International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 3, June 2019, pp. 2141~2151
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i3.pp2141-2151

Testing embedded system through optimal mining technique (OMT) based on multi-input domain

J. K. R. Sastry[1], M. Lakshmi Prasad[2]
[1]Department of Computer Science and Engineering, KLEF Deemed to be University, India
[2]Department of Computer Science and Engineering, NBKR Institute of Science and Technology, India

Article Info

Article history:

Received Jun 16, 2018
Revised Nov 19, 2018
Accepted Dec 10, 2018

Keywords:

Embedded system
Optimal mining technique
Output domain testing
Pseudo-exhaustive testing

ABSTRACT

Testing embedded systems must be done carefully particularly in the significant regions of the embedded systems. Inputs from an embedded system can happen in multiple order and many relationships can exist among the input sequences. Consideration of the sequences and the relationships among the sequences is one of the most important considerations that must be tested to find the expected behavior of the embedded systems. On the other hand combinatorial approaches help determining fewer test cases that are quite enough to test the embedded systems exhaustively. In this paper, an Optimal Mining Technique that considers multi-input domain which is based on built-in combinatorial approaches has been presented. The method exploits multi-input sequences and the relationships that exist among multi-input vectors. The technique has been used for testing an embedded system that monitors and controls the temperature within the Nuclear reactors.

Corresponding Author:

J. K. R. Sastry,
Department of Electronics and computer Engineering,
KLEF deemed to be University,
Vaddeswaram, Guntur District, Andhra Pradesh, India.
Email: drsastry@kluniversity.in

1. INTRODUCTION

Testing is playing a significant role in the development of any system which is a systematized process to verify the reliability, behaviour and performance of a system against considered stipulations. It enables a device or a system to be as defect-free as possible which act as a one of the detective measures, and verification is one of the corrective measures of quality.

Black-box testing inspects the functionality of an application without seeing into its internal structures or workings. It mainly concentrates on the functional requirements of the embedded system without considering the internal working of the system. The main aim of this testing is to select the acceptable test cases and detect as many faults based on requirements specification at least cost and time. Testing embedded systems involves testing software, hardware and both. Testing of hardware and software however can be conducted independently and then the testing has to be undertaken after the software is migrated into the hardware.

Embedded systems are a mixture of various computing devices, such as microcontrollers, application-specific integrated circuits, and digital signal processors. Some widely used systems in real world applications such as routers, power plant system, medical instrument system, home appliances, air traffic control station, and firewalls, telecommunication exchanges, robotics, industrial automation and smart cards etc. are example of embedded system. Falseness in hardware systems may be designated in terms of defect, error and faults.

Combinatorial testing is commonly utilized black-box practice that could dramatically diminish the number of test cases, as it is a highly competent technique to perceive software faults. This method originates test cases from input domain of the system under test. But, when the input domain is noteworthy and the output domain is much tiny, it is desirable to go for testing the output domain either exhaustively [1] or as much as possible.

For a few safety critical embedded systems, building test cases drawn from multi-input domain is a necessity as multiple inputs can occur at the same time. However the an embedded system must also be tested form other perspectives that include input, output, input-output and Multi-output domain Generation of test cases based on Multi-input perspective will be more suitable than other perspective's as it guarantees that all or as many possible input combinations are comprehensively tested.

Exhaustive testing [2] considering non multi-input domains is out of question when many input variables exist and they act in several combinations. Pseudo-Exhaustive testing aims at considering only those combinations that will most likely result in failure conditions. Optimal Mining Technique (OMT) derives test cases by choosing certain combinations of either the inputs or outputs based on the possibility of occurrence of multi- output or multi-input domain of an embedded system such as TMCNRS which monitors and controls temperatures within nuclear reactor systems.

In the case of TMCNRS, the occurrences temperatures within nuclear records can happen simultaneously are independently. In an embedded system processing tasks are designed for handling multi-inputs which occur simultaneously. These tasks must be tested thoroughly to guarantee the proper working of the embedded system. Test cases should be generated to verify the functionality of occurrence of proper outputs based on the Occurrence of Multi-input.

Problem

In the case of embedded systems, Inputs occur as a set in addition to the occurrence of independent inputs. The behaviour of an embedded system when multi-inputs occurs must be tested to find the whether the system has been properly developed to process multi-inputs that occur simultaneously.

Proposed solution

An improved optimal Mining technique is presented in this paper, the range of values that must be used for generating the test data have also been pre-identified and mapped with output variables. The multi input relationships and also the input output relationships which can be used as a database for mining relationship pattern for further modelling. The pattern of occurrence of the input variable can be determined by using a mining algorithm or through manual inspection.

2. RELATED WORKS

Lakshmi Prasad, *et al*., [3] had presented a comprehensive survey on combinatorial testing. In [4] a method is proposed that deals with generation of test cases based on the input domain considering with special consideration to testing standalone embedded system. The algorithm can successfully generate pairs for those input parameters and eliminate non related input pairs thereby reducing the size of the test suite to a minimum. Lakshmi Prasad, *et al*., [5] had proposed several combinatorial methods for testing an embedded system.

Gray. D. M. Cohen, *et al*., [6] introduced the combinatorial design approach for generating the test cases automatically. They have described an application which is developed using the method presented by them. They have shown that the time required for the development of test plan has been reduced considerably and they have also shown that the entire code has been covered using the test plan that has been used to generate test cases.

Cohen D. M., *et al*., [7] have presented a system called AETG. The approach presented by them considers all combinations of input parameters that include pair-wise, tripe-wise and n-wise. The approaches presented by them will breed all the valid test pairs ignoring the invalid test pairs. The numbers of test cases generally are of the logarithmic order of the number of input variables used. The AETG has been used for undertaking different types of testing that include unit, functional, acceptance, system, integration, regression and inter-operability testing.

Cohen D. M., [8] have presented a method and a system for enumerating a minimal number of test cases for systems with interacting elements that have relationships between the elements and the number of characteristics evaluated for each element. In the method, the user enters values for each of the elements and then defines relationships between the elements. Our method then enumerates a table of test cases for each relationship between elements using deterministic procedures, when applicable, and random procedures when deterministic procedures are not applicable. After a table of test cases is generated for each of the relation, the method combines relationships into a single table of test cases.

The significant expansion of autonomous control and information processing capabilities in the coming generation of mission software systems results in a qualitatively larger space of behaviours that needs to be "covered" during testing, not only at the system level but also at subsystem and unit levels Tung, [9]. A major challenge in this area is to automatically generate a relatively small set of test cases that, collectively, guarantees a selected degree of coverage of the behaviour space. They described an algorithm for a parametric test case generation tool that applies a combinatorial design approach to the selection of candidate test cases. Evaluation of this algorithm on test parameters from the Deep Space One mission reveals a valuable reduction in the number of test cases, when compared to an earlier home-brewed generator. Lei Y., *et al.*, [10] have used a criterion which is test specification based. The criterion considers each pair of input variables, and every value pair of the input pair considered and every value selected covered through a test case. They have evolved this strategy for carrying pair-wise testing.

Covering arrays have been augmented further by Cohen M. B., *et al*., [11] through inclusion of the concept called Annealing. This has led to special array containing several sub-arrays which all together contain all the t-tuples, each tuple appearing at least once. The strength of the array is measured through t number of tuples contained in the array considering all the sub-array contained in it. They have analyzed all the arrays that have strength of 3 tuples using recursive combinatorial construction and using search techniques. The technique used by them leveraged optimality and efficiency of the size through use of combinatorial construction and heuristic search.

Further addition to the recursive combinatorial generation added with heuristic search has been made through detection of interaction of multiple components that lead to different kinds of failures. R. Kuhn, *et al.*, [12] have applied this approach to the real world applications and obtained the analytical results. The way a system is tested depended on the type of the system. Choice of an appropriate testing method is crucial for making testing effective and rational. The application related to flow of water inside a carbon Nano tube that is single walled and in which several temperature gradients exist has been tested by Shiomi J., *et al*., [13] by using combinatorial method.

Ochoa, *et al*., [14] had introduced the box-fusion which is an approach to improve pair wise testing. Box-Fusion approach was guessed and a case study was carried out by using two software implementations: the Simple LTL Generator that builds Linear Temporal Logic (LTL) formulae with atomic propositions and the prospect algorithm that can produce LTL formulae from more than 31,000 possible input combinations. They have presented evaluation of Box-Fusion approach which considers, pair wise testing approach, annotated control flow graphs approach and regression testing approach.

M. Lakshmi Prasad, *et al*., [15]-[19] had built test cases by particle swarm optimization (PSO) for multi output domain embedded systems using combinatorial techniques. They also used neural network based strategy for automated construction of test cases for testing an embedded system using combinatorial techniques. They also developed generating test cases for testing web sites through neural networks and input pairs. They also generated test cases using combinatorial methods based multi-output domain of an embedded system through the process of optimal selection.

Abdul Rahman, *et al*., [20] had presented a survey on input-output relationship relation to test data generation strategies. They reviewed the existing combinatorial test data generation strategies supporting the IOR features specifically taking the nature inspired algorithm as the main basis. Benchmarking results illustrate the comparative performance of existing nature inspired algorithm based strategies supporting IOR.

Combinatorial methods can also be used for testing software that predominantly uses logical expressions based on Boolean or binary inputs. In the software used related to most of the safety critical applications, Boolean expressions are used extensively. S. Vilkomir, [21] has steadied effectiveness of combinatorial testing when binary inputs are used.

Deepa Gupta, *et al*., [22] had proposed a sequence generation of test cases using pair wise approach. They presented an approach which uses the series origination approach for pair wise test case origination. This approach makes certain to disseminate the required intent of trial run cases which cover all available relations between all instructions pairs at least once. Trial run selection specification is this approach is based on combinatorial testing.

Jose Torres-Jimenez, *et al*., [23] Covering arrays are combinatorial structures which have applications in fields like software testing and hardware Trojan detection. In this paper we proposed a two-stage simulated annealing algorithm to construct covering arrays. The proposed algorithm is instanced in this paper through the construction of ternary covering arrays of strength three. We were able to get 579 new upper bounds. In order to show the generality of our proposal, we defined a new benchmark composed of 25 instances of MCAs taken from the literature, all instances were improved.

S. Route [24] Clients today want more for less and the IBM test mantra of Test Less Test Right helps address this ask by placing Combinatorial Test Design (CTD) at the heart of the solution. This document presents two case studies of CTD implementation in client engagements and focuses on the approach, process and challenges addressed to scale up the implementation and make CTD a mainstream activity. The IBM Focus tool was used in both cases to implement Combinatorial Test Design for optimization of tests and for reducing test effort while increasing test coverage.

P. S., M. B., *et al*., [25] Combinatorial Testing is a test design methodology that aims to detect the interaction failures existing in the software under test. The combinatorial input space model comprises of the parameters and the values it can take. Building this input space model is a domain knowledge and experience intensive task. The objective of the paper is to assist test designer in building this test model. A rule based semi-automatic approach is proposed to derive the input space model elements from Use case specifications and UML use case diagrams. A natural language processing based parser and an XMI based parser are implemented. The rules formulated are applied on synthetic case studies and the output model is evaluated using precision and recall metrics. The results are promising and this approach will be of good use to the test designer.

Y. Yao, *et al*., [26] as an effective software testing technique, combinatorial testing has been gradually applied in various types of test practice. In this case, it is necessary to provide useful combinatorial testing tools to support the application of combinatorial testing technique on industrial scenarios, as well as the academic research for combinatorial testing technique. To this end, on the basis of the research results of this group, a suite of combinatorial testing tools has been developed, whose functions include test case generation, test case optimization, and etc. For the requirements from both industrial and academic scenarios, the tools should be configurable, scalable, modular, and etc. This paper gives a brief introduction to the design and implementation of these tools. Keywords-combinatorial testing, combinatorial testing tools, test generation, test prioritization.

3. APPLICATION OF OPTIMAL MINING TECHNIQUE TO PILOT PROJECT

The modified OMT algorithm and its application to the pilot project are presented below:

3.1. Steps of optimal MINING technique

a. Step-1

Determine the regular and embedded system specific input variables from the test requirements specification. Input variables that are of continuous nature have been selected. In this OMT method only, the multi-input variables that are of continuous in nature have been considered. The details of input variables i.e. regular and ES specific selected are shown in Table 1 and Table 2. The details of output variables i.e. regular and ES specific selected are shown in Table 3 and Table 4. The range of values that must be used for generating the test data have also been pre-identified and mapped with input variables. The relationships that exist between the input variables and its corresponding related input variables can be used as the basis for generating the test cases.

Table 1. Regular Input Variables Traced Out of Test Specification of the Pilot Project

Variable Serial No	Input Variable Name	Type of Variable	Start Value	End Value
I1	INIT-MESSAGE	Discrete	"TMCNRS"	"TMCNRS"
I2	KEY-1	Discrete	0	Z
I3	KEY-2	Discrete	0	Z
I4	KEY-3	Discrete	0	Z
I5	KEY-4	Discrete	0	Z
I6	KEY-5	Discrete	0	Z
I7	MSG-PW-ENTRY	Discrete	"Enter Password'	"Enter Password'
I8	PASS-WD	Discrete	"abcde"	"abcde"
I9	TEMP1	Continuous	1	255
I10	REF1	Continuous	1	255
I11	TEMP2	Continuous	1	255
I12	REF2	Continuous	1	255

Table 2. Embedded Specific Input Variables Traced Out of Test Specification of the Pilot Project

Variable Serial No	Input Variable Name	Type of Variable	Start Value	End Value
I13	R-Command	Constant String	-	-
I14	Th-Command	Constant String	-	-
I15	T-Command	Constant String	-	-
I16	O-Command	Constant String	-	-
I17	V-Command	Constant String	-	-
I18	MEM-LOC-1	Constant	0	255
I19	MEM-LOC-2	Constant	0	255
I20	TEST-PORT1	Constant String	-	-
I21	TEST-PORT2	Constant String	-	-
I22	TEST-PORT3	Constant String	-	-
I23	TEST-PORT4	Constant String	-	-
I24	TEST-PORT5	Constant String	-	-

Table 3. Details of the Regular Output Variables Related to the Pilot Project

Variable Serial No.	Output Variable Name	Type of Variable	Start Value	End Value
O1	LCD-STAT	Discrete	N	Y
O2	LCD-WRITE	Discrete	"ABC"	"ABC"
O3	HOST-STAT	Discrete	N	Y
O4	HOST-WRITE	Discrete	"ABC"	"ABC"
O5	TEMP1-OSIG	Discrete	N	Y
O6	TEMP2-OSIG	Discrete	N	Y
O7	PUMP1-OSIG	Discrete	N	Y
O8	PUMP2-OSIG	Discrete	N	Y
O9	BUZZER-OSIG	Discrete	N	Y

Table 4. Details of the Embedded Specific Output Variables related to the Pilot Project

Variable Serial No.	Output Variable Name	Type of Variable	Start Value	End Value
O12	TEMP1-TSIG	Continuous	0	32756
O13	TEMP1-VSIG	Continuous	0	32756
O14	TEMP1-RES-TIME	Continuous	0	32756
O15	TEMP1-THRU-TIME	Continuous	0	32756
O16	TEMP1-RANGE-STA	Discrete	N	Y
O17	PUMP1-TSIG	Continuous	0	32756
O18	PUMP1-VSIG	Continuous	0	32756
O19	PUMP1-RES-TIME	Continuous	0	32756
O20	PUMP1-THRU-TIME	Continuous	0	32756
O21	TEMP2-TSIG	Continuous	0	32756
O22	TEMP2-VSIG	Continuous	0	32756
O23	TEMP2-RES-TIME	Continuous	0	32756
O24	TEMP2-THRU-TIME	Continuous	0	32756
O25	TEMP2-RANGE-STA	Discrete	N	Y
O26	PUMP2-TSIG	Continuous	0	32756
O27	PUMP2-VSIG	Continuous	0	32756
O28	PUMP2-RES-TIME	Continuous	0	32756
O29	PUMP2-THRU-TIME	Continuous	0	32756
O30	BUZZER-TSIG	Continuous	0	32756
O31	BUZZER-VSIG	Continuous	0	32756
O32	BUZZER-RES-TIME	Continuous	0	32756
O33	BUZZER-THRU-TIME	Continuous	0	32756

b. 3Step-2

Determine the input-input relationships and also the relationships with the output variable which can be used as a database for mining relationship pattern for further modelling. Sample associativity and the relationships among the input and output variables are shown in the Table 5.

Table 5. Input Variables and its Relationship with Output Variables

S. No	Input Variables		Output Variables						
	Input 1	Input 2	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6	Output 7
1.	INIT-MESSAGE	-	LCD-STAT	LCD-WRITE	-	-	-	-	-
2.	MSG-PW-ENTRY	-	LCD-STAT	LCD-WRITE	-	-	-	-	-
3.	PASS-WD	-	LCD-STAT	LCD-WRITE	-	-	-	-	-
4.	TEMP1	-	LCD-STAT	LCD-WRITE	-	-	-	-	-
5.	TEMP1	-	HOST-STAT	HOST-WRITE	-	-	-	-	-
6.	TEMP1	-	TEMP1-OSIG	-	-	-	-	-	-
7.	TEMP1	-	TEMP1-OSIG	TEMP1-VSIG	TEMP1-THRU-TIME	-	-	-	-
8.	TEMP1	-	TEMP1-OSIG	TEMP1-VSIG	TEMP1-RANGE-STA	-	-	-	-
9.	TEMP1	-	TEMP1-OSIG	TEMP1-VSIG	PUMP1-OSIG	-	-	-	-
10.	TEMP1	-	TEMP1-OSIG	TEMP1-VSIG	BUZZER-OSIG	-	-	-	-
11.	REF1	-	LCD-STAT	LCD-WRITE	-	-	-	-	-
12.	REF1	-	PUMP1-OSIG	-	-	-	-	-	-
13.	REF1	-	REF1-RANGE-STA	-	-	-	-	-	-
14.	TEMP2	-	LCD-STAT	LCD-WRITE	-	-	-	-	-
15.	TEMP2	-	HOST-STAT	HOST-WRITE	-	-	-	-	-
16.	TEMP2	-	TEMP1-OSIG	-	-	-	-	-	-
17.	TEMP2	-	TEMP2-OSIG	TEMP2-VSIG	TEMP2-THRU-TIME	-	-	-	-
18.	TEMP2	-	TEMP2-OSIG	TEMP2-VSIG	TEMP2-RANGE-STA	-	-	-	-
19.	TEMP2	-	TEMP2-OSIG	TEMP2-VSIG	PUMP2-OSIG	-	-	-	-
20.	TEMP2	-	TEMP2-OSIG	TEMP2-VSIG	BUZZER-OSIG	-	-	-	-

c. Step-3

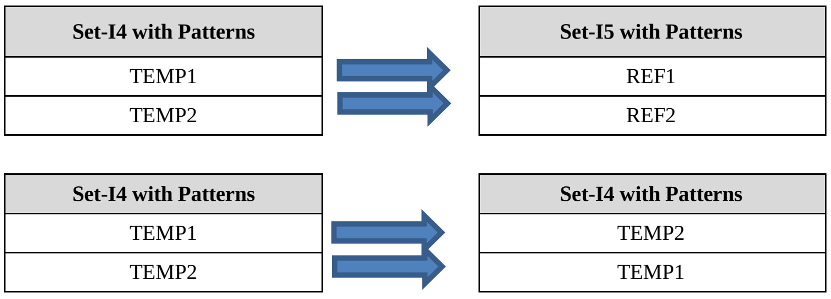
A set of input variables occurs in union. The set of input variables behave in a pattern. The pattern of occurrence of the input variable can be determined by using a mining algorithm or through manual inspection. Following are the input sets and the pattern of occurrence of those sets which can be mined or manually determined.

Set-I1 with Patterns
INIT_MESSAGE
MSG-PW-ENTRY
PASS-WD
Set-I2 with Patterns
TEST-PORT-1
TEST-PORT-2
TEST-PORT-3
TEST-PORT-4
TEST-PORT-5
Set-I3 with Patterns
Key-1
Key-2
Key-3
Key-4
Key-5

Set-I4 with Patterns
TEMP1
TEMP2
Set-I5 with Patterns
REF1
REF2
Set-I6 with Patterns
R-Command
Th-Command
T-Command
O-Command
V-Command
Set-I7 with Patterns
MEM-LOC-1
MEM-LOC-2

d. Step-4

Relate the input patterns through coherence between an input variable and input variable set as shown below.



Set-I6 with Patterns
R-Command
Th-Command
T-Command
O-Command
V-Command

Set-I6 with Patterns
R-Command
Th-Command

e. Step-5

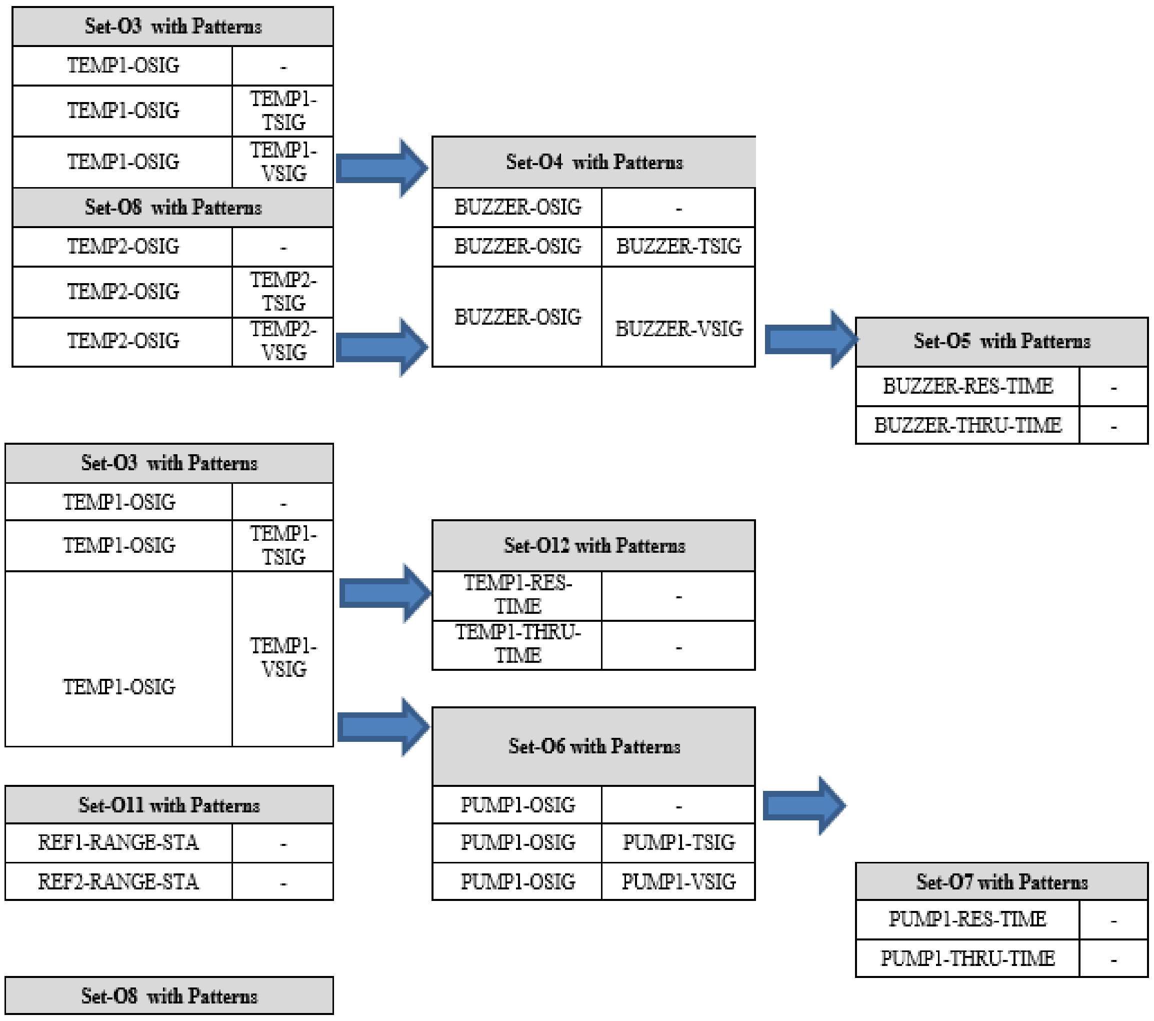
Trace out the output vectors having variables of similar nature and domain. Following are the output vectors related to example application.

Set-O1 with Patterns	
LCD-STAT	-
LCD-STAT	LCD-WRITE
Set-O2 with Patterns	
HOST-STAT	-
HOST-STAT	HOST-WRITE
Set-O3 with Patterns	
TEMP1-OSIG	-
TEMP1-OSIG	TEMP1-TSIG
TEMP1-OSIG	TEMP1-VSIG
Set-O4 with Patterns	
BUZZER-OSIG	-
BUZZER-OSIG	BUZZER-TSIG
BUZZER-OSIG	BUZZER-VSIG
Set-O5 with Patterns	
BUZZER-RES-TIME	-
BUZZER-THRU-TIME	-

Set-O6 with Patterns	
PUMP1-OSIG	-
PUMP1-OSIG	PUMP1-TSIG
PUMP1-OSIG	PUMP1-VSIG
Set-O7 with Patterns	
PUMP1-RES-TIME	-
PUMP1-THRU-TIME	-
Set-O8 with Patterns	
TEMP2-OSIG	-
TEMP2-OSIG	TEMP2-TSIG
TEMP2-OSIG	TEMP2-VSIG
Set-O9 with Patterns	
PUMP2-OSIG	-
PUMP2-OSIG	PUMP2-TSIG
PUMP2-OSIG	PUMP2-VSIG
Set-O10 with Patterns	
PUMP2-RES-TIME	-
PUMP2-THRU-TIME	-
Set-O11 with Patterns	
REF1-RANGE-STA	-
REF2-RANGE-STA	-

f. Step-6

Relate the output patterns through coherence between an output variable and output variable set as shown below.



Set-O3 with Patterns	
TEMP1-OSIG	-
TEMP1-OSIG	TEMP1-TSIG
TEMP1-OSIG	TEMP1-VSIG

Set-O8 with Patterns	
TEMP2-OSIG	-
TEMP2-OSIG	TEMP2-TSIG
TEMP2-OSIG	TEMP2-VSIG

Set-O4 with Patterns	
BUZZER-OSIG	-
BUZZER-OSIG	BUZZER-TSIG
BUZZER-OSIG	BUZZER-VSIG

Set-O5 with Patterns	
BUZZER-RES-TIME	-
BUZZER-THRU-TIME	-

Set-O3 with Patterns	
TEMP1-OSIG	-
TEMP1-OSIG	TEMP1-TSIG
TEMP1-OSIG	TEMP1-VSIG

Set-O12 with Patterns	
TEMP1-RES-TIME	-
TEMP1-THRU-TIME	-

Set-O6 with Patterns	
PUMP1-OSIG	-
PUMP1-OSIG	PUMP1-TSIG
PUMP1-OSIG	PUMP1-VSIG

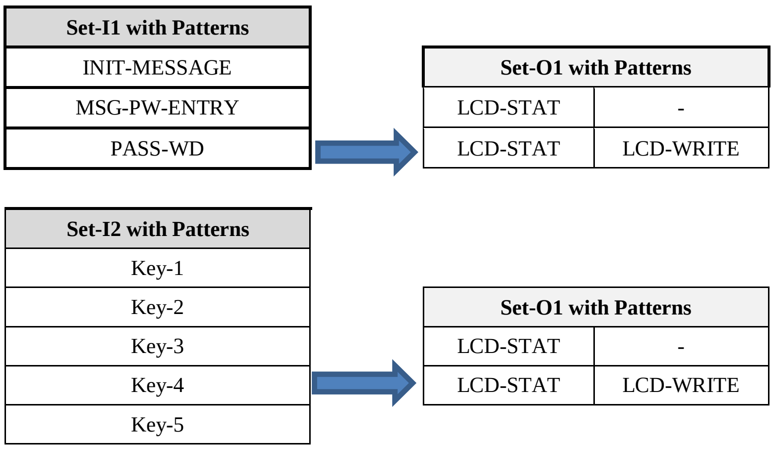
Set-O11 with Patterns	
REF1-RANGE-STA	-
REF2-RANGE-STA	-

Set-O7 with Patterns	
PUMP1-RES-TIME	-
PUMP1-THRU-TIME	-

Set-O8 with Patterns

g. Step-7

Develop input vector relationship with the output vector patterns.

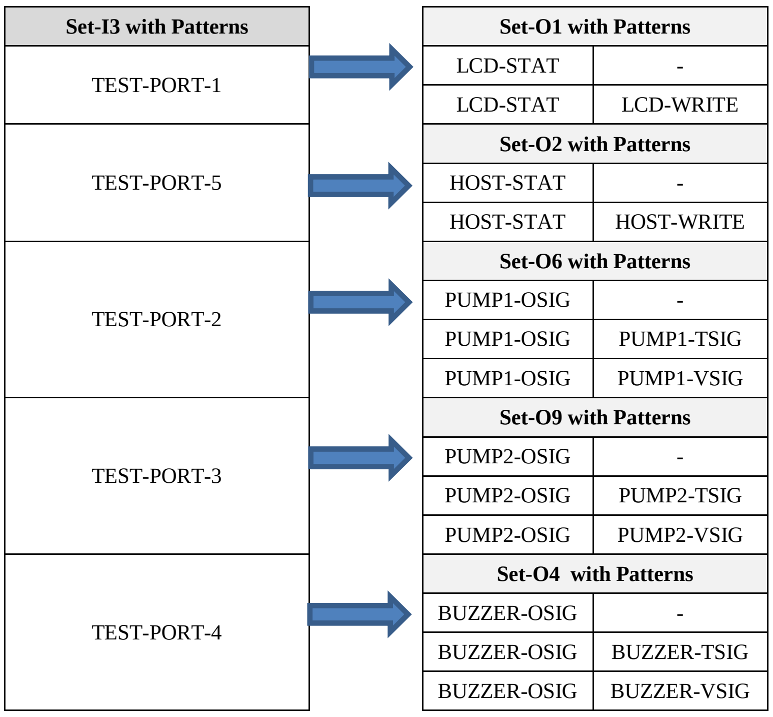


Set-I1 with Patterns
INIT-MESSAGE
MSG-PW-ENTRY
PASS-WD

Set-O1 with Patterns	
LCD-STAT	-
LCD-STAT	LCD-WRITE

Set-I2 with Patterns
Key-1
Key-2
Key-3
Key-4
Key-5

Set-O1 with Patterns	
LCD-STAT	-
LCD-STAT	LCD-WRITE



Set-I3 with Patterns	Set-O1 with Patterns	
TEST-PORT-1	LCD-STAT	-
	LCD-STAT	LCD-WRITE
	Set-O2 with Patterns	
TEST-PORT-5	HOST-STAT	-
	HOST-STAT	HOST-WRITE
	Set-O6 with Patterns	
TEST-PORT-2	PUMP1-OSIG	-
	PUMP1-OSIG	PUMP1-TSIG
	PUMP1-OSIG	PUMP1-VSIG
	Set-O9 with Patterns	
TEST-PORT-3	PUMP2-OSIG	-
	PUMP2-OSIG	PUMP2-TSIG
	PUMP2-OSIG	PUMP2-VSIG
	Set-O4 with Patterns	
TEST-PORT-4	BUZZER-OSIG	-
	BUZZER-OSIG	BUZZER-TSIG
	BUZZER-OSIG	BUZZER-VSIG

Generate test cases

Commencing from input cells trace out the pattern and pattern relationships till no linkage exists such that all tracks are identified. This could be considered as multiply function commencing from input to the last output variable. Each track as such is a test case by itself. The test cases generated are shown in the Table 6.

Table 6. Generated Test cases for TMCNRS Using OMT

Test Case Serial No.	Function Serial No.	Function to be tested	Sub Test Case Serial No.	Type of testing to be carried	Input Vatiable-1/ Command	Input Vatiable-2	Output Variable	Expected Output Value	Expected Output value
1	1	Write Initial message to LCD	1B	Test whether initial message is Displayed Properly	INIT-MESSAGE	-	LCD-STAT LCD-WRITE	Y "TMCNRS"	Y "TMCNRS"
2	2	Write enter password message on LCD	2B	Test whether the message to enter password is displayed properly	MSG-PW-ENTRY	-	LCD-STAT LCD-WRITE	Y "Enter Password'	Y "Enter Password'
3	8	Compare password and write password mismatch on to LCD	8B	Test for password Mismatch	PASS-WD	-	LCD-STAT LCD-WRITE	Y "Miss Match Password'	Y "Miss Match Password'
4	8	Compare password and write password mismatch on to LCD	8C	Test for password Match	PASS-WD	-	LCD-STAT LCD-WRITE	Y "Match Password "	Y "Match Password "
5	3	Read Key1 and write to LCD	3B	Test whether entered Key1 is displayed properly on LCD	KEY-1	-	LCD-STAT LCD-WRITE	Y "a"	Y "a"
6	4	Read Key2 and write to LCD	4B	Test whether entered Key2 is displayed properly on LCD	KEY-2	-	LCD-STAT LCD-WRITE	Y "b"	Y "b"
7	5	Read Key3 and write to LCD	5B	Test whether entered Key3 is displayed properly on LCD	KEY-3	-	LCD-STAT LCD-WRITE	Y "c"	Y "c"
8	6	Read Key4 and write to LCD	6B	Test whether entered Key4 is displayed properly on LCD	KEY-4	-	LCD-STAT LCD-WRITE	Y "d"	Y "d"

h. Step-8: Stop ()

4. COMPARATIVE ANALYSIS

Three methods exist in literature which can be used for generation of test cases that can be used for testing the embedded systems. The methods include generation of test cases using input domain, output domain, and generation of test cases for semi or pseudo exhaustive testing using genetic algorithms. All these methods do not take into account interrelation ships between input variables.

The association between the variables is only limited to adjacency. The methods are compared considering the testing requirements of the embedded systems and the techniques that must be used for undertaking the testing of the embedded systems. Table 7 shows comparison based on the suitability to generate the test cases that can be used for testing different features of the embedded system. From the table it can be seen that OMT is made for testing the embedded systems considering all the features that are related to the embedded systems.

Table 7. Comparison of the Test case generation methods (Combinatorial Methods) based on the suitability of the same for testing the embedded systems

S.No.	Parameter for comparison	Pseudo Exhaustive Testing	Optimal Mining Technique Procedure
1.	Test case Generation based on multiple inputs.	X	√
2.	Ability to generate test case for device testing.	X	√
3.	Ability to generate test cases related to performance testing (Throughput, response time).	X	√
4.	Ability to generate test cases that deal with internal signal processing.	X	√
5.	Ability to test external interfacing.	X	√

5. CONCLUSIONS

Optimal Mining Technique (OMT) is implemented to derive the test cases for multi-input domain embedded system such as TMCNRS. The methods used in the literature are not quite suitable for undertaking the testing of the embedded systems as they do not consider specific aspects of testing the embedded systems. Testing of the elements such as response time, throughput, testing the proper working of the devices etc. cannot be tested by the existing methods. The proposed method OMT considers all aspects of embedded system that must be tested. Experimental outcomes show that this approach can generate test cases with high competence and even fewer observable outputs from the embedded systems are considered.

REFERENCES

[1] D. R. Kuhn, *et al*., "Software Fault Interactions and Implications for Software Testing," *IEEE transactions on software engineering*, vol/issue: 30(6), 2004.
[2] D. Richard and V. Okum, "Pseudo-Exhaustive Testing for Software," *30th Annual IEEE/NASA Software Engineering WorkshopSEW-30 (SEW'06)*, 2006.
[3] M. L. Prasad and J. K. R. Sastry, "A Comprehensive Survey on Combinatorial Testing," *PONTE Journal*, vol/issue: 73(2), pp. 187-261, 2017.
[4] M. L. Prasad and J. K. R. Sastry, "A Graph Based Strategy (GBS) For Generating Test Cases Meant For Testing Embedded Systems Using Combinatorial Approaches," *Journal of Advanced Research in Dynamical and Control Systems*, vol/issue: 10(01), pp. 314-324, 2018.
[5] M. L. Prasad and J. K. R. Sastry, "Testing Embedded Systems using test cases generated through Combinatorial Methods," *International Journal of Engineering Technology*, vol/issue: 7(1), pp. 146-158, 2018.
[6] Gray, *et al.*, "The combinatorial design approach to automatic test generation," *IEEE Software*, vol/issue: 13(5), pp. 83-88, 1996.
[7] D. M. Cohen, *et al.*, "The AETG system: an approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol/issue: 23(7), pp. 437-444, 1997.
[8] D. M. Cohen, *et al*., "Method and system for automatically generating efficient test cases for systems having interacting elements," 1996.
[9] Y. W. Tung and W. S. Aldiwan, "Automating test case generation for the new generation mission software system," *IEEE Aerospace Conference*, pp. 431-437, 2000.
[10] Y. Lei and K. C. Tai, "A Test Generating Strategy for Pair-wise Testing," *IEEE Transactions on Software Engineering*, 2002.
[11] Cohen B., *et al*., "Combinatorial aspects of covering arrays," *Le Matematiche (Catania)*, vol. 58, pp. 121-167, 2004.
[12] R. Kuhn, *et al.*, "Practical Combinatorial Testing: beyond Pair wise," *IEEE Computer Society - IT Professional*, vol/issue: 10(3), 2008.

[13] J. Shiomi and S. Maruyama, “Water transport inside a single-walled carbon nano-tube driven by a temperature gradient,” *Nanotechnology*, 2009.
[14] Ochoa, *et al.*, “Box-fusion: An Approach to Enhance Pairwise Testing,” University of Texas at El Paso, 2016.
[15] M. L. Prasad and J. K. R. Sastry, “Building Test Cases by Particle Swarm Optimization (PSO) For Multi Output Domain Embedded Systems Using Combinatorial Techniques,” *Journal of Advanced Research in Dynamical and Control Systems*, 2018.
[16] M. L. Prasad and J. K. R. Sastry, “A Neural Network Based Strategy (NNBS) For Automated Construction Of Test Cases For Testing An Embedded System Using Combinatorial Techniques,” *International Journal of Engineering Technology*, vol/issue: 7(1), pp. 74-81, 2018.
[17] M. L. Prasad and J. K. R. Sastry, “Generating Test cases for Testing WEB sites through Neural Networks and Input Pairs,” *International Journal of Applied Engineering research*, vol/issue: 9(22), 2014.
[18] M. L. Prasad and V. C. Prakash, “Generation of less number of pair wise test cases using artificial neural networks (ANN),” *International Journal of Applied Engineering research*, vol/issue: 9(22), 2014.
[19] M. L. Prasad and J. K. R. Sastry, “Generation of Test Cases Using Combinatorial Methods Based Multi-Output Domain of an Embedded System through the Process of Optimal Selection,” *International Journal of Pure and Applied Mathematics*, vol/issue: 118(20), pp. 181-189, 2018.
[20] A. R. A. Alsewari, *et al.*, “Survey on input output relation based combination test data generation strategies,” *ARPN Journal of Engineering and Applied Sciences*, vol/issue: 10(18), 2015.
[21] S. Vilkomir, “Combinatorial Testing of Software with Binary Inputs: A State-of-the-Art Review,” *IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Vienna*, pp. 55-60, 2016.
[22] D. Gupta, *et al.*, “Sequence Generation of Test Case Using Pairwise Approach Methodology,” *Advances in Computer and Computational Sciences*, pp. 79-85, 2016.
[23] J. T. Jimenez, *et al.*, “A two stage algorithm for combinatorial testing,” *Optimization Letters*, vol/issue: 11(3), pp 457-469, 2017.
[24] S. Route, “Test Optimization Using Combinatorial Test Design: Real-World Experience in Deployment of Combinatorial Testing at Scale,” *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Tokyo*, pp. 278-279, 2017.
[25] P. S., M. B., M. S. Narayan and K. Rangarajan, “Building Combinatorial Test Input Model from Use Case Artefacts,” *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 220-228, 2017.
[26] Y. Yao, *et al.*, “Design and Implementation of Combinatorial Testing Tools,” *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague*, pp. 320-325, 2017.