AgentDiag: An Agent-Assisted Diagnostic Framework for Board-Level Functional Failures*

Zelong Sun, Li Jiang, and Qiang Xu
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Zhaobo Zhang, Zhiyuan Wang and Xinli Gu
Huawei Technologies, Inc.
Santa Clara, CA

Abstract

Diagnosing functional failures in complicated electronic boards is a challenging task, wherein debug technicians try to identify defective components by analyzing some syndromes obtained from the application of diagnostic tests. The diagnosis effectiveness and efficiency rely heavily on the quality of the in-house developed diagnostic tests and the debug technicians' knowledge and experience, which, however, have no guarantees nowadays. To tackle this problem, we propose a novel agent-assisted diagnostic framework for board-level functional failures, namely AgentDiag*, which facilitates to evaluate the quality of the diagnostic tests and bridge the knowledge gap between the diagnostic programmers who write diagnostic tests and the debug technicians who conduct in-field diagnosis with a lightweight model of the boards and tests. Experimental results on a real industrial board and an OpenRISC design demonstrate the effectiveness of the proposed solution.*

1 Introduction

VLSI testing is a critical step for the semiconductor industry to identify defective chips. Unfortunately, with imperfect defect models and limited testing time, some bad chips inevitably pass manufacturing test and defects-per-million rates continue to increase with technology scaling. If such chips are integrated into a complex electrical system that consists of many printed circuit boards and each board contains tens of even hundreds of ICs, the product will either fail in system-level functional tests (if lucky) or fail at customer sites with much higher service cost.

Generally speaking, when a system fails, debug technicians run various diagnostic tests and try to determine the root cause of the failure based on the acquired test syndromes. This is an extremely challenging task due to the sophisticated interactions between on-board components, which requires deep understanding of the board system and the corresponding diagnostic tests for accurate diagnosis. The problem is even

*The work of Z. Sun, L. Jiang and Q. Xu was supported in part by a research grant from Huawei Technologies, Inc.

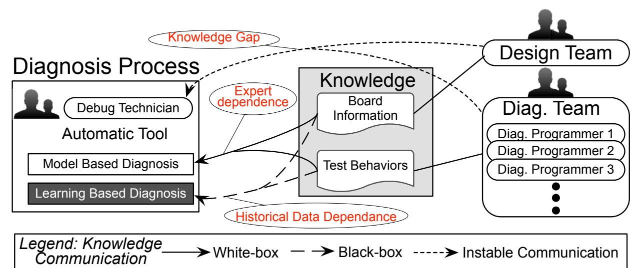


Figure 1. Knowledge Gap in Today's Diagnosis Process.

exacerbated by the fact that today's system design companies usually out-source the production of their systems to contract manufacturers (CMs, e.g., Foxconn and Jabil Circuit). Consequently, it is the CM's debug technicians, instead of the experts from system design companies who write the diagnostic tests, that are in charge of the actual diagnosis task. Due to the huge knowledge gap between diagnostic programmers and debug technicians, the diagnosis effectiveness and efficiency mainly relies on individual debug technician's knowledge and experience, and hence is not guaranteed. While direct communication between the two parties is possible and may be of help, its effectiveness is limited due to the the ambiguity of human language and possible personnel changes (see dotted arrows in Fig. 1).

To address the above problem, a number of automated diagnosis solutions have been presented in the literature over the past decades. On the one hand, many "white-box" diagnostic frameworks were constructed, wherein experts define rules [1–5] and/or build models [6–13] to facilitate diagnosis by abstracting their knowledge and understanding of the board system and the corresponding diagnostic tests. With the increasing complexity of the system, however, the difficulty to build a good "white-box" diagnostic framework rises up quickly. Moreover, with the accumulation of actual board failure data, it is essential to adjust diagnosis rules and/or tune diagnosis models to achieve better diagnosis accuracy. Such maintenance requires expert knowledge but is often unavailable due to the lack of resources. Motivated by the difficulty of "white-box" diagnostic framework construction and maintenance for

978-1-4799-0859-2/13/$31.00 ©2013 IEEE

today's complicated electrical systems, on the other hand, various "black-box" learning-based diagnosis solutions [14–19] were proposed recently. Such systems construct a reasoning engine for diagnosis, which makes use of machine learning and data analysis tools (e.g., support-vector machine (SVM) and artificial neural networks) to train a diagnosis system from the historical database. While relatively easy to construct, the diagnosis accuracy of such reasoning-based methods is usually not satisfactory, especially at the beginning of the product cycle with insufficient historical data.

In this paper, we propose an agent-assisted diagnosis framework for board-level functional failures, namely *AgentDiag*. The key idea of the proposed solution is to construct a simple yet effective *diagnosis agent* to bridge the knowledge gap between the diagnostic programmers who write diagnostic tests and the debug technicians who conduct actual failure diagnosis. The main contributions of this work include:

- By analyzing the board structure and diagnostic tests, *AgentDiag* introduces a novel evaluation metric namely *isolation capability* for every component, which is able to evaluate the quality of the diagnostic tests as a whole and facilitates to develop new diagnostic tests to enhance diagnosis accuracy;
- With the help of the proposed diagnosis agent, *AgentDiag* employs an adaptive diagnosis procedure that is able to dramatically improve diagnosis accuracy, especially for those reasoning-based diagnosis engines with insufficient historical data;

The remainder of this paper is organized as follows. Section 2 introduces existing diagnosis techniques for board-level functional failures and motivates this work. In Section 3, we provide an overview of the proposed *AgentDiag* framework. Section 4 and Section 5 detail the proposed evaluation metrics and the adaptive diagnosis procedure used in *AgentDiag*, respectively. Experimental results are then presented in Section 6. Finally, Section 7 concludes this work.

2 Related Works and Motivation

Rule-based diagnostic systems for board-level functional failures conduct diagnosis in a "IF-THEN" manner, by mapping test syndromes (e.g., Pass/Fail information of diagnostic tests and error messages from internal registers) to rules to be invoked [1, 2]. As can be imagined, such diagnostic system is not scalable for complicated boards as the number of rules is roughly exponential to the number of test syndromes.

Instead of defining explicit diagnosis rules, a model-based diagnostic system employs an approximate model to represent the board under diagnosis (BUD), and use it to identify defective component from the test observations. A formal causal model was proposed to improve the accuracy of bayesian inference based diagnosis in [7, 8], leveraging expert's causal knowledge about the system. Another hypotheses model was proposed in [20, 21] to infer the root cause given the manifestation of tests based on experts' structural knowledge of board and test. Taking the structural information of the board and causal knowledge of test behavior into the consideration, model-based diagnostic solutions are more scalable than rule-based ones and are likely to achieve higher diagnosis accuracy for complicated electrical systems. However, these models contains many hypotheses and pre-determined probabilistic parameters, which highly relies on the experts' knowledge. As the complexity of today's electrical systems is far beyond one's comprehension, the scalability of model-based diagnostic systems is also quite poor. What's worse, it is essential to update the parameters used in the model constantly with field diagnostic data to improve the system, but this may not be always possible since experts may leave the team or even the company.

To alleviate the difficulty of model construction and maintenance, machine learning and data analysis tools such as neural networks [15, 22], bayesian inference [14] and support-vector machine [23] are employed to construct a reasoning engine for diagnosis. By training the system with historical diagnosis data, it automatically infers the root cause of a new failed board given its test syndromes. The main limitation of reasoning-based diagnostic systems is the so-called *overfitting problem*, which occurs when the system has too many parameters relative to the number of observations. Generally speaking, a reasoning-based diagnostic system often uses the pass/fail information of all diagnostic tests as test syndromes for training, typically in the range of several hundred. To avoid overfitting, we should have thousands of successfully-diagnosed boards for training. Otherwise, the system begins to memorize training data rather than learning to generalize from trend [24, 25]. However, such a large database becomes available only at the later stage in the product cycle, which significantly constrains the effectiveness of reasoning-based diagnosis methods.

To sum up, existing "white-box" diagnostic systems try to pass expert knowledge to debug technicians as much as possible by constructing sophisticated diagnosis rules/models, but such strategy inevitably suffers from scalability issues with the increasing complexity of electrical systems; while the reasoning engines used in "black-box" diagnostic systems totally ignore knowledge about the BUD, thereby requiring large database to train the system, which is not available during product ramp-up stage. With the above, a natural question is whether we can develop an effective and efficient diagnostic system in between the two extreme solutions.

The above has motivated the proposed agent-assisted diagnostic system *AgentDiag*, wherein we build a light-weight model based on our knowledge about the BUD and its corresponding diagnostic tests. This model is then used to evaluate the quality of the diagnostic test set as a whole and guide reasoning-based diagnostic systems without requiring large amount of training data.

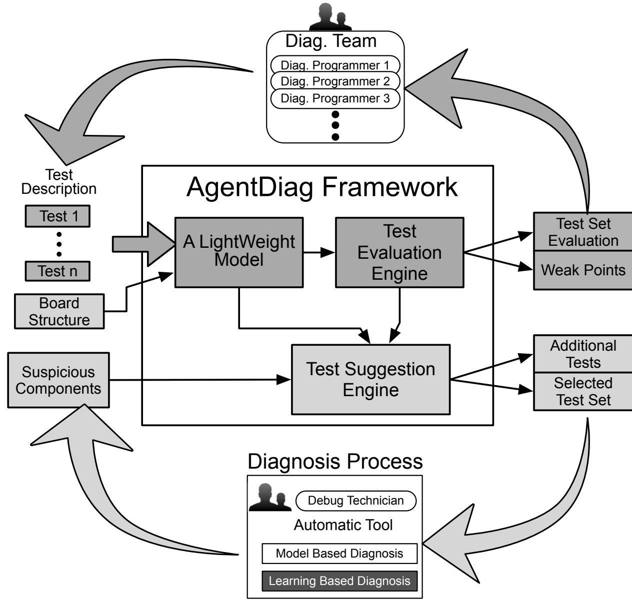


Figure 2. AgentDiag Diagnostic System

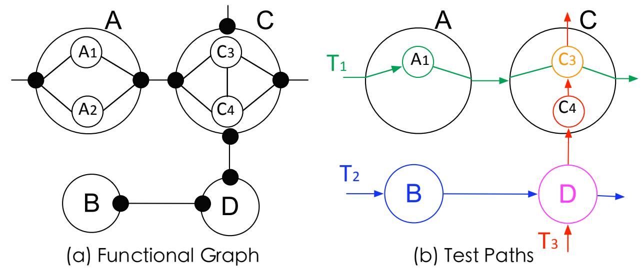


Figure 3. (a) Functional Graph Construction; (b) Test Path Construction.

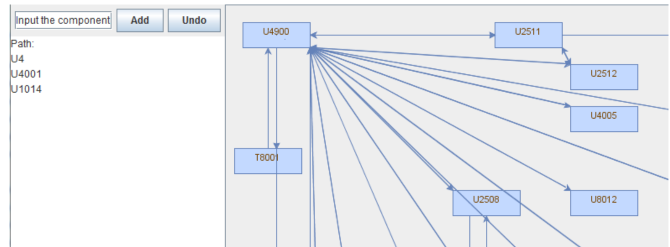


Figure 4. User Interface for *AgentDiag* Lightweight Model Construction.

3 AgentDiag Framework

In this section, we give an overview of the proposed *AgentDiag* solution for board-level functional failures and discuss the lightweight model used in *AgentDiag* in detail.

3.1 Overview

The proposed *AgentDiag* framework is shown in Fig. 2, which comprises three components: a *lightweight model*, a *test evaluation engine* and a *test suggestion engine*. *AgentDiag* sits in between diagnostic programmers who write diagnostic tests and debug technicians who conduct actual diagnosis for failed boards, and it bridges the knowledge gap between the two sides without necessarily constructing sophisticated diagnosis rules/models.

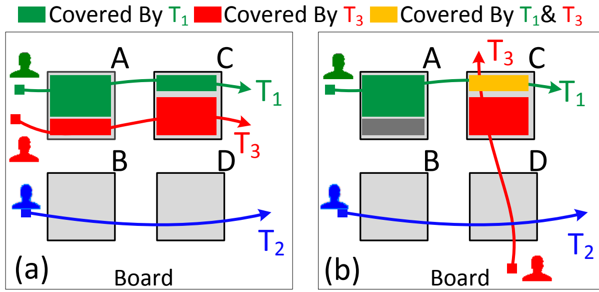
As can be seen in Fig. 2, there are two feedback loops with *AgentDiag* framework. At the top feedback loop, the *test evaluation engine* assesses the quality of the diagnostic test set as a whole and pinpoint those components that are difficult to be isolated during diagnosis (detailed in Section 4). The diagnostic programmers can then add/modify tests to improve diagnosis quality. At the bottom feedback loop, the *test suggestion engine* suggests test items to be applied by debug technicians and/or to be considered in a reasoning-based diagnostic system to improve diagnosis accuracy (detailed in Section 5).

Both the test evaluation engine and the test suggestion engine require the knowledge of the BUD and its corresponding diagnostic test sets, which is captured by the *lightweight model* in *AgentDiag*, as discussed in the following.

3.2 AgentDiag Lightweight Model

On the one hand, the lightweight model captures the structural information of the BUD, i.e., the components on the BUD and their interconnections, which can be automatically extracted from the board design specification. To be specific, we build a *functional graph* for the BUD structure. An example is shown in Fig. 3(a), in which each node represents a particular component while the interconnection between components is represented as an edge in the functional graph. Note that, the functional graph is constructed in a hierarchical fashion. That is, a node itself can contain a sub-level functional graph with structural knowledge at the sub-component level. In addition, the graph can represent connections between nodes in different hierarchical levels, because some components may be designed in-house with more detailed structural information while others may be third-party components without internal structural information.

On the other hand, the lightweight model requires the diagnostic programmers to input the behavior of diagnostic tests. In the simplest form, only the test path information (i.e., the components and the interconnects that test path go through) is needed (see Fig. 3(b)), which can be collected via a user-friendly interface (see Fig. 4). Note that, the test path information can also be input at different hierarchy level, e.g., component level and sub-component level. No doubt to say, fine-grained test path information leads to better diagnosis quality with *AgentDiag* when compared to coarse-grained test path information.



Component	A		C	
Faulty Block	A1	A2	C3	C4
Syndrome in (a)	(FP)	(PF)	(FP)	(PF)
Syndrome in (b)	(F***P***)	(P***P***)	(F***F***)	(P***F***)

P=Pass F=Fail e.g. (PF) = (T_1 pass T_3 fail)

(C)

Figure 5. Example of gap between different diag. programmers

From the above, we can see that, it is much easier to construct the lightweight model used in *AgentDiag* when compared to the construction of rules/models used in those "white-box" diagnostic systems, and it is applicable to complicated electrical systems. In addition, building the lightweight model is a one-time effort and it does not require frequent adjustment.

4 Test Evaluation Engine

A common limitation of existing diagnostic systems is that they "blindly" rely on the given diagnostic tests. Practically speaking, however, the development of diagnostic tests is often not truly diagnosis-oriented. For example, diagnostic programmers often reuse existing verification tests for diagnosis to reduce development cost, which, however, may cover many components with insufficient test syndromes. As a diagnostic system is essentially a "garbage in, garbage out" system, it is imperative to be able to evaluate the quality of diagnostic tests as a whole and suggest additional test items to improve diagnosis accuracy, if necessary.

Designing an effective set of diagnostic tests is not a trivial task. Let us take the example shown in Fig. 5 for illustration. For the simple BUD shown in this figure, it contains four components: A, B, C, and D. Suppose test T_1 passes through subcomponents A_1 and C_3 while test T_2 passes through components B and D. However, the test path of T_3 shown in Fig. 5(a) and the one shown in Fig. 5(b) are different. In Fig. 5(a), T_3 goes through subcomponents A_2 and C_4; while in Fig. 5(b), T_3 goes through subcomponents C_3 and C_4 as well as component D. For the sake of simplicity, let us assume the coverage of these tests on a particular component/subcomponent is 100% as long as they go through it. Intuitively speaking, T_3 in Fig. 5(a) is superior to the one in Fig. 5(b) as it covers subcomponent A_2 while the other one does not. However, the diagnosability of T_3 in Fig. 5(b) upon components A and C is in fact higher than that in Fig. 5(a). Let us consider the four cases that one of the subcomponents (A_1, A_2, C_3, and C_4) contains a fault. The associated test pass/fail syndromes for T_1 and T_3 are shown in 5(c). Clearly, we can differentiate the four cases with T_3 in Fig. 5(b) but not that in Fig. 5(a).

Motivated by the above, we introduce a novel evaluation metric to assess the diagnosability of the overall diagnostic tests, namely *isolation capability (ICap)*, as detailed in the following.

4.1 Isolation Capability

In order to achieve a high diagnosis accuracy, the application of diagnostic tests should strive to provide unique test syndromes to pinpoint the root cause component, i.e., with minimum ambiguity whenever possible. Generally speaking, however, it is usually inevitable that, under certain circumstances, failures in different components lead to the same test syndrome.

Therefore, in order to evaluate the diagnosability of a test set, for each component, we have the following definition:

- The *isolation capability (ICap)* of a test set *TS* for component c_i is defined as the probability that *TS* can *uniquely* locate the fault in c_i, under the condition that c_i is faulty.

Intuitively, when the test set produce a syndrome, the probability of uniquely locating the root cause depends on the number of suspicious components inferred from the test syndrome. In other words, the less suspicious components exist, the higher value for the *ICap*. To represent these suspicious components, we define:

- *Ambiguity Set(AS)*: The set of functional components that are suspicious to be faulty, based on the observed syndrome.

Given an observed test syndrome, based on the single faulty component assumption, we can derive the *AS* from the combination of paths of the test set. We assume that by applying test set on block c_i, the test syndrome λ has p tests *FAIL* and f tests *PASS* and we define them as pass test (PT) and fail test (FT) for block i. In an ideal case (i.e., assuming 100% coverage for each test), the functional fault is detected by all f *FAIL* tests. Meanwhile, the same fault cannot be detected by other p *PASS* tests. With the above assumption, there is only such one possible syndrome. The *AS* can be calculated as follow:

$$AS(\lambda) = \bigcap_{i=1}^{f} Path(FT_i) - \bigcup_{j=1}^{p} Path(PT_j) \quad (1)$$

where *Path* indicates the components in the test's path. Let us consider the example shown in Fig. 5(a). If A has an active

fault, T_1 and T_3 will be *FAIL* while T_2 is *PASS*. The $AS(\lambda)$ of component A contains A,C. However, in Fig. 5(b), the $AS(\lambda)$ of A has only A.

After we obtain the $AS(\lambda)$, the *ICap* of c_i should be one over the size of the $AS(\lambda)$ if all the components in the AS are equally to be faulty. In fact, however, some of them are more prone to defects than others. We introduce a parameter $ErrorRate$ to take this issue into account, which is defined as:

- $ErrorRate(ER)$: The probability of a functional component to be the root cause under the condition that an error occurs on the board.

Generally speaking, the defect rate of a component can be assumed to be proportional to its *Defect Part Per Million(DPPM)* obtained from its provider. Consequently, the *ER* of each component can be initialized as the percentage of its own *DPPM* over the summation of all the components' *DPPM* on the board. This parameter can be tuned with field diagnosis data after accumulating a large number of failure boards. With the above, the *ICap* of component c_i can be calculated by the following equation:

$$ICap(AS(\lambda),c_i) = ER(c_i)/\sum_{l=1}^{m} ER(c_l) \quad (2)$$

where m denotes the number of components in AS. Let us continue to discuss the above example, if the $ER(A)$ to $ER(D)$ in Fig. 5 is 40%, 30%, 20% and 10%, respectively. The $ICap(AS(\lambda),A)$ in Fig. 5(a) is 66.7%, while in Fig. 5(b), it is 100%.

In practice, the coverage of a test on a particular component that it goes through is not 100%. That is, when a fault within a component is activated, the tests passing through this component may not fail. To take the test coverage factor into account, Equation (2) is extended as following:

$$ICap(AS(\lambda),c_i) = \frac{ER(c_i) \times \bigcap_{j=1}^{f} TC(FT_j,c_i)}{\sum_{l=1}^{m} ER(c_l) \times \bigcap_{j=1}^{f} TC(FT_j,c_l)} \quad (3)$$

where FT_j is the failed test in the test syndrome λ and $TC(FT_j,c_i)$ is the ratio of functional faults within c_i that are covered by FT_j. Similarly, we further extend Equation (3) by eliminating the assumption of perfect coverage of p passed tests as following:

$$ICap(AS(\lambda),c_i) = \frac{ER(c_i) \times \bigcap_{j=1}^{f} TC(FT_j,c_i) \times \bigcap_{k=1}^{p} \overline{TC(PT_k,c_i)}}{\sum_{l=1}^{m} ER(c_l) \times \bigcap_{j=1}^{f} TC(FT_j,c_l) \times \bigcap_{k=1}^{p} \overline{TC(PT_k,c_l)}} \quad (4)$$

At last, we enumerate all the possible test syndromes λ when the component c_i is faulty and extend equation (4) as following:

$$ICap(TS,c_i) = \sum_{\lambda=1}^{\mu} ICap(AS(\lambda),c_i) \quad (5)$$

where μ is the total number of possible syndromes. Let us continue to consider the example in Fig. 5(b) and demonstrate the calculation of *ICap* of component C. Obviously, two possible test syndromes of T_1T_3 can be seen in Fig. 5(c). However, due to the imperfect test coverage, we need to consider two more scenarios: 1) if a fault in C_3 is covered by T_1 but escapes T_3, then we have another syndrome for T_1T_3, i.e., FP; 2) if the fault occurs outside of the two subcomponents C_3 and C_4, then the test syndrome becomes PP. Now, we have four possible syndromes, and according to equation (5), *ICap(AS(*T_1T_3*=*FF,FP,PF,PP*),C)* can be calculated using equation (4) with the coverage information of T_1 and T_3 upon components in AS, which is derived using equation (1).

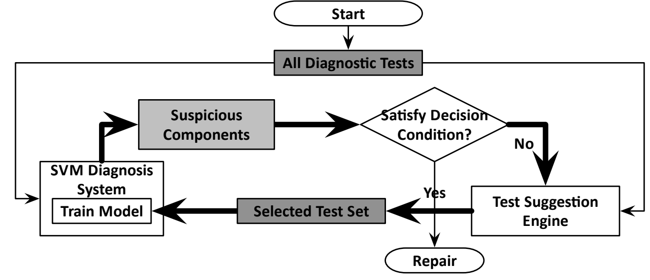


Figure 6. The Integration of Test Suggestion Engine with an SVM-based Diagnostic System.

5 Test Suggestion Engine

In this section, we demonstrate how the proposed *test suggestion engine* in *AgentDiag* works by integrating it into an SVM-based diagnostic system [23].

A typical SVM-based diagnostic system for board-level functional failures [23] works as follows. Given the historical successfully-diagnosed boards, the system is trained with the fault syndromes extracted from the log files as inputs while the replaced components as outputs. As the number of fault candidates (classes) is in the range of tens or hundreds in board-level failure diagnosis, multi-class SVMs are designed in the one-against-all manner in [23], i.e., if there are n candidates, n SVMs are designed and each SVM represents one component. All the training cases are used to determine the optimal separating hyperplane (OSH) of each SVM. The system ranks suspicious components (noted as *diag. score*) according to the decision function of the OSH. One widely-adopted syndrome is the pass/fail information of all tests formatted as a binary vector (Entry 1 means the test fails, otherwise the entry is 0), whose length is usually in the range of several hundred. Consequently, it inevitably suffers from the overfitting problem at the ramp-up stage of the product. While there exist a number of feature reduction techniques in the literature (e.g., [24, 25]), their effectiveness is questionable because these methods rely on statistical analysis of the training cases.

With the lightweight model in *AgentDiag*, the proposed *test suggestion engine* is able to adaptively select those effective tests (i.e., most relevant features) without conducting statis-

tical analysis for the training cases, thereby effectively solving the overfitting problem. The adaptive diagnosis procedure works as shown in Fig. 6. For a new failed board, we first employ the SVM-based diagnostic system in [23] for root cause analysis using the syndrome of the entire test set pass/fail information. Next, instead of ranking them as the output of the diagnostic system, we consider the list of suspicious components whose *diag. score* is positive for another round of diagnosis. In this iteration, we rely on the test evaluation engine discussed earlier to select those tests that contribute to the isolation capability of the suspicious components (the other components are regarded as fault-free) and use them only to train the SVM-based diagnostic system again. With the new multi-class SVMs with reduced features (fewer test pass/fail information are considered), we conduct diagnosis for this board again. The above procedure iterates until the *diag. score* of the 1st-ranked component exceeds a certain threshold.

It should be noted that the diagnostic tests are only run once, which can take up to several hours. Only the SVM models need to be built for multiple times in an adaptive fashion with the proposed diagnosis procedure, which only requires a few seconds for the training of hundreds of historical cases to complete and hence the runtime overhead can be safely ignored.

The proposed *test suggestion engine* can be also used by debug technicians for manual diagnosis in an interactive manner. That is, for a particular board functional failure, after conducting a few diagnostic tests, the corresponding debug technician usually has some suspicious components in mind based his/her knowledge and past experience. The test suggestion engine can then suggest those tests that are able to quickly identify root cause among these components by conducting isolation capability analysis with the help of the lightweight model and the test evaluation engine in *AgetnDiag*.

6 Experimental Results

In this section, we first verify the effectiveness of the *test evaluation engine* in *AgentDiag* based on an open source design with a K-NN based diagnostic system. Next, we demonstrate the effectiveness of the *test suggestion engine* in *AgentDiag* with field diagnosis data on a real industrial board with SVM diagnostic system.

6.1 Experimental Results on an OpenRISC SoC

6.1.1 Experimental Setup

OR1200 is a 32-bit microprocessor with five-stage pipeline [26]. 16 verification tests are provided along with the design. In this experiment, we treat OR1200 as a hypothetical board under diagnosis, in which the 14 sub-modules (including PC generator, ALU, Register File, etc.) are treated as components that may fail. Pin-level fault injection is performed with the Verilog Procedural Interface(VPI) [27]. Registers from the six memory units (Data Cache, Instruction Cache, IMMU, DMMU, GPRs and SPRs) are regarded as the test observation points, which are compared against golden values to determine whether a test passes or fails. The coverage of a test for a component is calculated as the number of pins, whose fault injection causes this test to fail, over the total number of pins in the fault injection experiment. *ER* is initialized to be proportional to the number of pins in a module.

We implement a simple yet widely-adopted learning-based diagnostic system based on K-nearest neighbor algorithm[1] (k-NN, $k = 1$ in our experiment) for OR1200, and we study the correlation between the diagnosis successful rate (SR) for a given diagnostic test set against the ICap value obtained with our test evaluation engine in *AgentDiag* to prove its effectiveness.

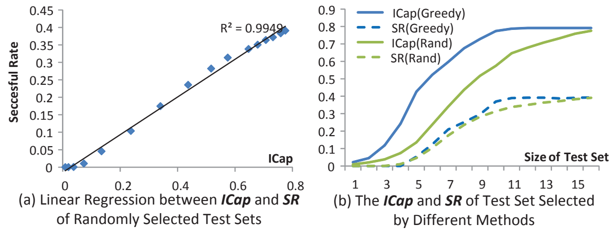
6.1.2 Experimental Results

In our first experiment, we make linear regression between the isolation capability *ICap* and the diagnosis successful rate *SR*. Figure 7(a) presents our experimental results. Faults are randomly injected for $100,000$ times and *SR* is the number of faults that are correctly diagnosed over the total number of injected faults. We calculate the coefficient of determination R^2 and it is as high as 0.9949, which clearly demonstrates the strong correlation between the two items and proves the effectiveness of *ICap* for capturing the diagnosability of the test sets.

Next, we compare the *ICap* and *SR* values between a randomly-ordered test set and the same test set ordered greedily with increasing *ICap* values. The greedy algorithm is performed as follows: in each step, we add a test that contributes the largest increment for the summation of *ICap* for all components. As shown in Fig. 7(b), the trend of the curves are quite consistent. We can observe that, with a small number of tests (when the number of tests is less than 4), both *ICap* and *SR* increase slowly. This is because, it is quite difficult to differentiate test syndromes with less test items. *ICap* and *SR* increases dramatically after we are given more tests, due to the fact that more unique test syndromes can be obtained. We can also observe that, while we are given 16 tests, in fact 11 tests give almost the same diagnosis successful rate, which is identified with the greedy algorithm. At the same time, we can observe the *ICap* value for the 11 tests is also the same as that of all 16 tests, which, again, demonstrates the effectiveness of the proposed evaluation metric for diagnosability.

Finally, our test evaluation engine outputs the *ICap* of module *floating point unit (FPU)* to be close to zero with the given 16 tests, which is also confirmed with fault injection results with *SR* close to zero. A closer examination of the given test

[1] In a k-NN diagnostic system, a new case is compared to all training cases to obtain the top k nearest ones based on the distance calculation.



(a) Linear Regression between ***ICap*** and ***SR*** of Randomly Selected Test Sets

(b) The ***ICap*** and ***SR*** of Test Set Selected by Different Methods

Figure 7. The Experimental Results of Open-Risc 1200 SoC

"float" for FPU shows that it cannot propagate faults to the observation points. We therefore add a new test that performs a large amount of floating point operations and tries to propagate its results out. After doing so, *ICap* for *FPU* reaches '1.0' and the *SR* for *FPU* is also increased to 100% in our experiments.

6.2 Experimental Results on An Industrial Board

6.2.1 Experimental Setup

We evaluate the effectiveness of the test suggestion engine by conducting experiments on an industrial board in volume production. There are around 150 components on this board and the diagnostic test set contains 508 tests. We generate the lightweight model based on the board specification and test path information, and we initialize the *ER* of each component to be equal to each other.

In total there are 286 successfully-repaired cases available when conducting this experiment. We randomly choose a subset of these cases as training cases (varied from 100 to 250) and validate the diagnostic system using the rest. Since using which cases for training affect the diagnosis results, we have conducted random training case selection for 100 times and report the average results. We compare the overall diagnosis successful rate within 1, 2 and 3 attempts between the original SVM-based diagnostic system [23] and the proposed adaptive one as discussed in Section 5.

6.2.2 Experimental Results

The comparison results are shown in Table 1, in which *SR* denotes the diagnosis successful rate with the proposed adaptive diagnosis method while SR' denotes that of the original SVM based method in [23], and the subscript i indicates the diagnosis successful rate within i attempts. As can be observed from the table, the proposed solution significantly increases diagnosis accuracy. The first hit rate (i.e., successful rate for the 1st attempt) is about 8% higher on average. This is expected because the proposed adaptive diagnosis procedure effectively address the overfitting problem with the original SVM solution, by eliminating those irrelevant features when training the system in later iterations. When considering the second and the third attempts, however, the improvement in terms of the overall diagnosis successful rate drops to around 5% and 2% on average, respectively. This is because, the proposed adaptive diagnosis procedure is still constrained by the candidate list output from [23] in the 1st iteration, which takes the entire test set as parameters to train the diagnostic system. Nevertheless, as we are more concerned about the first hit rate during diagnosis, the advantage of the proposed solution is obvious.

In terms of diagnosis efficiency, as can be seen from Table 1, the average number of iterations with the proposed method is around 3, and there is a slight increase of iteration count with the increase of training cases. The runtime of the proposed diagnosis procedure is around 5 to 6 seconds, while [23] takes less than 2 seconds. When comparing to the time spent on applying diagnostic tests, it can be safely ignored.

7 Conclusion

In this paper, we propose a novel agent-assisted diagnostic framework for board-level functional failures, namely *AgentDiag*. With the help of the lightweight model, *AgentDiag* is able to evaluate the diagnosability of the test set with the proposed test evaluation engine and improve diagnosis accuracy with an adaptive test suggestion engine. Experimental results on an industrial board demonstrate the effectiveness of the proposed solution.

References

[1] D. Allred, Y. Lichtenstein, C. Preist, M. Bennett, and A. Gupta. Agatha: An integrated expert system to test and diagnose complex personal computer boards. In *Proc. Innovative Applicat. Artif. Intell*, page 1991, 1991.

[2] N. Dev and B. Anderson. Pimtool, an expert system to troubleshoot computer hardware failures. In *Proceedings of the National Conference on Artificial Intelligence*, pages 853–860. John Wiley & Sons Ltd., 1997.

[3] G.W. Leng and L.K. Teen. Espcrm–an expert system for personal computer repair and maintenance. *Engineering Applications of Artificial Intelligence*, 5(2):121–133, 1992.

[4] J.G. Rowland and L.C. Jain. Knowledge-based systems for instrumentation diagnosis, system configuration and circuit and system design. *Engineering Applications of Artificial Intelligence*, 6(5):437–446, 1993.

[5] T. Satyanarayana, G. Subramanyam, and K.V. Rama Rao. Implementing an expert system for fault diagnosis of electronic equipment. *Engineering Applications of Artificial Intelligence*, 8(3):355–364, 1995.

Training #	SR_1	SR_1'	SR_2	SR_2'	SR_3	SR_3'	Avg.Iter#	$Avg.Iter'$#	Avg.Time	$Avg.Time'$
100	39.94%	30.65%	41.51%	35.48%	43.90%	41.40%	2.84	1	6.53	1.57
150	46.41%	40.44%	51.31%	47.79%	53.77%	50.00%	2.91	1	6.29	1.56
200	53.54%	45.35%	58.34%	54.65%	60.47%	58.14%	3.12	1	5.05	1.64
250	57.00%	49.88%	63.53%	58.33%	65.82%	63.89%	3.56	1	5.92	1.63

Table 1. Diagnosis Results for an Industrial Board with Varied Training Cases

[6] Z. Zhang, Z. Wang, X. Gu, and K. Chakrabarty. Board-level fault diagnosis using an error-flow dictionary. In *Test Conference (ITC), 2010 IEEE International*, pages 1–10. IEEE, 2010.

[7] Y. Peng and J.A. Reggia. A probabilistic causal model for diagnostic problem solving part i: Integrating symbolic causal inference with numeric probabilistic inference. *Systems, Man and Cybernetics, IEEE Transactions on*, 17(2):146–162, 1987.

[8] Y. Peng and J.A. Reggia. A probabilistic causal model for diagnostic problem solving part ii: Diagnostic strategy. *Systems, Man and Cybernetics, IEEE Transactions on*, 17(3):395–406, 1987.

[9] J.W. Sheppard and W.R. Simpson. Inducing diagnostic inference models from case data. *Research Perspectives and Case Studies in Systems Test and Diagnosis*, 13, 1998.

[10] J.W. Sheppard and W.R. Simpson. Managing conflict in system diagnosis. *Computer*, 31(3):69–76, 1998.

[11] W.R. Simpson and J.W. Sheppard. Encapsulation and diagnosis with fault dictionaries. In *AUTOTESTCON'96,'Test Technology and Commercialization'. Conference Record*, pages 441–446. IEEE, 1996.

[12] B. Chess and T. Larrabee. Creating small fault dictionaries [logic circuit fault diagnosis]. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(3):346–356, 1999.

[13] D. Mittelstadt, R. Paasch and B. DAmbrosiso Application of a bayesian network to integrated circuit tester diagnosis. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 9(01):51–65, 1995.

[14] Z. Zhang, Z. Wang, X. Gu, and K. Chakrabarty. Board-level fault diagnosis using bayesian inference. In *VLSI Test Symposium (VTS), 2010 28th*, pages 244–249. IEEE, 2010.

[15] Z. Zhang and S. Ozev. Parametric fault diagnosis for analog circuits based on neural networks. In *Proc. North Atlantic Test Workshop*, pages 1–6, 2008.

[16] A. Balakrishnan and T. Semmelbauer. Circuit diagnosis support system for electronics assembly operations. *Decision support systems*, 25(4):251–269, 1999.

[17] P. Cunningham, B. Smyth, and A. Bonzano. An incremental retrieval mechanism for case-based electronic fault diagnosis. *Knowledge-Based Systems*, 11(3-4):239–248, 1998.

[18] O. Jakubowicz and S. Ramanujam. A neural network model for fault diagnosis of digital circuits. In *Proceedings of the 1st IEEE International Conference on Neural Networks*, volume 2, page 611, 1990.

[19] A.A. Al-Jumah and T. Arslan. Artificial neural network based multiple fault diagnosis in digital circuits. In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, volume 2, pages 304–307. IEEE, 1998.

[20] D. Manley and B. Eklow. A model based automated debug process. In *IEEE Board Test Workshop*, pages 1–7, 2002.

[21] C. O'Farrill, M. Moakil-Chbany, and B. Eklow. Optimized reasoning-based diagnosis for non-random, board-level, production defects. In *Test Conference, 2005. Proceedings. IEEE International*, pages 7–pp. IEEE, 2005.

[22] Z. Zhang, K. Chakrabarty, Z. Wang, Z. Wang, and X. Gu. Smart diagnosis: Efficient board-level diagnosis and repair using artificial neural networks. In *Test Conference (ITC), 2011 IEEE International*, pages 1–9. IEEE, 2011.

[23] Z. Zhang, X. Gu, Y. Xie, Z. Wang, Z. Wang, and K. Chakrabarty. Diagnostic system based on support-vector machines for board-level functional diagnosis. In *IEEE European Test Symposium*, 2012.

[24] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.

[25] G. Brown, A. Pocock, M.J. Zhao, and M. Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13:27–66, 2012.

[26] http://www.opencore.org.

[27] Charles Dawson, Sathyam K Pattanam, and David Roberts. The verilog procedural interface for the verilog hardware description language. In *Verilog HDL Conference, 1996. Proceedings., 1996 IEEE International*, pages 17–23. IEEE, 1996.