SIMULATION AND FAULT DIAGNOSIS IN POST-MANUFACTURING MIXED SIGNAL CIRCUITS

Kyle Pawlowski, Sumit Chkravarty, Arjun Kumar Joginipelly
Kennesaw State University
GA, USA
kpawlow1@students.kennesaw.edu; Schakra2@kennesaw.edu; arjunrao143@gmail.com

ABSTRACT

A major problem in circuit board remanufacturing is the identification of parametric faults from age or stress to the individual passive components. We propose a deep machine learning system for simulating and identifying such faults. A simulated dataset is generated for the most common faults in a circuit. This dataset is used to train deep machine learning classification algorithms to identify and classify the faults. The accuracy of system is measured by comparing with real circuit boards in operation.

Keywords: Fault Diagnosis, Deep Machine Learning, Circuit Simulation, Neural Networks, Pattern Recognition & Classification.

INTRODUCTION

In the industry of post-manufacturing circuit board repair, there are different problems than those in the manufacturing process. While in manufacturing, issues are caused either by procedural errors or by design discrepancies, post-manufacturing faces problems of components falling outside of their specifications from age or overuse. These are called *parametric faults* and are difficult to diagnose compared to other faults which are more catastrophic.

Most of the current research in fault diagnosis deals with manufacturing, but post-manufacturing is also important. Many consumers would rather buy used products when the cost of new parts is high. Buying used parts can also be necessary when some components of mission critical systems fail and the original manufacturers no longer sell the replacements. Work in post-manufacturing can help guide new applications as well. Knowing how circuits fail after years of use can guide designs for better performance and durability. Companies that sell lasting, quality products earn reputations for doing so and their products are desired in the marketplace.

Despite the obvious differences, much of the research in the area of manufacturing can be applied to post-manufacturing as well. Some current work [1], [9] focuses on digital circuits where the use of machine learning methods is not exploited. For these circuits, traditional methods have been sufficient for finding faults. For mixed-signal circuits, there are many more possible states since the components have a continuous range of values they can take on, so other studies [2],[11] have used machine learning methods like genetic algorithms to find faults. This study uses a different machine learning approach for similar circuits. Many other works in this area [4] incorporate information about the circuit layout in their program at either the logic level, the schematic level, or the physical level. This system, however, only passes the circuit inputs and outputs into the network. This prevents the network from needing to understand electrical components on a universal level but makes it so that the network would need to be retrained whenever it needed to be applied to a new circuit.

Our method of fault prediction uses simulation of faulty circuit components to generate a dataset for a deep machine learning classification algorithm. After the faults are simulated, the results are framed into a database to provide supervised training for our deep network. Results are compared against the real circuit with known faulty components to verify the accuracy of our simulations.

SIMULATION

For this study, we simulated a circuit board which is used and remanufactured in actual industry. The schematic for this board can be seen in Figure 1. We chose this board specifically for its simplicity since it contains a limited number of components and can be simulated quickly without too much computing power. Despite being small, it still contains a variety of components which can be analyzed. These include both analog and digital components which are handled in different ways in the simulation. Using an existing circuit board allows the results from this study to be tested in a realistic situation and to prove the feasibility of this method for real operation.

First, the most common possible faults for the circuit being examined are identified and considered as categories of faults. This study considers both parametric faults in analog components, where component parameters fall outside of their specifications, and logical faults in logic components. Our study considers the faults most commonly found in capictors which include an increased serial resistance or a decreased capacitace. These can be found in capacitors that have experienced dielectric breakdown from age or overuse [3]. Since many of the circuit boards repaired in the remanufacturing process are exposed to harsh conditions like constant use and exposure to the weather, such faults are common. Faults for logical components are considered using only a few high-level logical faults. For the case of the inverter shown in this study, we using the inverter being stuck-on and stuck-off as the possible cases. These are commonly seen in Integrated Circuits (IC) when connections within the IC are either bridged or broken by stress or age.

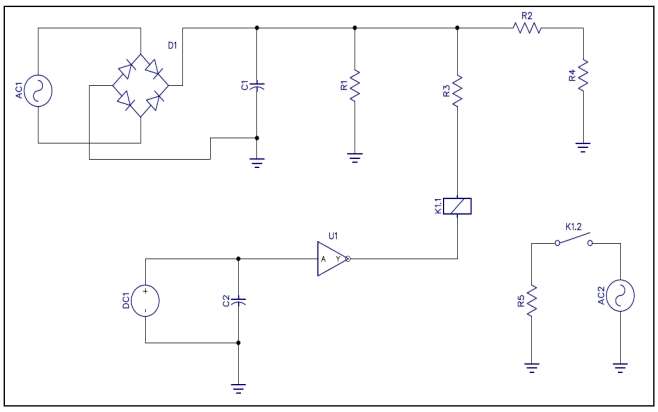

Figure 1. Schematic of the circuit used.

For each category, we run simulations across a range of values for the faulty parameters. For capacitors, these are chosen to be capacitance values of 80% or less of the nominal capacitance and serial resistance values 200% or more of the nominal serial resistance as falling outside of that range will start to cause circuit failures [8]. For analog components, this method of simulation can generate thousands of examples of component behaviors ranging from just outside of acceptable values to catastrophic failure. Presumably, the amount of data which can be accumulated in this way is limited by how ridiculously out of specification a component can realistically be and how similar consecutive parameter values can be within the defined range while still providing useful information. The goal is to provide as many examples as possible of circuit configurations that could be found in the field since these will make the network adaptable to all of these situations.

To represent the case where no component is faulty, simulations are performed for every component across the range of values which are considered to be within specifications. This is necessary to generate a comparable number of examples when the ranges of values for any one component are more rigidly bound than in the case of actual faults. Usually, the simulator needs to use much finer steps when iterating through parameters to generate enough data to keep up with the rest of the categories. Not having enough examples of this case could result in the network not being able to identify when there is not a fault. Having a no-fault case is a valuable feature which can prevent the network from having costly false alarms. Potentially, having this category can be avoided in a situation where the circuit being tested is known to have some fault and all possible faults are considered as categories. Eliminating the no-fault case could be done if there was a separate functional test of the circuit perform beforehand, but that is obviously not ideal.

The same technique can be used for digital components which do not naturally have a large range of failure behaviors. Digital components can be represented by all of the individual transitors and how they behave in the circuit, but this is tedious to simulate and does not provide much benefit in a practical sense. Knowing exactly which transistor failed in a logic IC does not help the technician who is trying to fix the circuit. Without representing the digital components at a much lower level, faults can only be represented in a handfull of ways since their failures are not parametric in nature. The logic level of the component can only be considered to be stuck at high or stuck at low. However, we can still iterate through the acceptable values of the analog components for each logical fault in the same way as with the no-fault case. This way, enough examples can be generated to create the necessary ground truth.

Every simulation creates an array of fault behaviors as characterized by the sequences of readings from different test-points in the circuit. Each test-point where voltage data is read from is chosen for being connected to the input and output connectors on the real circuit board. With this setup, the same connectors which are connected to the board during regular operation can be used to read important data from the board. Each test-point produces a sequence of data which is concatenated to form one channel in a block of sequences. This data block is saved and organized by the generated fault category. Each fault category ends up having hundreds of examples each with multiple sequences of test point data for a set time period. Organizing the data in this way makes it simple to pipe it into a deep learning network.

DEEP LEARNING MODEL

The goal of this study is to classify the sequences of voltages into the correct fault category. To accomplish this, we use a *Long-Short Term Memory* (LSTM) [5] model which is a *Recurrent Neural Network* (RNN) designed to analyze sequences in time. LSTMs are often used with analyzing spoken and written language [12] where patterns that the network has seen in the past can affect how it interprets the present information. These can also be applied to voltage data in analog circuits where readings from a single point in time cannot provide enough information for identifying faults. We also use a primitive form of *time/frequency*-LSTM (TF-LSTM) [6]. This is done by including the Fast Fourier Transform (FFT) of the voltage-time data to give the network an impression of the differences in the frequency behaviors of the components. This is especially useful in our case where the effects that capacitors have on circuits can be extremely subtle if noticeable at all.

In our model, several channels of voltage data are passed into an LSTM block as shown in Figure 2. This LSTM block contains multiple LSTM layers which take in a piece of the sequences representing the current moment and outputs from the previous LSTM layers acting as a memory of past events. Each layer has parameters which indicate the importance of past layers and affect how much a layer will weigh the past events. These parameters, along with others, are tuned to produce the lowest possible loss during training. The output from the final LSTM layer is then fed into a fully connected layer. The fully connected layer condenses the dimensionality of the network and produces a number for each fault category which correspond to how likely it finds that category to be the correct fault. The classification layer interprets which of those categories the proceeding layers have found most likely and returns it as the fault category.

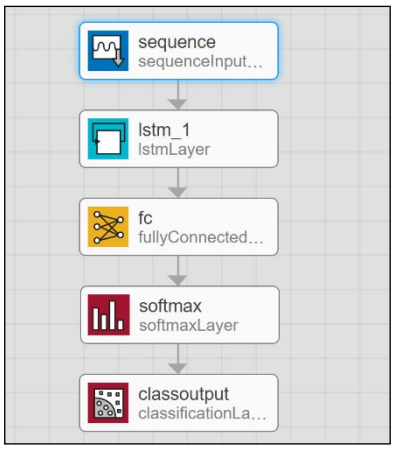

Figure 2. Network architecture

Since the LSTM uses supervised learning, it must learn how to tune its parameters to produce the desired result through examples which are provided at training time. To train the model, we feed in the fault behaviors data generated by our simulations as the training data and the corresponding faults as the associated labels. Each example contains a label designating which fault category it falls into and two sequences for every test-point: one for the voltage-time data and another for the magnitude-frequency data. In training, the network can run its classification algorithm on the input data, compare its answer against the provided label, and adjust the parameters of its algorithm accordingly. The network can repeat this process thousands of times to complete its training.

The steady improvement due to this training process can be seen in Figure 3, where the accuracy (in blue) goes up, and the loss (in orange) goes down as more iterations are run. The improvements start to level off as the network reaches its optimal solution. More training iterations at this point provide little improvement. The darker lines running in the middle of the trends are the results from validation data. This data has not been used in training but is only for testing purposes. This validation trend is much more stable than the rest of the regular training results. This is because the regular data the network is trained on is randomly selected while the validation data has been evenly selected from all parts of the dataset. The peaks in accuracy come from selections of data that contain more examples which are towards the extremes. The troughs are from selections where component parameters are close to the threshold where they are no longer considered faults. This shows that the network can perform much better when given data from more extreme faults. Less catastrophic failures are more difficult for the network to find.

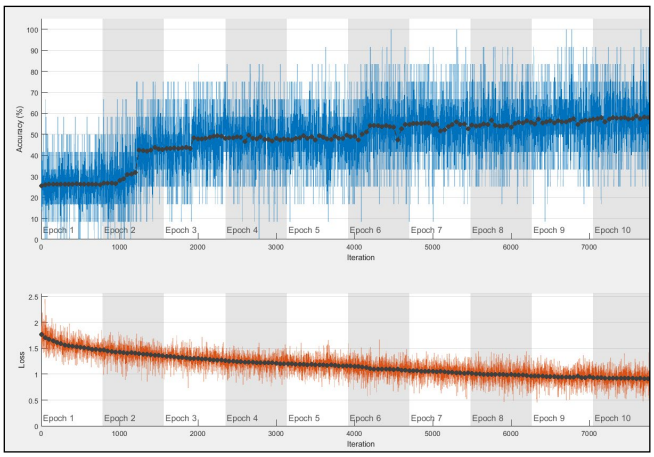

Figure 3. Changing network accuracy and loss during training

Once training has completed, the network can analyze data from a real circuit in operation and predict what single fault is present in the circuit. This post-training process, called a *forward pass*, is much less computationally expensive then the training process and can be performed relatively quickly. This is one of the advantages of a machine-learning algorithm. Although the computation required is burdensome to start, the final result is fairly efficient. This is especially important from a practical perspective where a test system needs to avoid costly hardware for it to be viable for real applications.

RESULTS

A number of different variations of the same architecture have been tested for their effectiveness in producing useable results. The standard method involved having five fault categories, one for no faults (*F0*) and two for each of the two capacitors in the circuit (*F1-F4*). Each capacitor had a category for having too large of a serial resistance (*F1* and *F3*) and another for having too small of a capacitance value (*F2* and *F4*). This network was able to capture the behaviors of some of the fault categories achieving an accuracy of 60.0%. Being able to differentiate between these categories with this level of accuracy shows that the network was able to tune its parameters during training so that it can capture some of the circuit behavior. The area where the network falls short is in identifying the faults *F0* and *F4* as shown in Table 1. This could be because the capacitor causing this fault has only a subtle effect on the circuit acting as a smoothing capacitor for high-frequency noise. The capacitor causing *F1* and *F2*, on the other hand, is a large value capacitor which the network has no problem identifying as the cause of the fault.

Table 1. Results of five-category network test

Fault	Total Examples	Correct
F0	612	80
F1	660	638
F2	600	483
F3	653	643

F4	611	5

Although a component can have fail for different reasons, all will result in replacing the component for someone trying to repair that circuit. Because of this practical reality, a network was also trained that only had one fault category for each component. This network had two categories for the capacitors and one for the no-fault case. The network was still trained and tested using the same data from the different causes of the component failure, but these were all considered to be a part of the same fault category. This means both a small capacitance and a large serial resistance in the first capacitor would be considered put in the *F1* category. The hope here is that the network can perform better when it does not need to waste resources on differentiating between categories which ultimately produce the same result.

This network performed well as shown in Table 2. getting a total accuracy of 80.5%. Combining the categories from the same capacitors appeared to positively affect performance and having fewer categories to choose from greatly increased the chances of a correct answer from the network. One of the more surprising improvements was in *F0*. Despite not being combined with any other categories as compared to the last network, it has been helped by the reorganization and retraining. This could be because the network can dedicate more of its parameters towards classifying this fault with fewer categories to worry about. Still, *F0* is the consistently the category these networks have trouble with. Since there is a hard line where we have considered components to belong in one of the other fault categories but the range of parameters in the training examples are still relatively continuous, this no-fault category ends up with many examples which are extremely similar to the fault categories. That makes these edge cases difficult for the network to categorize correctly.

Alternatively, the same result could be accomplished by just combining the categories from the previous five-category network during the classification step instead of retraining a new network. Table 3. shows this method performs worse than the network trained for three categories achieving only a scaled accuracy of 68.1%. This supports the idea that the subtler details that can be gained by representing every fault separately do not help the network have a better understanding of each component as a whole.

Table 2. Results of three-category network test

Fault	Total Examples	Correct
F0	612	277
F1	630	606
F2	632	625

Table 3. Results of repurposing five category network

Fault	Total Examples	Correct
F0	612	80
F1	660	648
F2	600	483
F3	653	650
F4	611	610

Comparing Table 3. to the results found in Table 1, there is a jump in the number of *F4* examples that could be identified. This means that much of the inaccuracy in the regular five-category network came from confusing *F3* and *F4*, both parametric faults in the same capacitor. This could be because an increased serial resistance and a decreased capacitance can lead to some of the same circuit behavior. Combining these categories has also helped the retrained network in Table 2. since it did not need to tune its parameters for classifying between the two categories. Depending on the application, differentiating between these could be necessary, but for most cases it would not be and the increase in accuracy certainly makes this method more desirable. If such a network is needed, steps would need to be taken to improve the performance of the network for it to produce reliable results.

Another network also included the inverter as a cause of faults. Looking at the inverter at the logic level, we considered two fault conditions: stuck on and stuck off (*F5* and *F6*). Adding the inverter could not help with finding the faults in capacitors but could help with the overall accuracy of the network since its faults are not parametric in nature as in the capacitors' case and therefore, might be easier to diagnose. This will also provide a more comprehensive solution for fault diagnosis. By including logic components, a single network should be able to diagnose the faults for an entire board in a single pass. This network does a decent job as shown in Table 4. It still suffered from some of the same problems the other networks did. It still had trouble differentiating between the two different faults for the same capacitor in *F3* and *F4*, and *F0* was still a problem likely for the same reasons as discussed earlier. As expected, the network was able to perform well on the categories for the inverter which helped the network achieve an overall accuracy of 64.2% putting it above the previously shown five-category network in Table 1.

Table 4. Results of seven-category network test

Fault	Total Examples	Correct
F0	612	59
F1	660	655
F2	600	557
F3	653	0
F4	611	306
F5	611	611
F6	611	611

We also trained a network which included the inverter as a source of faults but considered only one fault for each component. This resulted in a network with four fault categories to choose from. Unlike in the case of detecting capacitor faults which has consistently improved with the

combination of their fault categories, the network performance suffered from having the two inverter cases combined into one fault as shown in Table 5. This is likely because while for capacitors, having an increased serial resistance and a decreased capacitance can result in similar behavior, the cases of an inverter being stuck-on or stuck-off obviously have significantly different effects on the circuit behavior. Despite this shortcoming, the network still achieved an accuracy of 74.7% providing a huge improvement over the seven-category model in Table 4. This means the improvements in the other categories made up for the losses in the inverter category.

We also tried the same technique used for the other networks of combining the fault categories at the classification step. Although it has the advantage in the inverter category as shown in Table 6, this method fell short of the retrained method getting only a scaled accuracy of 64.0%. The difference here provides some insight into the advantages and disadvantages between these networks. Combining fault categories before training seems to help when there are more capacitors in the circuit. This rule could extend to other analog component as well. However, for logic components, the other method of combining categories after training, during the classification step, is better. A combination of the two methods could use different classification methods for the different types of components though this could be tedious for larger circuits. For a large circuit, one method would need to be chosen over the other depending on what components primarily make up the circuit.

Table 5. Results of four-category network test

Fault	Total Examples	Correct
F0	612	143
F1	630	626
F2	632	632
F3	611	455

Table 6. Results of repurposing seven-category network

Fault	Total Examples	Correct
F0	612	59
F1	660	656
F2	600	561
F3	653	326
F4	611	306
F5	611	611
F6	611	611

We were also able to use the network on the real circuit in operation. One of the faults was created by putting a 15Ω resistor in series with one of the capacitors. This is comparable to an old capacitor which has started to break down from years of wear. We were able to connect the circuit to a fixture that simulates the conditions the circuit would be under in regular operation and connected some of the input and outputs of the circuit to a digital oscilloscope. To confirm the accuracy of the simulated data for this circuit, we compared the data graphically, as shown in Figure 4, and found that our simulations were accurate. There are minor differences caused by some of the circuit properties which are not considered by the simulation, but they are close enough for the purpose of this study. After running the real observations through the network, we found it was able to successfully identify the cause of the failure from four other fault categories without ever seeing data from a real circuit before. This demonstrates the viability of this approach for real applications.

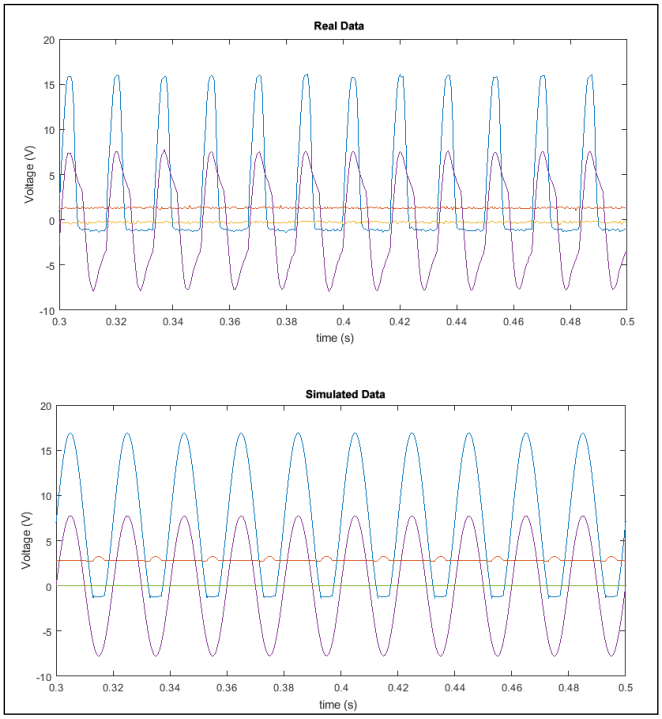

Figure 4. Comparison of real data and simulated data

The test-points chosen for this circuit were such that they were connected to the physical pin on the board which our fixture plugs into. Because of this, a slight modification of the fixture could allow this method to be used just by plugging the right cables into the connectors already on the board. This process could also be applied to any number of other circuits. If there is a schematic available for the circuit, it can be simulated, potential faults can be identified, and this process can be implemented. It takes a little bit of work to get started, but the benefits are enormous if this system is used on high value circuit boards.

FUTURE WORKS

The results found in this study are promising and could be expanded in future works. First of all, though the deep-learning architecture shown in this paper uses advanced techniques such as LSTMs, it is still fairly simple having only one LSTM block with 40 hidden layers. Exploring what would be the most optimal network architecture for this application could make this method more viable. Adding

more layers could help the network represent more complex relationships in the circuit behaviors. Although LSTMs are designed for sequences in time, other papers [7] have been found to achieve some of the same effect without the use of RNNs when tested on machine translation tasks. Using other methods instead of LSTMs could make training faster if they are more easily parallelizable. Other machine learning methods could also be added to improve speed and efficiency of the training process.

Also, this method involves training one deep-network for each circuit that needs to be analyzed. During training, the network captures the behavior of the specific circuit it is being trained for, so it cannot be applied to any other circuits without retraining. More studies can be done on the possibility of making a network that can identify faults in any circuit. Such a system would likely need to incorporate schematic information about the circuit as well as the inputs and outputs that this study has used. Those pieces of information together could possibly allow the network to understand how different components behave in general. An interesting part of this would be figuring out how to codify schematic information so that it can be passed in as an input to the network.

In addition, this study has been performed on a small circuit which has proved that the concept works, but work has yet to be done on adapting it for larger circuits. As components start to have a smaller relative effect on the outputs of the circuit, their failures will be harder to detect. The network could be modified to capture these more subtle behaviors. Another problem that could arise is different components creating similar changes in the outputs of the circuit. The components shown in this study are far enough apart that they create noticeably different effects on the circuit, but in a larger circuit, this might not be the case.

Finally, one of the limitations of the method shown in this study is that it only considers the case where there is one component failing in the circuit at a time. This was done to avoid the obvious complexity of multiple circuit failures, but future works can solve this issue. Already, there are methods [10] where one system can generates a list of possible faults instead of just one. Another system narrows down the list to only a handful of possible candidates. A similar system can be implemented with the method proposed in this study. This could be done by selecting multiple categories in the classification layer when they meet a certain threshold and choosing between them with another network.

CONCLUSION

In this paper, an effective method for predicting circuit failures in post-market circuit boards through simulation and deep learning is proposed and implemented. Normally, such failures are hard to diagnosis and difficult to fix. This test can be performed quickly to provide results that can lead to circuit fixes. We have shown that with a small amount of development time, real tests can be created for circuits that incorporated this method. The accuracy of the networks shown here support the idea that the time saved repairing circuit boards using the proposed method can far exceed the costs of misclassifications. With verification from real circuit boards in operation, we know that our data provides an accurate ground truth for the deep network to learn.

REFERENCES

[1] R. Desineni, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," in *International Test Conference*, Santa Clara, CA, 2006.

[2] Shangcong Feng, "Research on Fault Diagnosis of Mixed-signal Circuits Based on Genetic Algorithms," in *International Conference on Computer Science and Electronics Engineering*, Hangzhou, China, 2012.

[3] M. H. a. K. D. Jiri Vodrazka, "Aging Analysis of Metalized Film Capacitors," in *16th International Conference on Mechatronics- Mechatronika* , Brno, Czech Republic, 2014.

[4] R. Desineni, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," in *IEEE International Test Conference*, Santa Clara, CA, USA, 2006.

[5] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, pp. 1735–1780, 1997.

[6] J. Li, A. Mohamed, G. Zweig, and Y. Gong, “LSTM time and frequency recurrence for automatic speech recognition,” in Proc. ASRU, 2015, pp. 187–191.

[7] A. Vaswani, "Attention Is All You Need," in *Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

[8] H. C. J. S. H. G. A. Mejdoubi, "Remaining Useful Life Prognosis of Supercapacitors Under Temperature and Voltage Aging Conditions," *IEEE Trans. on Industrial Electronics,* pp. 4357-4367, May 2018.

[9] H. T. H. Y. K. Y. T. Hosokawa, "A Diagnostic Fault Simulation Method for a Single Universal Logical Fault Model," in *Proc. of IEEE 22nd Pacific Rim International Symposium on Dependable Computing*, Christchurch, New Zealand, 2017.

[10] X. L. R. B. Y. Xue, "Improving Diagnostic Resolution of Failing ICs Through Learning," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems,* vol. 37, no. 6, pp. 1288-1297, 2018.

[11] Joginipelly Arjun Kumar, "Efficient FPGA Architectures for Separable Filters and Logarithmic Multipliers and Automation of Fish Feature Extraction Using Gabor Filters," in *University of New Orleans Theses and Dissertations*, Louisiana, USA, 2014.

[12] J. J. Y. Moriya, "LSTM Language Model Adaptation with Images and Titles for Multimedia Automatic Speech Recognition," in *IEEE Spoken Language Technology Workshop*, Athens, Greece, 2018.