International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8, Issue-1, May 2019

Testing Embedded Systems from Multi-Output Domain Perspective

J Sasi Bhanu, JKR Sastry, M Lakshmi Prasad

Abstract*: *The Modern embedded systems must be highly reliable and if not the products cannot even reach the market. Testing of embedded systems must be carried from multiple perspectives which include Input Domain, Output domain, Input-Output domain, Multi-Input domain, and Multi-output domain to ensure that the embedded systems are thoroughly tested. That will ensure delivery fail free embedded systems.. Sometimes the outputs generated which are either internal or external by an embedded system are interdependent and sometimes occur as a vector of outputs. Embedded systems must be tested for proper occurrence of designed outputs in the order and the sequence expected. Test cases are to be generated that will verify whether the outputs are occurring in the order and the sequence expected. In this Paper an algorithm and approach has been presented for generating test cases that when tested leads to testing embedded systems comprehensively ensuring that the outputs occur in the order and sequence expected.

Index Terms*: *Optimal selection procedure, Output domain, Multi-output domain, Testing Embedded Systems.

I. INTRODUCTION

Testing an embedded system thoroughly from different perspectives that include input-domain, out-put domain, Input-output domain, Multi-input domain and Multi-output domain are absolutely required. The test cases generated considering each domain must be consolidated to provide whole set of test cases that helps testing an embedded system comprehensively. 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 an embedded system comprehensively considering every domain will help developing defect-free embedded systems which have high market value. Testing an embedded system can be carried with the help of details known externally by following Black-Box testing Approach. In this case only functional testing of the embedded system can be carried. Testing of an embedded system can be carried by using white-Box testing considering both external and internal details of the embedded systems

Embedded systems must be tested considering hardware and firmware which are used to build the embedded systems. Testing Hardware and software however can be conducted independently and then the testing is undertaken after the software is migrated to the hardware. Hardware of an embedded system includes variety of hardware devices which are interconnected. Some of the devices include microcontrollers, application-specific integrated circuits, A/D converters, Sensors, Actuators and processors. Embedded systems are wide open spanning across several kinds applications that typically include power plant systems, networking gadgets, storage systems. The firm is the software residing in the internal memory of the embedded system will run the application for which the firmware is written. To ensure that the embedded systems is perfectly developed and is fail-free, thorough testing of the hardware and firmware needs to be done.

Combinatorial testing has been proved to be the best as it produces least number of test cases that will help in comprehensively testing any system including the embedded system. Combinatorial testing can pursue considering any of the domains of a system including the output domain, input domain, input–output domain, multi-input domain and Multi-output domain

Combinatorial testing may simply involve inputs, outputs and Input-output relationships. For a few safety critical embedded systems, building test cases drawn from output domain will be more suitable than from input domain as it guarantees that all or as many possible output combinations are comprehensively tested. Sometimes outputs from an embedded system either happen in unison or as inter-dependent vector of outputs generally called as Multi-output Vector. To arrive at the inter-dependency between the outputs, one has to consider the external outputs and internal outputs that cause the external outputs. Testing an embedded system from multi-output perspective is quite important to ensure that the internal systems are functioning quite effectively that are responsible for generating specific outputs in the order sequence required.

A pilot project has been developed for monitoring and controlling the temperatures (TMCNRS) within the Nuclear reactor systems. This system produces several outputs that are inter-dependent and the outputs must happen in proper order and sequence. In this system, operation of pumps and buzzers is independent. But the status of the pumps and the buzzers depend on the input temperatures detected by the temperature sensors. Test cases should be generated to verify the functionality of all the pumps and the buzzers and the generation of the outputs must be in-line with internally generated signals.

Revised Manuscript Received on 30 May 2019.
***** Correspondence Author
Dr. J Sasi Bhanu*, Siddhartha Institution of Technology and Sciences, Narapally, Hyderabad, Telangana.
Dr. JKR Sastry, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur District, AP, India
Dr. M Lakshmi Prasad, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur District, AP, India.

In this paper, the authors have proposed a method called Optimal Selection Procedure (OSP) that generates the test cases by choosing certain combinations of outputs based on the relationships that exists among the output variables. The optimal selection procedure has been applied to generate the test cases that help testing TMCNRS considering various outputs generated by the system that include status of pumps, and buzzers and the internal signals generated.

II. RELATED WORK

Genetic algorithms have been used by Chandra Prakash et al., [1, 2] for generating test cases in the criticality regions of embedded systems, Genetic algorithms have been used in conjunction with combinatorial methods, Output domain of an embedded system that controls temperatures within nuclear reactor has been considered and they have shown how exhaustively test cases have been generated in the criticality regions of the embedded systems. They have employed pseudo-exhaustive test case generation method to generate test cases that can find many faults at different levels. The generator developed by them considered all combinations of outputs and their relationships with Inputs. The number of test cases generated through the method proposed by them is found to be quite less compared to other methods.

Neural network based methods are in use for solving several problems including its use for generating test cases or for conducting actual testing. Ruilian Zhao et al., [3] has combined genetic algorithms and neural networks for generating test cases. They have created a function model that substitutes software. The function model identifies several computing nodes that represent hidden nodes in the neural network. Genetic algorithm is used to finding the corresponding inputs using the outputs generated through neural network model.

There is interdependence among various faults that can occur while software is in execution. It becomes difficult to test software when faults occur due to occurrence of many other faults. It has been presented by D. Richard Kuhn et al., [4] that few of the conditions built into the software causes many of the faults considering many of the domains for which software is written. Selecting combinations of inputs that can cover all kinds of faults that can occur will lead to generation of test cases that can exhaustively test software. However, the behavior of the software involves complex event sequences it is difficult to find accurate input combinations.

Covering arrays is another method that has been advanced quite rapidly. Two methods have been proposed by D. Richard et al., [5] that include covering arrays that represents interactions among the input variables and the method model checking using which the test cases are generated. They have tested the method on a traffic collision avoidance system. They have considered 6 way interactions among the input variables. Several tools existed for generating combinatorial test cases. R. Kuhn et al. [6] have presented a commission of various methods using real-world applications. They have presented empirical analysis of the methods.

Breeding test cases through genetic algorithms have been presented by D. Berndt, J. Fisher et al., [7]. The test cases can be breaded based on severity, novelty and proximity. A fitness function has been presented that reveal the behavior of microorganisms. They have presented the technique through a case study and using several visualization techniques using which the behavior is analyzed using fossil records.

Branch and find fault method along with genetic algorithm has been used for generating the test cases by B. F. Jones et al., [8]. The test cases generated by that method are fewer in number when compared to ransom testing. The methods also cover entire software. However, the computation effort in this case is very high. The test cases generated are very close to the sub-domains of inputs. Genetic algorithms have been used for exploiting the faults that can be caused when branching takes place within the software execution.

Adequate testing has to be carried to ensure entire coverage of the software requiring more number of test cases. Automated tools are required to generate adequate number of test cases. One such tool has been presented by Kamal Zuhairi Zamli et al [9]. The tool presented them uses combinatorial approach for generating the test cases. The have considered many combinations of inputs, environments on which the hardware and software is used and many of the system conditions for generating test cases through combinatorial methods. The conformance of the test cases against the system specification has also been shown by the authors.

A testing complex system is a challenge. A kind of system is required for testing software systems. Once unit testing is carried, building on it generates test cases for carrying system level testing is possible by suing the genetic algorithms. The test cases used for unit testing can be used as test case generation seeds. Watkins et al [10] have presented vocabulary of attributes and its related framework for generating test cases. The have also presented some visualization techniques using which one can find the system failures right in the Initial stages.

Pair wise testing method is quite frequently used combinatorial method for generating test cases. Huge number of test pairs gets created when number input variables are quite high. Meta- heuristic search techniques are required to find the pairs that are minimum and lead optimum number of test cases. Xiang Chen et al., [11] have used PSO (Particle swarm optimization) algorithm to find optimum input pairs. They have presented many algorithms based on PSO to systematically generate all the test cases. They have used two strategies that include IPO-Like and the other one-at-a-time strategy. In both these algorithms PSO is used for generating the test cases. They have also presented a method using which the search space can be defined.

PSO also has been used by B.S. Ahmed and K.Z. Zamli [12] for generating test cases using a t-way strategy. They compared test size reduction achieved by using this method against other similar methods presented in the literature. Some improved algorithms in similar lines have been presented by Jiang Shouda [13].

Particle Swarm techniques is one moth important technique that is being used extensively for implementing various applications. A t-way testing method based on particle swarm method has been proposed by K. Rabbi, et al., [14]. They have implemented a testing strategy for generating optimum number of test cases. They have also presented a comparative analysis and shown that particle swarm method produced very few but effective number of test case.

Several strategies exists for carrying combinatorial testing, A survey has been conducted Abdul Rahman [15] and presented several test strategies that considers the Input-output domain if an embedded system. They have compared all the available algorithms with nature inspired algorithm considering all input-output relations. They have presented that combinatorial testing methods can also be used for testing software that uses extensively logical and Boolean expressions. Testing software that is predominantly developed using Boolean expressions has been presented by S. Vilkomir [16],

A pair-wise test case generation method has been used by Deepa Gupta [17]. The have shown how test cases can be generated considering the input sequences. The have shown how input sequences can be explored for generation of test cases. This approach makes sure the generation of test cases that are required all relations between every pair of instructions at least once.

A two stage annealing algorithm that has been simulated has been presented by Jose Torres-Jimenez et al., [18]. The algorithm constructs the ternary covering arrays that help generating the test cases which can be used for testing the embedded systems comprehensively. Combinatorial test data generation method has been presented that focusses on process, approach and constraints by S. Route [19]. They have employed IBM developed tool for generating the test cases based on the combinations of the inputs and optimization of test cases and to reduce the effort required to comprehensively test embedded systems.

Use case artifacts are the best models that can be used for generating the test cases based on combinatorial methods. A rule based model has been used for finding the input domain for use case models P. S et al., [20]. They have implemented two parsers that are natural language and XML based. The rules that are formulated have been used for generating the output domain of a given system.

Several testing tools have been developed by Y. Yao et al., [21 based on combinatorial methods. The have considered several testing scenarios comprising of industrial, and academic domains. They have developed several testing tools that deal with test case optimization, test case generation etc. The tools developed by them are scalable, configurable and modular.

Many contributions have been presented for testing the software [22][23][24][25][26][27][28][29][30][31][32] but most of the contributions have been focused at input perspective only.

III. PILOT EMBEDDED SYSTEM

A prototype model has been developed for monitoring controlling the temperatures within a nuclear reactor system. The system is famously called as TMCNRS system. The Block diagram of the system is shown in 1. Temperatures within the nuclear reactor tubes are measured through the sensors that are mounted on to the tubes. The analog signals synonymous to the temperatures sensed are sent to an A/D converter after amplifying the same through operational amplifiers. The temperatures within the nuclear reactors are read through firmware that has been installed within a microcontroller system to which A/D converter has been connected. The temperatures read are stored in the RAM in digital form.

Two pumps are interface to the Micro controller system through relays using which the pump start stop operations are controlled. The pumps when started will pump a colorant into a nuclear reactor so that temperatures with the reactor tubes will be brought down to the reference temperatures. The pumps are made to pump the coolant into the tubes when the sensed temperatures are more than the reference temperatures. A buzzer is connected to the Micro Controller which will be triggered when absolute of difference two sensed temperatures is > 2. Maintenance of proper temperatures gradients is one of the most important crucial issues for proper enrichment of uranium. The Micro controller is connected to a PC through RS232C interface for feeding the reference temperatures

The nuclear reactor system is a safety critical system. The system is built through authentication and authorizations system. The passwords are fed through a keyboard connected to the Micro controller system. The microcontroller based system is connected with an LCD for displaying messages that show the status of process taking place within the nuclear reactor system. The alerts to the operator are also displayed through LCD. The functional requirements of the TMCNRS system are shown in Table I. Table II shows the relationships among the Input-output variables.

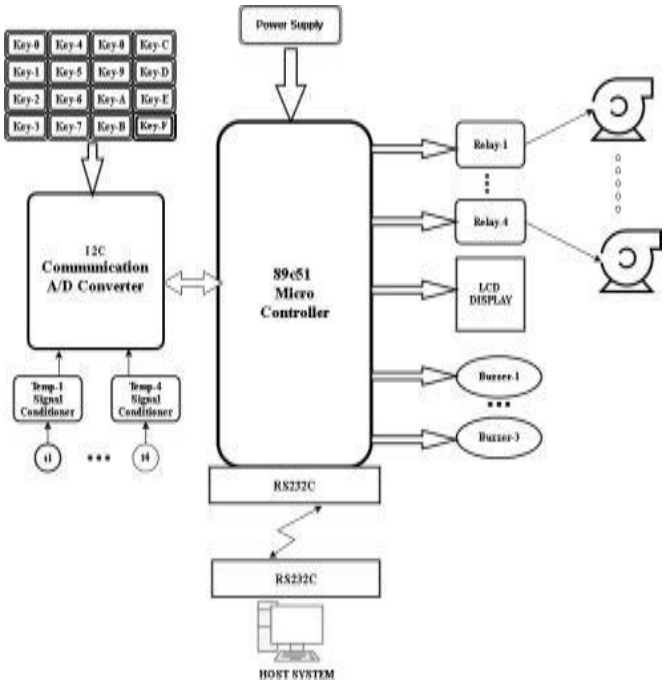

Fig. 1 System Integration Diagram of TMCNRS

Table I. Requirement condition of TMCNRS

S. No.	Functional requirements
Req 1	Must be able to read the reference temperatures through a PC connected to the Micro Controller through an RS232C interface

Req. 2	Within every 10 Milli secs Temp-1 must be sensed, sent to HOST and displaced on LCD. PUMP-1 must be made to run when TEMP1- > reference temp-1 or PUMP-1 must be stopped
Req. 3	Within every 10 Milli secs Temp-2 must be sensed, sent to HOST and displaced on LCD. PUMP-2 must be made to run when TEMP2- > reference temp-1 or PUMP-1 must be stopped
Req. 4	If the absolute difference between the sensed temperatures > 2 then the buzzer must be triggered or lease the buzzer must be stopped

Table II Relationship between input parameters and status of Pump and buzzer outputs

Temp. sensed by Sensor 1 (in ^oC)	Temp. sensed by Sensor 2 (in ^oC)	Expected Status of Pump1 Ref=30^o C	Expected Status of Pump2 Ref=32^{o}C	Expected Status of Buzzer1
30	31	0 (Pump OFF)	0 (Pump OFF)	0 (Buzzer OFF)
20	33	0 (Pump OFF)	1 (Pump ON)	1 (Buzzer ON)
33	30	1 (Pump ON)	0 (Pump OFF)	1 (Buzzer ON)
32	33	1 (Pump ON)	1 (Pump ON)	0 (Buzzer OFF)

IV. GENERATION OF TEST CASES FOR TMCNRS USING OPTIMAL SELECTION PROCEDURE

The first step in the selection procedure is to find Output Domain. In the current scenario, we have two output domains that have to be covered, one is Pump Status Code (PSC) i.e. in the above system OSC is pump and another is Buzzer Status Code (BSC).

As mentioned in [1], 28 test cases can be generated to cover all the BSC codes and 27 test cases can be generated to cover all the OSC codes for 8 sensors and 7 buzzer TMCNRS system. Authors in [1] suggested pseudo-exhaustive testing wherein, test cases are generated to cover all OSC codes and then test cases are generated to cover all BSC codes. So, total of 28+27 test cases are required to cover all possible outputs in multi output system. Algorithm 1 relates to covering all possible BSC codes, and Algorithm 2 to cover all possible PSC codes.

Algorithm 1:

This algorithm explains how to derive the test case from the given BSC code. Assuming buzzer code will be 1 when absolute difference of consecutive temperature sensors is more than 2 (assumed Gradient value), otherwise 0.

Let BSC be a vector containing n-1 binary values each representing a set of bits (b1, b2, b3…bn-1) where bi (i=1, 2… (n- 1)), takes value either 0 or 1.

Let PSC be a vector containing n binary values, each represented by a set of bits showing the status of the pumps related to specific temperatures ti (i=1, 2 …n).

As the buzzer output depends on the absolute difference of adjacent temperatures, Let the first temperature value is selected randomly from 1 to 99 and second value ti ($2 \le I \le n-1$) is derived from the following equation. First temperature is defined as t1= random (1, 99), then the temperatures ti (i= 2 …n) is calculated as

ti-1 + Random [-2, 2] if bi= 0

Random [1, 99] such that |Random [1, 99] - ti-1| > 2 if bi= 1 for $2 \le I \le n$ (1)

Initialize BSC array = {0, 1, 2, 3…, (2n-1-1)} of a multi output embedded system each value representing a binary value

While (No. of BSC elements) > 0

{

Get a BSC code from BSC array.

Generate the test case from the above BSC code using equation 1.

Remove the selected BSC code from the BSC array. Decrement the number of BSC elements.

}

Algorithm-2:

This algorithm tells how to derive the test case from the given PSC code based on the reference temperature inputs.

Let (Ref1, Ref2, and Ref3…Refn) be the Reference temperature of the sensors used in the system.

Let (p1, p2, p3…pn) be the given PSC code, where pi (i=1, 2...n) takes value either 0 or 1. (t1, t2, t3…tn) be the test vector being generated from the given PSC code and reference temperatures.

Random [1, Refi] if pi= 0

ti =Random [Refi+1, 99] if pi= for 1 <= I <=n

Initialize OSC array = {0, 1, 2, 3..., (2n-1)} of a multi output embedded system

While (no. of OSC elements >0)

{

Get an OSC code from OSC array.

Generate the test case from the above OSC code using equation 2.

Remove the selected OSC code from the OSC array. Decrement the number of OSC elements

}

An optimization technique has been proposed [1] which uses union of two sets and getting a set that is much smaller compared to other combinations I.E. 256 <= SIZE OF THE RESULTANT SET <= 28+27. Genetic algorithms have been used [1,2] for generation test cases. This approach generates test cases for covering the output domain and then uses the same generating the test cases based on input domain.

In similar lines, we proposed yet another algorithm which derives test cases that covers all possible BSC and OSC codes together till all BSC codes are covered, by properly choosing the OSC code and BSC code based on Feasibility Table as shown in table 3. Test cases for the left over OSC codes are generated using Pseudo code 2.

Algorithm 3:

This algorithm tells how to derive the test case from the given OSC code and BSC code based on the reference temperature inputs

Step 1:

Let (Ref1, Ref2, Ref3…Refn) be the Reference temperature of the sensors used in the system.

Let (p1, p2, p3…pn) be the given PSC code, where pi (i=1, 2...n) takes value either 0 or 1. Assuming buzzer code will be 1 when absolute difference of consecutive temperature sensors is more than 2, otherwise 0.

Let (b1, b2, b3…bn-1) be the given BSC code, where bi (i=1, 2, (n- 1)) takes value either 0 or 1. (t1, t2, t3…tn) be the test vector being generated from the given PSC code and reference temperatures.

Before proceeding to generate the test case using BSC code and PSC code together, we should check the feasibility of the existence of these two codes first from Table III. Random [x, y] generates the random number between x and y. This is achieved in the program using rand () function as x+ rand () % (y-x+1).

TABLE III FEASIBILITY CHECK TABLE

FEASIBILITY CHECK:

p_i	p_{i+1}	b_i	Feasibility	Formula
0	0	0	✓	t_1=Random[1,Ref_1] $t_{i+1}=t_i$ + Random[-2,2] such that $t_{i+1} < Ref_{i+1}$ for $1 \le i \le n$
0	0	1	✓	t_1=Random[1,Ref_1] t_{i+1}= Random[1,Ref_{i+1}] such that $\|t_i - t_{i+1}\| > 2$ for $1 \le i \le n$
0	1	0	X	Not Feasible
0	1	1	✓	t_1=Random[1,Ref_1] t_{i+1}= Random[Ref_{i+1} +1,99] such that $\|t_i - t_{i+1}\| > 2$ for $1 \le i \le n$
1	0	0	X	Not Feasible
1	0	1	✓	t_1=Random[Ref_1+1,,99] t_{i+1}= Random[1,Ref_{i+1}]such that $\|t_i - t_{i+1}\| > 2$ for $1 \le i \le n$
1	1	0	✓	t_1=Random[Ref_1+1,,99] $t_{i+1}=t_i$+ Random(-2,2) such that $t_{i+1} > Ref_{i+1}$ for $1 \le i \le n$
1	1	1	✓	t_1=Random[Ref_1+1,,99] t_{i+1}= Random[Ref_{i+1},+1, 99]such that $\|t_i - t_{i+1}\| > 2$ for $1 \le i \le n$

Let us consider an example of PSC code = (0,1,0,0) and BSC code = (0,0,0) and Reference temperatures as (30,32,34,36). First find temperature t1 using the equation (2). t1= Random [1, Ref1]; say 30 Now let us calculate t2

From equation (1) to satisfy BSC code, t2= t1+ Random [-2, 2] i.e. 30±2

From equation (2) to satisfy OSC code, t2 = Random [Ref2+1, 99] i.e. Random [32+1, 99]

Now we are left with two solutions for t2 which doesn't co-exist together.

This kind of selection of PSC code and BSC code together is not possible. i.e. pi =0; pi+1 = 1 and bi=0. Table 3 tells about the feasibility of the possible values of pi and bi.

PSC code (p1, p2, p3, p4) and BSC code (b1, b2, b3) is said to feasible: If pi, pi+1 and bi satisfies the above table for each i = 1,...n-1 (3)

Step-2:

Optimal Selection of test case using Feasibility Table. In this section, we will discuss about how to derive the test vector i.e. (t1, t2, t3…tn) satisfying both the OSC and BSC code.

Let us consider an example of PSC code = (1, 0, 0, 1) and BSC code = (1, 0, 1) which means pumps p1, p4 should be ON and p2, p3should be OFF, and the buzzers b1, b3 should be ON and b2should be OFF.

We can observe that the selected PSC code and BSC code is feasible from equation (3). For the given example, p1=1, p2=0 and b1=1 and from the feasibility table, we can get t1 and t2 as below.

t1=Random [Ref1+1, 99]

t2=t 1+ Random [1, Ref2] such that |t1-t 2| > 2 t1 and t2 are derived now, move to next

values in OSC code i.e. p2 = 0, p3=0 and b2=0. Now let's proceed to derive t3. We proceed to derive t3 directly from the table t3=t 2+ Random [-2, 2] such that t 3 ≤ Ref3.Next step is to derive t4, from provided input of p3=0, p4=1 and b3=1, t4= Random [Ref4+1, 99] such that |t4-t3| > 2.

In this same manner we derive the test case for any number of temperature sensors and for given PSC and BSC code as long as the feasibility is met.

Main Algorithm:

Initialize PSC array = {0, 1, 2, 3..., (2n-1)}
Initialize BSC array = {0, 1, 2, 3…, (2n-1-1)

While (no. of BSC elements>0)
{
For each BSC code, find a PSC code which is feasible based on feasibility table.
Use the above BSC code and PSC code generate the test case using Algorithm 3.
Remove the BSC code from BSC array Decrement the no. of BSC elements by 1 Remove the PSC code from PSC array Decrement the no. of PSC elements by 1
}
While (no. of PSC elements >0)
{
Get a PSC code from PSC array.
Generate the test case from the above PSC code using equation 2.
Remove the selected PSC code from the PSC array. Decrement the no. of PSC elements.
}

TABLE IV OPTIMIZED TEST SUITE GENERATED FROM MULTI OUTPUT DOMAIN OF TMCRS USING OSP

Test case No.	Input Vector								Expected Output	
	Temp1 (Ref. Temp= 30)	*Temp 2 (Ref. Temp= 32)*	*Temp 3 (Ref. Temp= 34)*	*Temp4 (Ref. Temp= 36)*	*Temp5 (Ref. Temp= 38)*	*Temp6 (Ref. Temp= 40)*	*Temp7 (Ref. Temp= 42)*	*Temp8 (Ref. Temp= 44)*	PSC (Binary)	BSC (Binary)
1.	12	12	14	12	14	16	17	18	00000000	00000000
2.	23	25	23	21	20	20	19	96	00000001	00000001
3.	26	26	26	25	24	26	69	70	00000011	00000010
…	…	…	…	…	…	…	…	…	…	…
254.	73	54	51	89	56	53	16	67	11111101	11111111
255.	97	57	90	85	56	90	98	39	11111110	11111111
256.	83	58	71	50	42	64	87	61	11111111	11111111

With the first part of algorithm, we get the test cases that coverall the possible BSC codes (2n-1) along with partial coverage of PSC codes (2n-1). And we will be left with another 2n-1 PSC codes in the PSC array. Second part of the algorithm derives test case from the leftover PSC codes from the above step. For simplicity, the PSC codes and BSC codes in the respective arrays are denoted in decimal form rather than in the binary form.

V. EXPERIMENTAL RESULTS

In this paper Optimal Selection Procedure (OSP) is implemented to derive the test cases for multi output domain that can be considered with respect to an embedded system. The optimized test suit derived from the multi output domain of TMCNRS with a configuration of 4- Sensors, 4-Pumps, 3-buzzers using optimal selection procedure is shown in Table IV. The optimized test suit derived for TMCNRS with a configuration of 8-Sensors, 8- Pumps, 7-buzzers using optimal selection procedure is shown in Table V.

In order to calculate the effectiveness of this approach, the results have been compared with the results generated by the Genetic algorithm [1] and particle swarm optimization technique also. This approach yields to produce a less number of test cases with high efficiency in terms of time when compared to the previous existing technique i.e. genetic algorithms and particle swarm optimization as shown in Table VI. Execution time for various input parameters using 64 bit windows system is shown in Table VII.

TABLE V OPTIMIZED TEST SUITE GENERATED FROM MULTI –OUTPUT DOMAIN OF TMCRS

Test case No.	Input Vector				Expected Output	
	Temp1 (Ref.Temp=30)	*Temp 2 (Ref.Temp=32)*	*Temp 3 (Ref.Temp=34)*	*Temp4 (Ref.Temp=36)*	OSC (Binary)	BSC (Binary)
1.	12	12	14	12	0000	000
2.	30	31	32	98	0001	001
3.	15	13	35	37	0011	010
4.	3	4	67	35	0010	011
5.	22	56	57	55	0111	100
6.	18	51	50	19	0110	101
7.	10	46	32	34	0100	110
8	26	78	8	97	0101	111
9.	96	30	28	21	1000	101
10.	61	32	34	38	1001	101
11.	48	29	67	34	1010	111
12.	58	28	48	76	1011	111
13.	98	74	9	12	1100	111
14.	72	66	17	92	1101	111
15.	32	64	59	5	1110	111
16.	84	53	65	88	1111	111

VI. CONCLUSION

Input domain and output domain based combinatorial testing using combinations of inputs and outputs have been proved to be effective methods when combined with genetic algorithms or heuristic search methods. There are many applications that produce clusters of output called Multi- output domain. The output domain as such can be recognized as clusters of output vectors, each vector having common characteristics. In such situations Optimal Selection Procedure presented in this paper will generate most effective test cases that cover 100% of the source code. In the example used for showing the effectiveness of the procedure, two output vectors have been chosen. While one output vector shows the status of pumps, the second vector deals with the status of buzzers.

VII. ASSOCIATION BETWEEN PSEUDO EXHAUSTIVE TESTING USING GA, PSO AND OPTIMAL SELECTION PROCEDURE

S.No	Type of Testing	No of test case required			
		For 4-Pumps 3-Buzzers Config.	*For 6-Pumps 5-Buzzers Config.*	*For 8-Pumps 7-Buzzers Config.*	*For 10-Pumps 9-Buzzers Config.*
1.	Exhaustive Testing of input Domain	96,059,601	$9.4x10^{11}$	$9.2x10^{15}$	$9.0x10^{19}$
2.	Exhaustive Testing of Output Domain	128	2048	32,768	5,24,288
3.	Pseudo Exhaustive Testing using Genetic Algo. Ref[1]	16	64	259	1260
4.	Particle Swarm Optimization (PSO)	16	64	259	1026
5.	Optimal Selection Procedure (OSP)	**16**	**64**	**256**	**1024**

TABLE VII. EFFICIENCY OF OSP WITH RESPECT TO EXECUTION TIME (SECONDS)

S.No	Name of the Technique	*For 4-Pumps 3-Buzzers Config.*	*For 6-Pumps 5-Buzzers Config.*	*For 8-Pumps 7-Buzzers Config.*	*For 10-Pumps 9-Buzzers Config.*
1.	Genetic Algorithms	0.007	0.75	5.09	119.91
2.	Particle Swarm Optimization (PSO)	0.004	0.216	3.18	22.35
3.	**Optimal Selection Procedure (OSP)**	**0.004**	**0.013**	**0.076**	**0.476**

REFERENCES

1. C. P. Vudatha, S. Nalliboena, S. K. Jammalamadaka, B. K. K. Duvvuri and L. S. S. Reddy, "Automated generation of test cases from output domain of an embedded system using Genetic algorithms," 2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, pp. 216-220, 2011.
2. C. P. Vudatha, S. Nalliboena, S. K. Jammalamadaka, B. K. K. Duvvuri and L. S. S. Reddy, "Automated generation of test cases from output domain and critical regions of embedded systems using genetic algorithms," 2nd National Conference on Emerging Trends and Applications in Computer Science, Shillong, pp. 1-6, 2011.
3. Ruilian Zhao et al, "Neural-Network Based Test Cases Generation Using Genetic Algorithm,"13th IEEE International Symposium on Pacific Rim Dependable Computing, pp.97-100, 2008.
4. D. Richard Kuhn et al, "Software Fault Interactions and Implications for Software Testing," IEEE transactions on software engineering, Vol.30, No.6, June 2004.
5. D. Richard et al, "Pseudo-Exhaustive Testing for Software", 30th Annual IEEE/NASA Software Engineering WorkshopSEW-30 (SEW'06), 2006.
6. R. Kuhn et al. "Practical Combinatorial Testing: beyond Pair wise", IEEE Computer Society - IT Professional, Vol. 10, No. 3, May/June 2008.
7. D. Berndt, J. Fisher et al, "Breeding Software Test Cases with Genetic Algorithms", IEEE Proceedings of the 36th Hawaii International Conference on System Sciences, 2003.
8. B. F. Jones et al, "A Strategy for using Genetic Algorithms to Automate Branch and Fault-based Testing", The Computer Journal, Vol. 41, No. 2, 1998.
9. Kamal Zuhairi Zamli et al, "A Tool for Automated Test Data Generation (and Execution) Based on Combinatorial Approach", International Journal of Software Engineering and Its Applications Vol. 1, No. 1, July, 2007.
10. Watkins et al, "Breeding Software Test Cases for Complex Systems", IEEE Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.
11. Xiang Chen, Qing Gu, Jingxian Qi and Daoxu Chen, "Applying Particle Swarm Optimization to Pairwise Testing," COMPSAC, 2010 IEEE 34th Annual Computer Software and Applications Conference, pp. no. 107- 116, 19-23 July 2010.
12. B.S. Ahmed and K.Z. Zamli, "PSTG: A T-Way Strategy Adopting Particle Swarm Optimization," AMS, 2010 Fourth Asia International Conference on Mathematical/Analytical Modeling and Computer Simulation, pp.no.1-5, 26-28 May 2010
13. Wang Jian Feng and Jiang Shouda, "An Improved Algorithm for Test Data Generation Based on Particle Swarm Optimization," In Proceedings of the 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC '11). IEEE Computer Society, Washington, DC, USA, pp.no.404-407, 2011.
14. K. Rabbi, Q. Mamun and M. R. Islam, "An efficient particle swarm intelligence based strategy to generate optimum test data in t-way testing," IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, 2015, pp. 123-128, 2015.
15. Abdul Rahman A. Alsewari, Nasser M. Tairan and Kamal Z. Zamli, "Survey on input output relation based combination test data generation strategies," ARPN Journal of Engineering and Applied Sciences vol. 10, no.18, October 2015.
16. 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, 2016, pp. 55-60, 2016.
17. Deepa Gupta, Ajay Rana, Sanjay Tyagi, "Sequence Generation of Test Case Using Pairwise Approach Methodology," Advances in Computer and Computational Sciences, pp 79-85, 2016.
18. Jose Torres-Jimenez, Himer Avila-George, Idelfonso Izquierdo- Marquez, "A two stage algorithm for combinatorial testing," Optimization Letters, Volume 11, Issue 3, pp 457–469, March 2017.
19. 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.
20. 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.
21. Y. Yao, Y. Yan, Z. Wang and C. Liu, "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
22. Patil, R., Prakash, V.C., Neural network based approach for improving combinatorial coverage in combinatorial testing approach, Journal of Theoretical and Applied Information Technology, 96(20), pp. 6677-6687
23. Srinivas, K., Mohammed Ismail, B., Test case prioritization with special emphasis on automation testing using hybrid framework, Journal of Theoretical and Applied Information Technology, 96(13), pp. 4180-4190
24. Kondhalkar, V.V., Chandrapraksh, V., Automated generation of test cases for conducting pairwise plus testing, Journal of Advanced Research in Dynamical and Control Systems, 10(7 Special Issue), pp. 1484-1492
25. Chandra Prakash, V., Tatale, S., Kondhalkar, V., Bewoor, L., A critical review on automated test case generation for conducting combinatorial testing using particle swarm optimization, International Journal of Engineering and Technology(UAE), 7(3), pp. 22-28
26. Siva Prasad, K., Raja Sekhar, K., Rajarajeswari, P., , An integrated approach towards vulnerability assessment & penetration testing for a web application, International Journal of Engineering and Technology(UAE), 7(2.32 Special Issue 32), pp. 431-435

27. Maddipati, S.S., Srinivas, M., Statistical testing on prediction of software defects EAI Endorsed Transactions on Energy Web, 5(20), pp. 1-6
28. Prasanth, Y., Bharathi, N.V., Tejaswini, B.S., Krishna, P.S., Analysis and prediction of defects using reliability growth models, International Journal of Civil Engineering and Technology, 9(1), pp. 22-27
29. Shahabuddin, S.M., Yalla, P. , Impact of lean software development into agile process model with integration testing prior to unit testing, Journal of Theoretical and Applied Information Technology, 95(22), pp. 6163-6175
30. Jammalamadaka, K., Ramakrishna, V., Test case selection using logistic regression prediction model, International Journal of Mechanical Engineering and Technology, 8(11), pp. 786-796
31. Shahabuddin, S.M., Prasanth, Y., Integration testing prior to unit testing: A paradigm shift in object oriented software testing of agile software engineering, Indian Journal of Science and Technology, 9(20),91223
32. Pasala, S., Prasad, M.S.R., Rama Krishna, V., Test case prioritization on real time applications, Journal of Theoretical and Applied Information Technology, 87(2), pp. 237-249

AUTHORS PROFILE

Dr. J Sasi Bhanu is a Professor working at Siddhartha Institute of Technology and Sciences Narapally, Hyderabad She is B.TECH, M.TECH and Ph. D from Computer Science and Engineering, JNTU Kakinada. She has been specializing in the field of Embedded Systems, Web Technologies and Software Engineering. She has to her credit more than 45 papers published in reputed journals which are Scopus and SCI Indexed

Dr. JKR Sastry a double doctorate in CSE, and Management and works for KLEF University as a Professor. He has published more than 245 papers which are indexed into Scopus, SCI and Google

Dr. M. Lakshmi Prasad is a recent Ph. D from KLEF University and specializes in the area of Combitorial testing of Embedded Systems.