.Georgiana 45-54
[4] H. C, Ana-Maria B, and Ioan L 2018 Automatic testing of automotive electronic modules for cranking conditions International Symposium on Electronics and Telecommunications
[5] Wanzeng Cai, Binwen Wang and Shoulong Zhang 2017 Remote Control and Data Acquisition of Multiple Oscilloscopes Using LabVIEW International Conference on Computer Technology, Electronics, and Communication (ICCTEC) 920-24
[6] Serban M, Vagapov Y, Chen Z, Holme R, and Lupin S 2014 Universal platform for PCB functional testing International Conference on Actual Problems of Electron Devices Engineering (APEDE) 402-409
[7] Zhang M Z, Wang H Q, and Chen Y. M 2013 Development of PCBs Functional Tester with Relay Matrix Network Applied Mechanics and Materials 336-338 76–79
[8] Zeng Z, Zhu X, Qi S, Shen X, and Cai S 2018 Preliminary Exploration of Instrument Control Technology IOP Conference Series: Materials Science and Engineering 394
[9] Zaiming F, Zhixiang Z, Yijiu Z and Min M 2017 The merging design method of instrument software based on the SCPI command set 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI) 44-48
[10] Chen Y, Liu Y and Jiao X 2017 Design of LXI bus and SCPI analytical platform for battery simulator 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE) 315-321
[11] Liu gui-li and Kong quancun 2013 Design of Virtual Oscilloscope Based on GPIB Interface and SCPI 11th IEEE International Conf Electronic Measurement & Instruments 294-98
[12] Wei cheng and Fenggui wang 2015 Remote automatic test system based on MATLAB using VISA over LAN IEEE International Conference on Advanced Mechatronic Systems 384-87
[13] Jiang chao, Xu wu-bin and Li bing 2013 Design of Instrument Control System Based on LabVIEW TELKOMNIKA Indonesian Journal of Electrical Engineering 11 6 3427-32
[14] Chance Elliott, Vipin Vijayakumar and Wesley Zink 2007 National Instruments LabVIEW: A Programming Environment for Laboratory Automation and Measurement Journal of the Association for Laboratory Automation, 12 1 17-24
[15] Yang H, Meng C, and Wang C 2020 Data-driven Feature Extraction for Analog Circuit Fault Diagnosis Using 1-D Convolutional Neural Network. In: IEEE Access 1–1 8 18305-315
[16] Zhang C, Pan X, He G, and Yu J 2018 A Fault Dictionary Method for the Diagnosis of
Catastrophic Faults Using Power Supply Current 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE) 295-98

[17] Saeedi S, Pishgar S. H, and Eslami M 2019 Optimum test point selection method for analog fault dictionary techniques Analog Integrated Circuits and Signal Processing

[18] Ma Q, He Y, and Zhou F 2016 A new decision tree approach of support vector machine for analog circuit fault diagnosis Analog Integrated Circuits and Signal Processing 88(3) 455–63

[19] Yang C, Zhen L, and Hu C 2019 Fault Diagnosis of Analog Filter Circuit Based on Genetic Algorithm IEEE Access 7 54969–80

[20] Ke Huang, Stratigopoulos, H.-G, and Mir S 2010 Fault diagnosis of analog circuits based on machine learning Design Automation & Test in Europe Conference & Exhibition

[21] Aravind Balaji B, Sasikumar S, Ramesh K  SCPI based integrated test and measurement environment using LabVIEW IOP Conference Series: Materials Science and Engineering 1045

[22] SCPI Consortium, Standard commands for programmable instrumentation. (SCPI) http://www.ivifoundation.org/docs/scpi-99.pdf

[23] National Instruments Corp,USA, LabVIEW Measurements Manual[M] 2003:5-4-5-10.

[24] Virtual Instrument Software Architecture (VISA), https://www.ni.com/visa.