Proceedings of the International Test Conference, October 2003

Optimization of Test/Diagnosis/Rework Location(s) and Characteristics in Electronic Systems Assembly Using Real-Coded Genetic Algorithms

Zhen Shi and Peter Sandborn

CALCE Electronic Products and Systems Center
University of Maryland
College Park, MD 20742

Abstract

This paper presents a framework for optimizing the location(s) and characteristics (fault coverage/test cost, rework success rate/rework cost) of Test/Diagnosis/Rework (TDR) operations in the assembly process for electronic systems. A new search algorithm called Waiting Sequence Search (WSS) is applied to traverse a general process flow in order to perform the cumulative calculation of a yielded cost objective function. Real-Coded Genetic Algorithms (RCGAs) are used to perform a multi-variable optimization that minimizes yielded cost. Several simple cases are analyzed for validation and a general complex process flow is used to demonstrate the applicability of the algorithm.

1. Introduction

Performing trade-offs of where in an assembly process to test and what level of test, diagnosis and rework to perform are key to optimizing the cost and yield of an electronic system's assembly. The level of testing, diagnosis and rework performed is defined by the following *feature parameter* combinations: fault coverage/test cost, diagnosis success rate/diagnosis cost, and rework success rate/rework cost.

Previous research efforts have treated the economics of test for electronic systems. Test economics modeling and methodology development ranges from classical relations between fault coverage and yield [1], to the development of economic models for analyzing the financial aspects of test investments [2] and the tradeoffs associated with design for test [3]. Many modeling efforts have focused on the development of test step models for inclusion in process flow[1] based cost analyses, e.g., [4]-[8]. These models vary in complexity, but in general they take incoming costs and yields (from upstream process steps) and determine the cost and yield of the product after testing to specified fault coverage has been performed. In most cases, some type of diagnosis and rework can also be modeled. Persons involved with modeling the economics of a product's manufacture can use these models within the larger manufacturing process flows.

Previous efforts using Test/Diagnosis/ Rework (TDR) models have been confined to evaluating the impacts on product manufacturing when the locations of the test operations (relative to the manufacturing process steps) are manually chosen. Optimization of the feature parameter combinations of testing concurrent with a search to determine the optimum location(s) for TDR operations in a process is not known to have been addressed in previous work. The target of this paper is to address the following questions:

- How much fault coverage should be required?
- Where (in a process) do I put the test locations?
- When do I need to do the rework?
- How many rework attempts should be made?

The trade-off process has the purpose of making the best choice between various possible versions of tests and reworks, and to suggest the values of feature parameters of testing in order to minimize an objective function associated with the process flow. To achieve the goal, the following three items have been investigated:

1. Computations of a single TDR operation.
2. A searching algorithm to traverse all process steps in a general process flow.
3. Multi-variable optimization using Real-Coded Genetic Algorithms (RCGAs).

[1] A process flow is a set of process steps in an application-specific sequence.

2. The Optimization Method

The framework for optimizing the implementation of TDRs in a process flow is shown in Figure 1. In the model, the process flow for manufacture or assembly of the electronic system is first generated. Then candidate TDR operations are inserted into the process flow in all possible locations, i.e., between all process steps as an initial guess (if the TDR operations can be inserted according to specific experiences the optimization will be more efficient). Starting with the process flow and the initial guess of TDR operation locations, a pre-analysis is executed to sort similar process steps and merge branches to decrease complexity of process flow. A new search algorithm (see Section 4) is applied to traverse the entire process to perform the cumulative calculation of the objective function (yielded cost). RCGAs are used to perform a multi-variable optimization that minimizes yielded cost.

The remainder of the paper is organized as follows: Section 3 reviews the cost model used for the computations of a single TDR operation in the process flow. Section 4 proposes a new graph-based search method, named "Waiting Sequential Search", which is applied to general process flows in order to search all the step nodes in the graphical representation and derive cumulative objective function. In Section 5, RCGAs are used to implement the optimization of the TDR operations and the parameters associated with them in the process flow. Test cases are also provided in Section 5 to verify and demonstrate the applicability of the model.

3. The Test/Diagnosis/Rework Model

To involve all the effects of the feature parameters on a general process flow and how they relate to each other, a comprehensive model was developed by Trichy *et al.* [7]. This model provides a detailed formulation of the feature parameters for a single TDR operation. Figure 2 shows the content of the model and detailed formulations are given in [7][2].

This model provides the accurate computation of the feature parameters: C_{out} and Y_{out}, etc. for a single TDR operation in a process flow in which

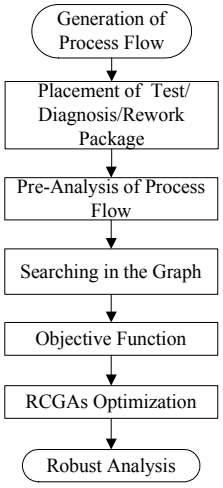


Figure 1: The framework for optimization of TDR operation location(s) and characteristics in a general process flow.

several variables—C_{test} and C_{rew} are defined as the constants. For real processes C_{test} is not a constant and instead is related to the fault coverage of the test. The rework yield (rework success rate) is also not a constant. It depends on the specific rework actions taken. In practice the rework operation may cause additional defects to be inserted in the product, [9]. For use in this work, the model in [7] is extended by defining general forms of the relationships among the feature parameters, e.g., the costs of test and rework in terms of fault coverage and rework yield respectively.

A relationship between the cost of test (proportional to test time or number of tests) and fault coverage has been suggested by Goel [10]. Empirical data shows that the test process can be divided in two phases. A relatively small subset of T_I (number of tests in Phase I see Figure 3) of the total set of tests provides a fault coverage ranging from 65% to 85% for most combinational logic circuits [10]. For Phase II of the test generation, the number of additional tests required is approximately a linear function of the number of untested faults remaining at the end of Phase I. Unlike Phase I, in Phase II each generated test tends to detect fewer faults than the one before it and the average cost per detected fault increases.

[2] Note, the following typographical errors should be corrected in [7]: In (2) and (3), the maximum of the summation should be n-1 instead of n, and (4a) can be used for either definition of f_p with $N_{d_{n+1}}$ changed to N_{dl_n}. In (13), the subscript of N_r should be i-1 instead of i when i>0.

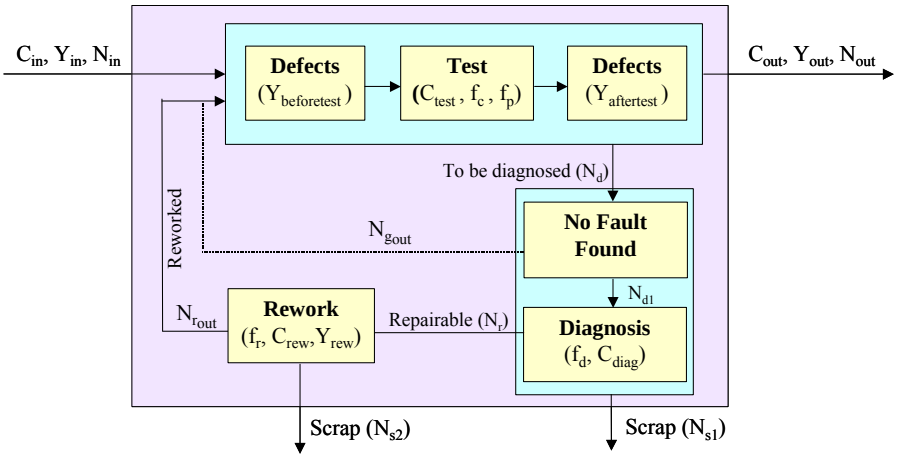


Figure 2: Organization of the Test/Diagnosis/Rework (TDR) operation, [7].

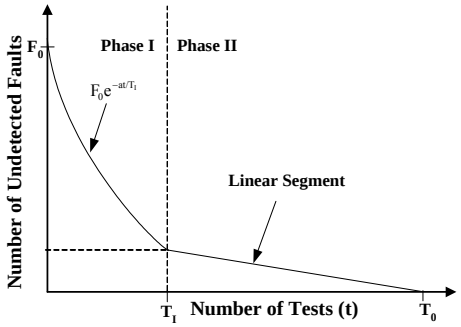


Figure 3: Typical curve of untested faults versus the number of tests, [10].

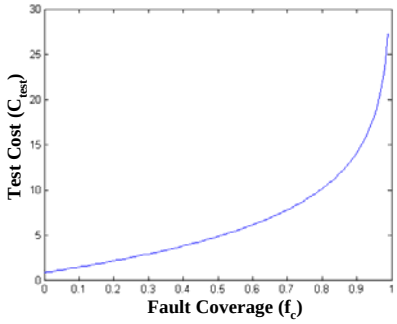


Figure 4: Example relationship between the test cost and fault coverage.

For the purpose of simplifying the relationship, an exponential function can be used to approximately simulate Phase I and a linear function in Phase II on the assumption that Phase II could not reach a full coverage (100% of fault coverage) in practical testing. Assuming that the test cost is proportional to the number of tests, a relationship between test cost and fault coverage can be derived,

$$C_{test} = p_t[b_t \ln(1-f_c) - r_t] + C_{ft}, f_c \in (0,1) \quad (1)$$

where p_t is the cost coefficient; b_t is the coefficient of test characteristic, r_t is the fault ratio, f_c is the fault coverage of test, C_{ft} is the fixed cost of test[3]. Figure 4 shows a plot of (1) using the values in Table 1.

Table 1: Example values of factors in (1) and (2).

p_t	b_t	r_t	C_{ft}	p_r	b_r	r_r	C_{fr}
0.02	-288	8.2	1	0.02	-300	10	1

[3] C_{ft} accounts for fixed costs associated with testing, i.e., there is a minimum fixed cost for having even a small fault coverage.

There is similar relationship between rework yield (y_r) and rework cost (C_{rew}) for the first rework attempt,

$$C_{rew} = p_r[b_r \ln(1-y_r) - r_r] + C_{fr}, y_r \in (0,1) \quad (2)$$

An alternative model for calculating the rework cost has been proposed in [8], which relates C_{rew} with the cost of rework equipment and rework time.

Functional relationships between fault coverage and test cost, and rework yield and rework cost obviously depend on the type of system being considered. The relationships in (1) and (2) were used for the remaining work in this paper as examples only. The methodology that is the subject of this paper, will work successfully with alternative models.

After the single TDR operation is resolved, the next issue is to cumulatively compute the objective function of feature parameters of a general process flow that includes multiple TDR operations. For the purpose of derivation of the objective function, search algorithms are needed to traverse the process flow with the computation performed according to the sequence among process steps.

4. Process Flow Search Algorithm

Based on the analysis of the TDR model in Section 3, we know how to compute the feature parameters of a single test step. A general process flow may, however, have many different (possibly independent) TDR activities located within it. The next issue to be addressed is how to obtain the objective function (yielded cost, i.e., cost divided by yield) for an entire general process flow. Graphs are useful in representing the process flow, i.e., complex systems involving binary relationships among process steps [11].

4.1 Graphical representation of a process flow

First, we review the basic graph notations that are used in this paper. A graph G consists of a set of nodes V and a set of edges E, $G =< V, E >$. Here G denotes the entire process flow, V denotes the process steps and an edge $< X, Y > \in E$ denotes the directed flow between two adjacent process steps. The degree of a node is the number of the neighbors adjacent to it. We write $X \rightarrow Y$ when $< X, Y > \in E$ is in a directed graph (DIGRAPH) [11,12]. We define $PRED(X)$ as the set of all predecessors (process steps preceding X) of node X, and $SUCC(X)$ as the set of successors (process steps after X) of X. if $X \rightarrow Y$, then $X \in PRED(Y)$ and $Y \in SUCC(X)$. The indegree $id(v)$ of a node X is the cardinality of $PRED(X)$ and the outdegree $od(v)$ of a node X is the cardinality of $SUCC(X)$. The graphical representation of an example complex process flow is shown in Figure 5.

4.2 Definitions of types of vertices (process steps)

From the analysis of a generic process flow, four types of basic steps have been identified. The process steps can be defined in the following ways:

- Start Step, $od(v) = 1$ and $id(v) = 0$,

There are no inputs to the Start Step and just one output from it. There may be multiple Start Steps in a complex process flow.

- Sequence Step, $od(v) = 1$ and $id(v) = 1$

There is one input and one output for a sequence step. Sequence Steps are the most common type of step in process flows for electronic systems.

- Cross Step, $od(v) = 1$ and $id(v) \geq 1$

There are multiple inputs and just one output associated with this kind of step. The complexity of the problem is significantly increased by each Cross Step in the process flow.

- End Step, $od(v) = 0$ and $id(v) \geq 1$

An End Step represents the end of the process flow, which merges all the branches to one. The objective function of the process flow is derived from an End Step. There may only be one End Step in a process flow.

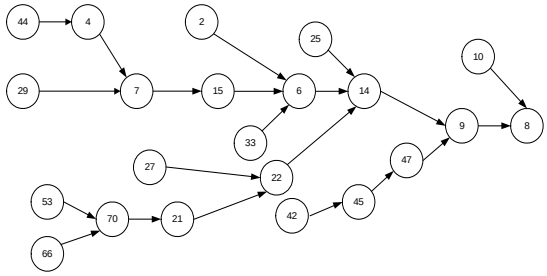


Figure 5: Example graphical representation of a complex process flow.

4.3 Waiting Sequential Search (WSS)

To derive the objective function of a process flow, every process step needs to be searched in order to perform the cumulative computation. Significant previous work on graph-based search algorithms exists, e.g., [13,14]. Several algorithms have been applied to resolve search problems similar to the one posed here, [11,12]. As to the graph-based representation of our process flows, there are several specific features that make it attractive to propose a new search algorithm to perform an efficient search in the process flow.

From the Figure 5, the following features of the general complex process flow can be observed:

- The $od(v) \leq 1$ is always true for all the vertices (process steps) in the process flow;
- There are sequential search requests for the graph, i.e., only when all the predecessors $PRED(Y)$ of $v(Y)$ have been visited, then $v(Y)$ could be visited.

WSS begins from the lowest-numbered vertex that belongs to a Start Step then proceeds to search the next step. By checking the type of the successor, the algorithm decides to continue to search to the next Sequence Step or wait at a Cross or End steps. After moving to the next step, the corresponding computation of feature parameters of the previous step is performed and the outcome is stored in a data table in which all the property information associated with process steps is recorded. If Cross or End types of steps

are encountered, the visitation status of all predecessors will be checked. If all the predecessors have been checked, the searching continues, if not, the search begins from another Start Step type of vertex until the last Start Step vertex is visited. The algorithm requires that the searching of the next step continue only after all the branches of the present step have been visited. There are waiting actions for the sequential search based on the characteristics of the process flow. The searching process for the complex process flow example shown in Figure 5 using WSS is described below:

1. First, begin from the lowest number of Start Steps (2), (2) → (6), check whether all the other branches have been searched, compute then wait;
2. (10) → (8), check the other branches and wait;
3. (25) → (14), wait;
4. (27) → (22), wait;
5. (29) → (7), wait;
6. (33) → (6), wait;
7. (42) → (45) → (47) → (9), wait;
8. (44) → (4) → (7) → (15) → (6) → (14), wait;
9. (53) → (70), wait;
10. (66) → (70) → (21) → (22) → (14) → (9) → (8), (8) is an End Step, all the branches have met and the process ends.

4.4 Multi-variable optimization function

The objective of optimizing TDR location(s) and characteristics is to minimize the yielded cost of the entire process flow [15]. The objective function in which feature parameters of all possible TDR operations are considered can be derived from sequential cumulative computation from the Start Steps to the End Step. When the WSS algorithm traverses the entire process flow, a cumulative function is computed to be used as the objective function that needs to be minimized in the optimization. The optimization problem becomes:

$$\min_{x \in X} \sum_{i=1}^{n} C_Y(x_{1_i}, x_{2_i}, \cdots x_{m_i}) \quad (3)$$

where,

- m = number of feature parameters to be optimized;
- n = total number of process steps with all possible TDR operations included;
- C_Y = yielded cost of the process flow, cumulative cost divided by final yield.

In the optimization, first the TDR operations would be placed in all possible locations or be chosen according to expert suggestions. The fault coverage (f_{c_i}), rework yield (Y_{r_i}) and rework attempts for the i^{th} TDR operation need to be optimized in order to minimize the total yielded cost of the process flow. For example, for the complex process flow in Figure 5, there are 22 (including the one location after the End Step-8) possible TDR operation locations following each of the process steps if there were no specific locations constraints as to where a TDR operation could not be placed provided by the user. The objective function can be written as (4), if just fault converge and rework yield of the TDR operations are to be optimized,

$$\min_{f_{c_i}, Y_{r_i} \in X} \sum_{i=1}^{22} C_Y(f_{c_i}, Y_{r_i}), X \in [0,1] \quad (4)$$

In Section 5, Real-Coded Genetic Algorithms (RCGAs) are applied to minimize the objective function with feature parameters constrained to practical ranges.

5. Optimization with Real-Coded Genetic Algorithms (RCGAs)

To optimize feature parameters of possible TDR operations in order to find the global optimum of yielded cost of the process flow, genetic algorithms (GAs) are used. GAs are stochastic, directed search algorithms that have proved useful in finding global optima in both static and dynamic environments [16]. GAs work with large populations of candidate solutions that are repeatedly subjected to selection pressure and which undergo naturally occurring genetic operations in the search for improved solutions.

Complex process flows in electronic systems assembly with hundreds of process steps are not uncommon. A general optimization of such a process flow requires an equally large stream of TDR operations resulting in several hundred of variables to be optimized for the trade-off analysis. Traditional Genetic Algorithms (GAs) use the binary representation that evenly discretizes a real variable space. Although binary-coded GAs (BCGAs) have been successfully applied to a wide range of optimization problems, they suffer from the disadvantage that when they are applied to the real-world problems involving a large number of real variables the string encoding the variables have a large string length [16,17] (because the binary substrings representing each parameter with the desired precision are concatenated to

represent an individual). Another drawback of the BCGAs applied to parameter optimization problems in continuous domains comes from the discrepancy between the binary representation space and the actual problem space. A simple solution to these problems is the use of the floating-point representation of parameters [18,19]. Using RCGAs, an individual is coded as a vector of real numbers corresponding to the variables and the string length reduces to the number of variables.

With RCGAs integrated, a process flow cost optimization modeling system according to the framework in Figure 1 has been developed to determine the optimum placement and characteristics of TDR operations in general process flows. The system results in suggestions of where the TDR operations should be inserted and not inserted, and what the optimal values of feature parameters of testing and rework should be based on the optimization analysis of the objective function. The following subsections provide the verification of the optimization model compared with functional computations in simple cases and a general process flow demonstration.

5.1 Verification strategy

To provide testing and demonstration of the approach the following test cases are used:

Test Case 1: Single-test case, fault coverage varies and no rework is included;

Test Case 2: One-TDR operation case in which fault coverage and rework yield are allowed to vary (a maximum of one rework attempt assumed);

Test Case 3: A general process flow case with multiple TDR operations and variable fault coverages and rework yields (a maximum of one rework attempt assumed).

Test Cases 1 and 2 are trivial and used to validate the optimization model by comparison with hand calculations of the yielded cost associated with the process flow. The cases are categorized into high-yield and low-yield inputs in order to reflect the impacts of input yield on the optimum. Test Case 3 demonstrates the value of the optimization methodology on a case where the optimum placement and characteristics of the TDR operations is not obvious.

Table 2: Values of input parameters

	C_{in}	Y_{in}	C_1	Y_1	C_2	Y_2
High-Yield	21	0.97	10	0.98	7	0.93
Low-Yield	21	0.48	10	0.32	7	0.93

5.2 Analysis of the simple cases

For the simple process flow in Figure 6, the optimum solution can be determined from a simple step computation.

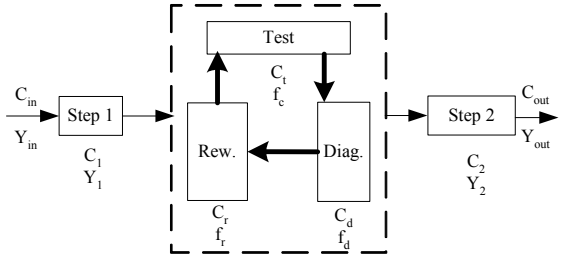


Figure 6: Simple two-step process flow with one possible TDR operation inserted between them.

5.2.1 Single-test case (Test Case 1)

If there is no test, the C_{out} and Y_{out} are computed using (5) and (6) respectively,

$$C_{out} = C_{in} + C_1 + C_2 \tag{5}$$

$$Y_{out} = Y_{in} Y_1 Y_2 \tag{6}$$

the final yielded cost can be derived as (7),

$$C_Y = \frac{C_{out}}{Y_{out}} = \frac{(C_{in} + C_1 + C_2)}{Y_{in} Y_1 Y_2} \tag{7}$$

To evaluate this expression, the input parameters in Table 2 are used. Substituting the inputs in Table 2 into (7), the yielded cost is given as (8),

$$C_Y = \begin{cases} 266.017 & \text{(low - yield)} \\ 41.64037 & \text{(high - yield)} \end{cases} \tag{8}$$

If one test (no rework) operation is inserted between Step 1 and Step 2 in Figure 6, the cost and yield of the process flow is given by,

$$C_{out} = \frac{(C_{in} + C_1) + C_{test}}{(Y_{i_n} Y_1)^{f_c}} + C_2 \tag{9}$$

$$Y_{out} = (Y_{in} Y_1)^{(1-f_c)} Y_2 \tag{10}$$

$$C_Y = \frac{C_{out}}{Y_{out}} = \frac{(C_{in} + C_1 + C_{test}) + C_2 (Y_{in} Y_1)^{f_c}}{Y_{in} Y_1 Y_2} \tag{11}$$

substituting (1) into (11), the yielded cost of the process flow becomes,

$$C_Y = \frac{(C_{in} + C_1 + C_{ft} - r_t p_t) + b_t p_t \ln(1-f_c) + C_2 (Y_{in} Y_1)^{f_c}}{Y_{in} Y_1 Y_2} \tag{12}$$

The equation can be rewritten as,

$$C_Y = A + B\ln(1-f_c) + CD^{f_c} \tag{13}$$

where $A = \frac{C_{in} + C_1 + C_{ft} - r_t p_t}{Y_{in} Y_1 Y_2}$, $B = \frac{b_t p_t}{Y_{in} Y_1 Y_2}$,

$C = \frac{C_2}{Y_{in} Y_1 Y_2}$, $D = Y_{in} Y_1$.

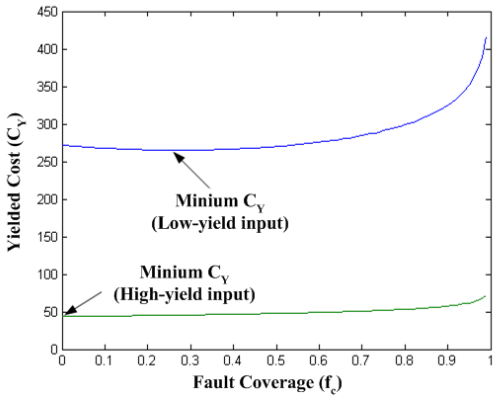


Figure 7: Yielded cost of process flow versus fault coverage of test.

Figure 7 plots (13) for the high-yield and low-yield input cases respectively:

1. For high-yield inputs, the yielded cost of the one-test case primarily depends on the change of $\ln(1-f_c)$ instead of the power of the fault coverage (i.e., D in (13) is very small). The minimum of yielded cost should be at the point when the fault coverage is zero. This is to say that the minimum yielded cost is for no test inserted into the process flow, which makes intuitive sense.
2. For low-yield inputs, we can see that (13) clearly differs from the characteristic of curve of yielded cost at high-yield inputs due to increasing importance of the power of the fault coverage in determining the yielded cost of the product resulting from the process flow. The minimum of yielded cost should be between \$260 and \$280 while the optimal fault coverage falls into the range from 0.2 to 0.3.

Verification for high-yield inputs

The results of the simple calculations above are used to check the optimization methodology developed in this paper. RCGAs are applied to cases of high-yield and low-yield inputs respectively. The specification of RCGAs is:

- Maximum generation= 20
- Population size= 50
- Mutation probability= 0.1
- Crossover probability= 0.95
- $f_T = 0.1$ [4]
- Termination criteria = maximum generations

[4] The threshold of fault coverage, f_T , is defined as the minimum non-zero fault coverage that we can purchase. For any fault coverage below this threshold, there is considered to be no test present (zero fault coverage and zero test cost).

The results from the optimization of yielded cost in the single-test case at high-yield inputs are shown in Figure 8.

In the first generation, the RCGAs have found the optimized solution and show the best fitness (the best value of objective function in one population) of 41.64037 when the corresponding fault coverage is set at a very small value. After the evolution of 20 generations (termination criteria), the algorithm converges to the same value of yielded cost, which agrees with (8). In the last generation, the fault coverage optimization terminates at a small amount of fault coverage (0.01495), lower than the threshold for testing to be present. The test is thus removed from the process flow based on the optimization analysis from the RCGAs, which coincides with the characteristics of Figure 7 (high-yield relation). In addition to validating the optimal yielded cost of the process flow, RCGAs are continuously convergent in searching for the optimum, see Figure 8, and the mean (mean value of objective function in one population) of yielded cost levels out as well.

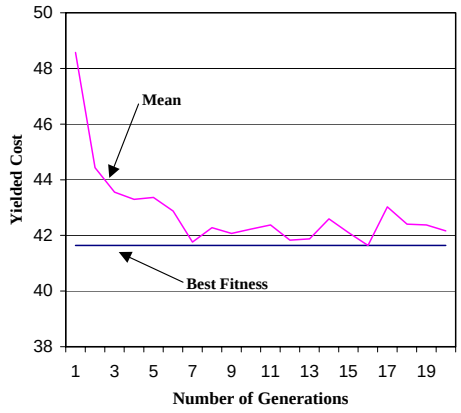
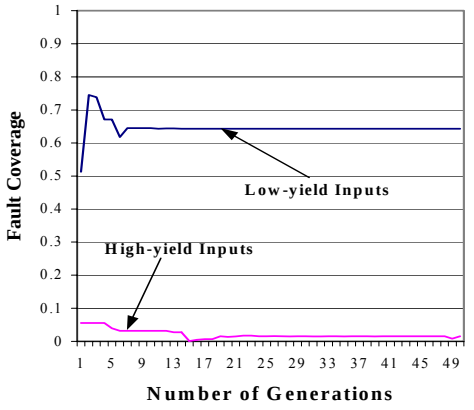


Figure 8: Optimization of yielded cost in the single-test case (Test Case 1) at the high-yield level.

Verification for low-yield inputs

The specification of RCGAs used in the low-yield input scenario is identical to that applied in the high-yield case except the maximum number of generations has been increased to 50. The optimal solution of \$265.1056 and a fault coverage of 0.2707 that agrees with the observation from Figure 7 (low-yield relation) and it is smaller than the result of computed in (8) in which there is no test present.



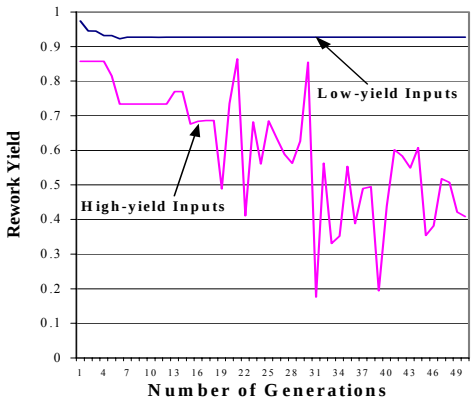


Figure 9: Optimization histories of feature parameters in one-TDR case (Test Case 2).

5.2.2 One-TDR case (Test Case 2)

In this test case, a complete TDR operation is placed between the two process steps in Figure 6 in order to trace the effect of rework yield to yielded cost as well as fault coverage. Optimizations of one-TDR case with feature parameters at high-yield and low-yield inputs are given in Figure 9. The optimal values of fault coverage and rework yield validate the same conclusion that there is no need for a TDR operation for high-yield inputs. In Table 3, the optimum of yielded cost reaches $41.6403 (the same value with the one in single-test case) when the fault coverage convergent to a very small amount (smaller than f_T) and the rework yield levels out without convergence in Figure 9, i.e., the rework is useless and should be removed for the high-yield inputs.

For the low-yield inputs, the rework operation plays an important role on improving the yield of process flow. Table 4 shows an optimal yielded cost of $99.2924, which is much lower than the value we have computed in Section 5.2.1 when the no rework operation was considered.

Table 3: Results of RCGAs applied in the one-TDR case at high-yield inputs (Test Case 2).

Gen.	Best Fitness ($)	Mean ($)	Fault Coverage	Rework Yield
1	44.26007	51.63859	0.055283	0.857406
2	44.26007	46.5915	0.055283	0.857406
3	44.26007	45.59374	0.055283	0.857406
⋮	⋮	⋮	⋮	⋮
48	41.6403	43.80089	0.015053	0.507313
49	41.6403	43.77923	0.007691	0.421582
50	41.6403	43.86956	0.015051	0.408746

Table 4: Results of RCGAs applied in the one-TDR case at low-yield inputs (Test Case 2).

Gen.	Best Fitness ($)	Mean ($)	Fault Coverage	Rework Yield
1	103.0425	184.1419	0.513008	0.973706
2	100.6619	129.323	0.7451	0.944974
3	100.5055	111.2269	0.738354	0.944896
⋮	⋮	⋮	⋮	⋮
48	99.29243	110.2903	0.643252	0.927038
49	99.29243	110.9406	0.643252	0.927038
50	99.29243	111.3758	0.643252	0.927038

5.2.3 Optimization in complex process flow (Test Case 3)

For use as a general demonstration, the process flow shown in Figure 10 has been used (the inputs of steps in Table 5). Possible TDR locations have been marked. For the comparison analysis, the projection of optimum fault coverage and rework yield of all possible TDR operations for various fixed cost values of test and rework are shown in Figure 11 and 12. Figure 13 presents the optimization of yielded cost of the complex process flow, in which the optimum of the objective function converges as the number of generations of RCGAs increases.

Figure 11(a) has low (inexpensive) test and rework; as a result, 9 of the 11 possible locations for tests are present (i.e., have fault coverages above the threshold for testing). Because rework is also inexpensive in case (a), rework is being done at all the actual test locations. Case 11(b) has inexpensive testing (same as case 11(a)), but the rework is expensive. As a result, significantly fewer rework opportunities are actually exercised. Notice also that the optimum test locations (and fault coverages) change even though the characteristics of the testing are the same due to the inclusion or exclusion of rework possibilities. Figure 12 shows the same test costs as 11(a) and (b), but no rework is allowed – the

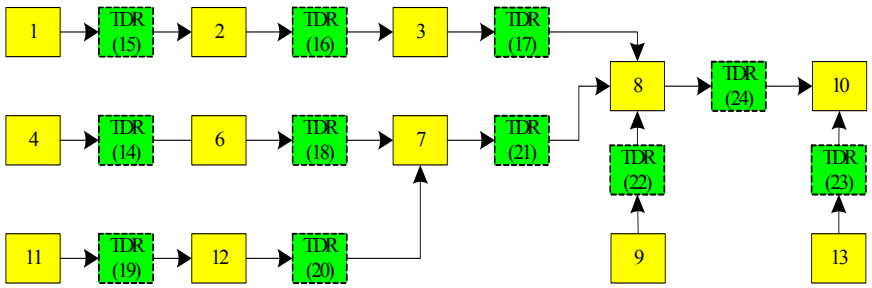
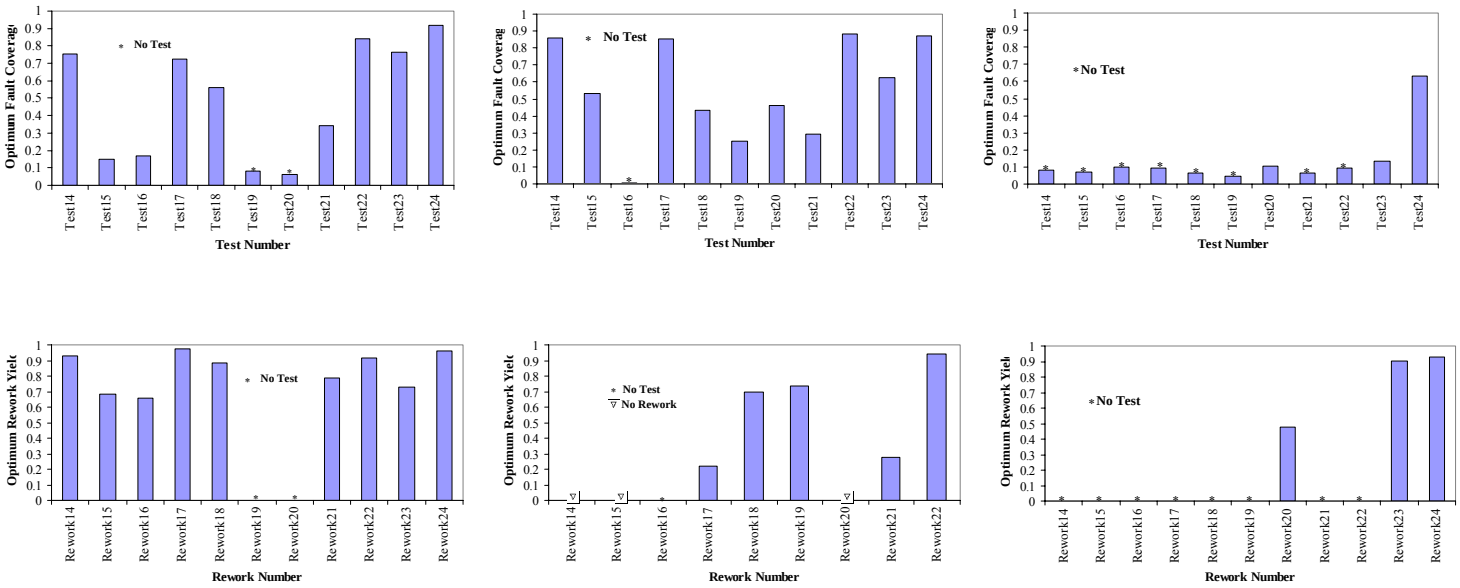


Figure 10: A general process flow with all possible TDR operations.

Table 5: Characteristics of and inputs to the steps in process flow in Figure 10.

step	$cost_{in}$ ($)	$yield_{in}$	cost ($)	yield
1	41	0.91	21.1	0.95
2	62.1	0.92	12.3	0.88
3			14	0.89
4	60	0.36	23.8	0.94
6			45	0.96
7			11.3	0.86
8			33.8	0.92
9	37	0.42	13	0.90
10			15	0.92
11	9	0.95	60	0.95
12			78	0.91
13	75	0.96	54	0.94



(a): C_{ft}= \$1 and C_{fr}= \$1 **(b): C_{ft}= \$1 and C_{fr}= \$100** **(c): C_{ft}= \$50 and C_{fr}= \$100**

Figure 11: Computed optimum feature parameter values for TDR operations in the process flow shown in Figure 10. Various fixed test and rework costs used in (1) and (2) assumed.

optimum test locations and fault coverages again differ from cases 11(a) and 11(b). Case 11(c) has expensive test and expensive rework. As a result, only 3 or 11 possible test locations are used (Test 20, 23 and 24 only), however, rework is included with all three of these tests.

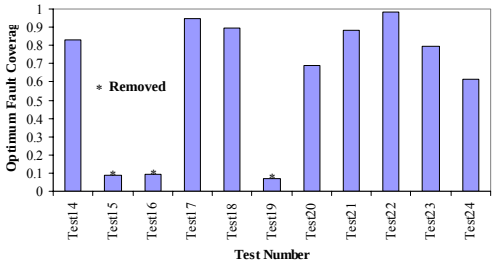


Figure 12: Computed optimum fault coverage of test operations (C_{ft}= \$1 and no rework).

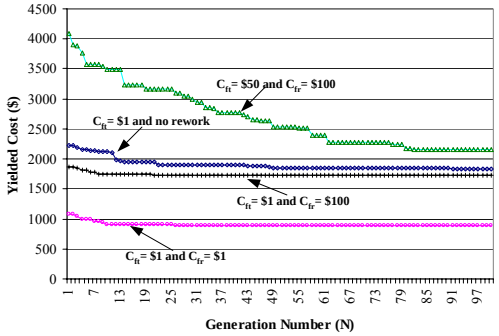


Figure 13: Optimization of yielded cost in Test Case 3 for various level of fixed cost of test and rework operations.

The optimization using RCGAs for the process flow in Figure 10 takes less than 3000

seconds for 100 generations, thus making it an attractive option for complex electronic systems.

6. Summary

This paper describes a framework for the optimization of Test/Diagnosis/Rework (TDR) location(s) and characteristics in an electronic system manufacturing processes. A new search algorithm is developed and used to analyze complex process flows and to obtain values of a multi-variable function with a stream of feature parameters included. An optimization modeling system with Real-Coded Genetic Algorithms (RCGAs) integrated has been developed to optimize critical parameters and possible TDR locations for general process flows. The methodology developed and demonstrated in this paper guides the placement of TDR operations in practical manufacturing processes. The ability to optimize the TDR operations can also be used as the feedback to DFT of the electronic systems showing which portion should be redesigned to accommodate the testing for a higher level of fault coverage and where there is less need for test to decrease the cost of products.

References

1. T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Trans. on Comp.,* vol. C-30, no. 12, pp. 987-988, December 1981.

2. E. H. Volkerink, A. Khoche, L. A. Kamas, J. Revoir, and H. G. Kerkhoff, "Tackling Test Trade-offs from Design, Manufacturing to Market Using Economic Modeling," *Proc. of Int. Test Conference,* pp. 1098-1107, Nov. 2001.

3. P. K. Nag, A. Gattiker, S. Wei, R. D. Blanton, and W. Maly, "Modeling the Economics of Testing: A DFT Perspective," *IEEE Design & Test of Comp.,* vol. 19, pp. 29-41, Jan/Feb 2002.

4. C. Dislis, J. H. Dick, I. D. Dear, I. N. Azu, and A. P. Ambler, Economics modeling for the determination of test strategies for complex VLSI boards, *Proc. of the Int. Test Conf.*, pp. 210-217, 1993.

5. M. Tegethoff, and T. Chen, "Defects, fault coverage, yield and cost, in board manufacturing," *Proc. of the Int. Test Conf.*, pp. 539-547, 1994.

6. M. Abadir, A. Parikh, L. Bal, P. Sandborn, and C. Murphy, "High level test economics advisor," *Journal of Electronic Testing: Theory and Applications*, vol. 5, no. 2&3, pp. 195-206, 1994.

7. T. Trichy, P. Sandborn, R. Raghavan, and S. Sahasrabudhe, "A New Test/Diagnosis/Rework Model for Use in Technical Cost Modeling of Electronic Systems Assembly," *Proc. of Int. Test Conf.,* pp. 1108-1117, 2001.

8. M. Driels and J. S. Klegka, "Analysis of Alternative Rework Strategies for Printed Wiring Assembly Manufacturing Systems," *IEEE Trans. on Components, Hybrids and Manufacturing Tech.*, vol. 14, pp. 637-644, Sept. 1991.

9. B. Davis, *The Economics of Automatic Testing,* McGraw-Hill Book Company, 1994.

10. P. Goel, "Test Generation costs analysis and projections," *Proc. of Design Automation Conference,* pp. 77-84, 1980.

11. N. Rao, "On Parallel Algorithms for Single-Fault Diagnosis in Fault Propagation Graph Systems," *IEEE Trans. on Parallel and Distributed Sys.,* vol. 7, pp. 1217-1223, Dec. 1996.

12. K. Choi and A. Chatterjee, "Efficient Instruction-level Optimization Methodology for Low-Power Embedded Systems," *Proc. of the 14th Int. Symposium on System Synthesis,* pp. 147-152, 2001.

13. G. Chartrand and O. R. Oellermann, *Applied and Algorithmic Graph Theory,* McGraw-Hill, Inc., 1993.

14. R.E. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM J. Computing*, vol. 1, no.2, pp.146-160, 1972.

15. D. Becker and P. Sandborn, "One the Use of Yielded cost in Modeling Electronic Assembly Processes," *IEEE Trans. on Electronics Packaging Manufacturing*, vol. 24, pp. 195-202, July 2001.

16. R. L. Haupt, and S. E. Haupt, *Practical Genetic Algorithms,* Wiley, 1998.

17. A. Oyama, S. Obayashi, and K. Nakahashi, "Wing Design Using Real-coded Adaptive Range Genetic Algorithm," *Proceedings of IEEE Int. Conf. on Systems, Man, and Cybernetics,* vol. 4, pp. 475-480, 1999.

18. C. Z. Janikow and Z. Michalewicz, "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms," *Proc. of the 4th Int. Conference on Genetic Algorithms*, pp. 31-36, 1991.

19. A. H. Wright, "Genetic Algorithms for Real Parameter Optimization," *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 205-218, 1991.