International Journal of Engineering & Technology, 7 (2.7) (2018) 146-158

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Testing embedded systems using test cases generated through combinatorial techniques

Lakshmi Prasad Mudarakola[1]*, J. K.R. Sastry[2], V. Chandra Prakash[3]

[1]Ph.D Scholar, Dept. of Computer Science & Engineering, NBKR Institute of Science and Technology, Vidyanagar, Nellore, AP, India.

[2] Dept. of Electronics and Computer Science Engineering,Koneru Lakshmaiah Educational Foundation, Vaddeswram, Guntur District, AP, India.

[3]Dept. of Computer Science Engineering, Koneru Lakshmaiah Educational Foundation,Vaddeswram, Guntur District, AP, India

**Corresponding author E-mail: prasad.hinduniv@gmail.com.*

Abstract

Thorough testing of embedded systems is required especially when the systems are related to monitoring and controlling the mission critical and safety critical systems. The embedded systems must be tested comprehensively which include testing hardware, software and both together. Embedded systems are highly intelligent devices that are infiltrating our daily lives such as the mobile in your pocket, and wireless infrastructure behind it, routers, home theatre system, the air traffic control station etc. Software now makes up 90% of the value of these devices. In this paper, authors present different methods to test an embedded system using test cases generated through combinatorial techniques. The experimental results for testing a TMCNRS (Temperature Monitoring and Controlling Nuclear Reactor System) using test cases generated from combinatorial methods are also shown.

Keywords: *Combinatorial Test Methods; Embedded Systems; Software Testing; Test Case Generation; Clean Room Software Engineering.*

1. Introduction

This Generation of test cases which are required for testing the embedded system comprehensively considering hardware, software and both has been presented through different combinatorial methods especially considering input domain, output domain, input-output domain and multi-output domain. These test cases are to be used for testing the embedded system and finding out the reliability of such a system based on the test results.

The testing of the embedded system as such must be undertaken at HOST (remote PC), Target (Embedded Systems) and considering both hardware and software together. The test process as such is initiated from the HOST. The test results obtained after undertaking testing are stored on the HOST and based on the results reliability of the software is assessed.

Testing of an embedded system can be viewed in terms of testing Hardware, Hardware independent code, and hardware dependent code. Hardware can be tested using the TARGET (Embedded system) with the test cases initiated form a HOST (PC) which is interfaced to TARGET generally through RS232C or USB interface. Hardware independent code can be tested completely on a HOST with hardware dependent code commented. Hardware dependent code can be tested using both TARGET and HOST.

Several methods are in use for undertaking testing of embedded systems. The methods that include Scaffolding, Assert macros and Instruction set simulators can be used for undertaking testing of Hardware independent code on the HOST. Logic Analyzers can be used for testing hardware. The hardware dependent code can be tested using in-circuit emulators and monitors considering both Target and HOST.

The environment required for undertaking testing must be set both at Target and HOST before commencing the actual process of undertaking the testing, For testing embedded systems several methods are in use which include scaffolding, instruction set simulator, In-Circuit emulator, processor that implements assert macros, monitors and many third party tools.

Hardware devices like Oscilloscopes and Logic Analyzers are used for undertaking the testing of the hardware. The test environment that can be used for testing a single standalone embedded systems is shown in the Figure 1.

The test gadgets are connected to the target embedded system through probes. The entire testing process is invoked from HOST through a test process. Test manager can invoke the test process through user interface provided at the HOST side. The test process will initiate a process to be undertaken at HOST or TARGET or HOST + TARGET based on the type of the test case that must be tested.

Four methods are used for undertaking testing at HOST which includes Scaffolding, Assert macros, Third party tools and Instruction Set Simulators. The method logic analysis is used for undertaking testing of the hardware through commands initiated from the HOST. In circuit emulators and monitors are used for undertaking the testing at the target with complete process of testing initiated from the HOST. Actual testing to be done is intuited from a HOST.

Testing is undertaken through a method that may involve Target (If testing hardware), or may involve HOST (If software is to be tested) or may involve (If testing is to be done using Hardware and Software. The process of undertaking testing at Target, HOST and

both using the related method for undertaking testing is shown in figure2.

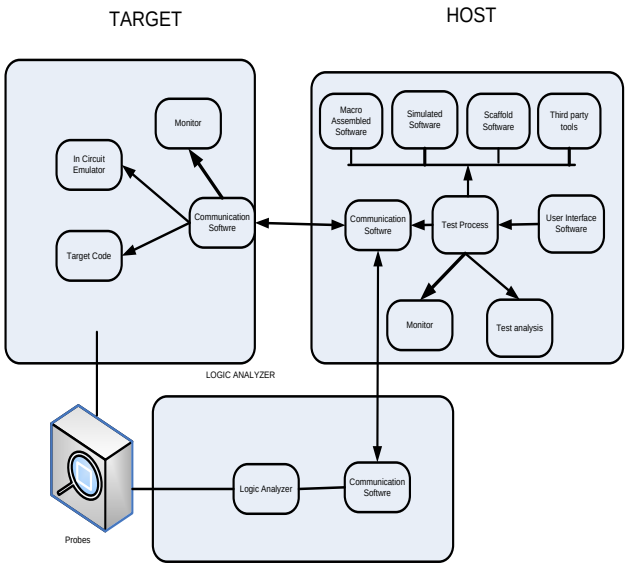

Fig. 1: Environment setting for testing embedded systems

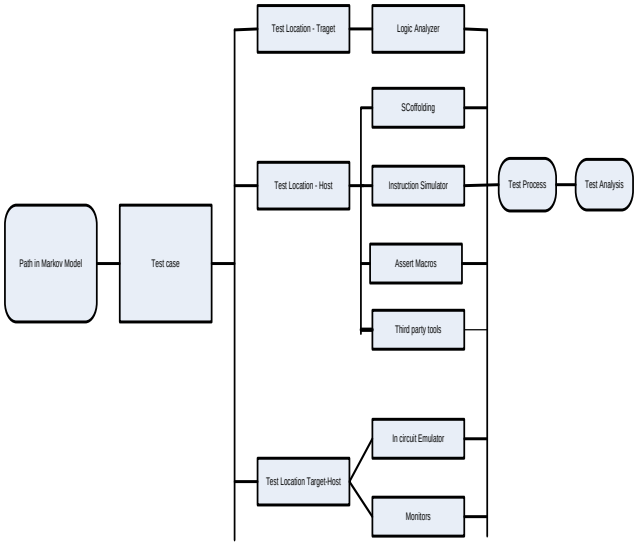

Fig. 2: Process flow for undertaking the testing

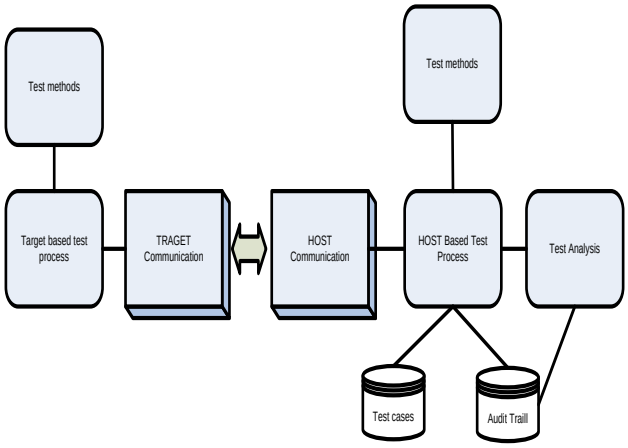

Fig. 3: Test process flow

All the generated test cases using the combinatorial methods are accumulated at the HOST and classified according to the Test Method. Testing is undertaken by using a test method one after the other by picking test cases from a database and the test results are written to an audit trail which is used for analyzing and Certification. Figure 3 shows the test process flow.

The generated test cases are stored in a database and the test cases are ordered as per the test case method. The test process resident on the HOST reads the test cases one after the other from the database file and facilitates the execution of the same by following suitable process flow and the results obtained out of the testing process are updated into an audit trail.

2. Literature Survey

It will be disastrous if the embedded systems meant for monitoring and controlling the mission critical and safety critical systems fails for any reason. There should not be any chance for failures to occur in such a systems [1]. Through testing must be done in that case especially in the critical areas of the embedded systems. During the process of the development of the embedded systems inspections, reviews, walkthrough, verification and validation must be carried so that the bugs if exists any will be traced early in the life cycles of the embedded systems [2].

Testing plays very important in the development of high quality embedded systems. The intermittent stage and processes are thoroughly checked to find whether the requirements projects by the user are actually met. After completing the design and development of hardware and software, integration of both is undertaken and testing is undertaken through in circuit emulators and monitors [3].

Majority of the testing can be carried on a HOST where the firmware is actually developed. 80% of the firmware is hardware independent and therefore can be tested on the HOST where is actually developed. Generating the most wanted a test case that actually tests every important part of the embedded system. Different kinds of strategies must be adapted such that all the required test cases can be generated. Many methods are to be used for testing the embedded systems. The test cases should be mapped to the methods which should be used for undertaking testing. Historical evidence is required for mapping the test cases to testing methods [4].

The comprehensive testing of the embedded system involves the usage of several methods, tools, techniques and locations [5] have proposed a test process architecture model for undertaking the integrated testing that will help in undertaking the comprehensive testing of the embedded system.

CRSE has not provided any direct methods using which testing of the embedded systems are undertaken, However some authors have used some of the models stated by CRSE for undertaking the testing of embedded systems. [6] have used statistical testing method for undertaking the testing.

The UML related use case models and statistical methods can be used for generating the test cases that could be used for testing embedded systems [7]. These methods do not support generation of test cases for real-time systems. No control logic as such is used when the test cases are generated using the usage models.

Rajasekhra Rao et al., [8] have recommended several models for undertaking the comprehensive testing of the embedded systems. He has recommended that Comprehensive testing of the embedded system is a necessity when testing of Hardware, software and both has to be undertaken using several distinct methods at different testing locations.

Chandra Prakash[9] has explained the way testing of the embedded systems can be carried using clean room software engineering methodology into which the models built by Rajasekhra Rao which are meant for testing standalone embedded systems have been inbuilt.

Kamesh DBK [10] has proposed several new models for undertaking testing of the embedded systems which are built into the testing path of clean room software engineering methodology.

3. Mapping Test cases Generated by Combinatorial Methods to Test Methods

It has been shown the way test cases are generated considering the input domain, output domain, multi- output domain, and input-output domain in the previous chapters. The test cases must be mapped to the kind of testing method that must be used for undertaking testing.

A testing method dictates the location where the testing must be carried. The type of testing method to be used is dependent on the kind of testing that must be carried. Standard list of test cases that are normally used for undertaking testing of standalone embedded systems can be maintained. The type of method that must be used for undertaking testing of an embedded system using standard test cases can be pre-identified.

Various types of test methods can be used for undertaking testing of a stand-alone embedded system that include scaffolding, assert macros, Instruction set simulation, in-circuit emulation, monitors and Logic Analyzers.

Firmware can be tested using the methods scaffolding and, instruction set simulators. The assert macro method can be used for testing existence of proper environment for making the ES application run in proper manner.

The method Logic analyzer can be used for testing the hardware while the methods in-circuit emulators and monitors can be used for testing firmware along with the hardware. Table I shows some of the sample standard test cases that are mapped to testing methods.

Table 1: Master Test Cases Mapped With the Method to Be Used For Testing

Test Case Type No.	Test Case Description	Test Method
1.	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	Scaffolding
2.	Testing for processing input Fed by a single device through directly calling Interrupt routines	Scaffolding
3.	Testing Response time for each of the event based processing	Simulation
4.	Response time for processing the events that occur simultaneously.	Simulation
5.	Testing Significance of the Bytes (En-dean)	Simulation
6.	Testing for Usage of proper Devices addresses specially MAC addresses.	Assert Macros
7.	Testing for proper data bits related to the devices	Assert Macros
8.	Testing for proper setting of values for data bits	Assert Macros
9.	Testing for Range of values to be contained in a Variable	Assert Macros
10.	Testing for Timing of the signals	Logic Analyzer
11.	Testing for occurrence of signals in a proper sequence	Logic Analyzer
12.	Testing for Validity of the Signals	Logic Analyzer
13.	Testing for proper fetching of addresses related to instructions	Logic Analyzer
14.	Testing the inbuilt Peripheral devices	In-Circuit Emulator
15.	Testing for the response time of a function	In-Circuit Emulator
16.	Testing for scanning of Multiple JTAG Device connections	Monitor
17.	Testing for multiple debugging connections to JTAG Devices	Monitor

More test cases can be added to the master list as the user gains more experience while undertaking testing of the embedded systems.

The test cases that are generated by combinatorial methods are identified with the method that should be used for undertaking testing through mapping of the generated test cases with the standard test cases. The test cases are then grouped based on the method that should be used. The mapping of generated test cases to the test methods are shown in Table II.

4. Testing Embedded through Different test methods

4.1. Testing through Scaffolding

In an embedded system 80% of the code is hardware independent and therefore can be tested on the HOST where the embedded application has been developed and cross compiled to generate image to be loaded into micro controller for execution. The hardware dependent code is scaffolded, meaning the code related to HW is commented and dummy data is used either for transmitting or receiving from a hardware device.

The scaffold software must be able to call interrupt service routines to simulate occurrence of events from hardware devices, must be able to call timer related functions for managing the passage of time etc. Using the scaffolding software several test cases can be input for testing individual program units. Integration between the programs units, testing of occurrence of external events, testing of passage of time etc. are undertaken using scaffolding.

Scaffolding mechanism become complicated when the scoffolded input at one embedded system generates as a scaffold output into another embedded system, in which case the scaffolding software running on a different distributed embedded systems must communicate with each other. While one ES system scaffolds an input device the other scaffolds the output device.

Thus scaffolding technique can be used for undertaking unit testing, event based and time based integration testing when the processes related to them are distributed to be resident of different embedded systems. The scaffold software resident on either of the embedded systems must be able to communicate the inputs and recording of the output consequent to undertaking the testing.

The process that is implemented for undertaking testing through scaffolding is shown in the Figure 4. All the test cases which are to be tested through scaffolding are converted into test scripts and the same are stored in a script file. The original application is updated to scaffold the code using a separate process to which the test cases are provided as inputs. The generated code is compiled, linked, relocated and the image is tested. The image takes the Script as input and produce Test results as output. Then the test results are used to update the test database. The test cases that must be tested along with identified input variables and the selected data for the variables are shown in Table II.

The original application code is scaffolded means the hardware dependent code is commented and the same is placed below. The test scripts generated based on the test cases that must be tested through scaffoldings is shown in Table III. Parser is added to the code which reads the test case as script and executes the same.

4.2. Testing through Assert Macros Scaffolding

A separate process generates the test script required for testing and the test script is stored in a separate file. The test cases that are to be tested using the assert macros are read from the database and test scripts are written into a separate file and the actual testing process reads the script file and executes the test cases one after the other.

The original application code is updated with assert macros that are required for undertaking the testing. The generated code is compiled, linked and relocated and the image is produced. The execution of the image takes the Script as input and produce Test results as output which are written into audit trail.

Figure 5 shows the testing process related to testing the Application through scaffolding and asserts macro techniques and the test results obtained through assert macro testing are shown in Table IV.

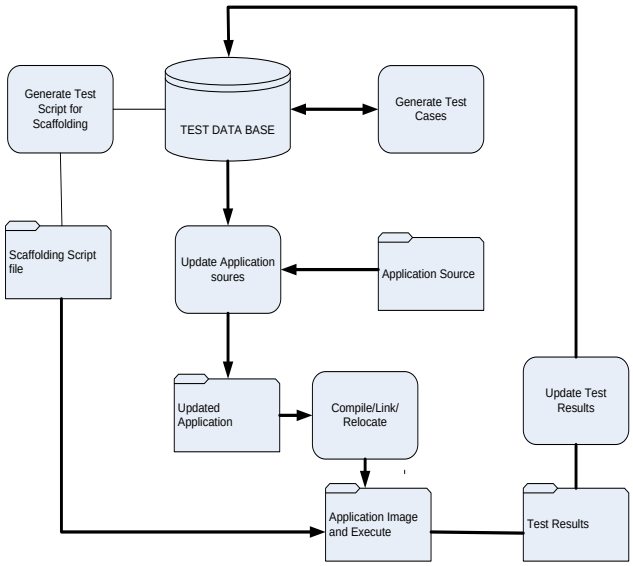

Fig. 4: Testing Embedded Systems through Scaffolding

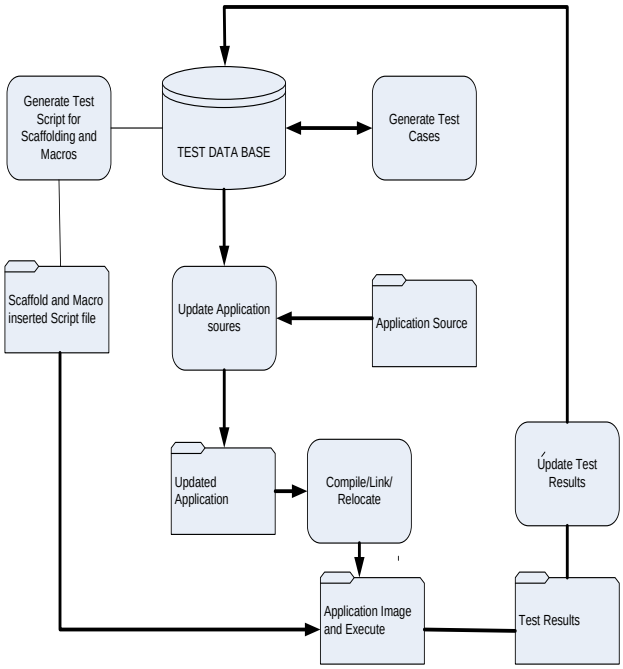

Fig. 5: Process flow for testing through Assert Macros

Test macros are inserted into the code and the code is compiled and executed. The Macros are interested into the code at the location required so that testing is undertaken as required. Following is the sources that are inserted with the macro code.

4.3. Testing through Instruction set simulators

The test cases that are to be tested using the simulators are read from the database and test script is written into separate files containing the commands and command line arguments which can be executed by the chosen simulator.

The simulator undertakes actual test execution by reading through the test script each of the test case one after the other and the test results are written to the audit trail. The test script and the application image are provided as input to the simulator for undertaking the testing. Figure 6 shows the testing process related to testing the Application through Simulators. The test results obtained through Instruction set simulator are shown in the Table V.

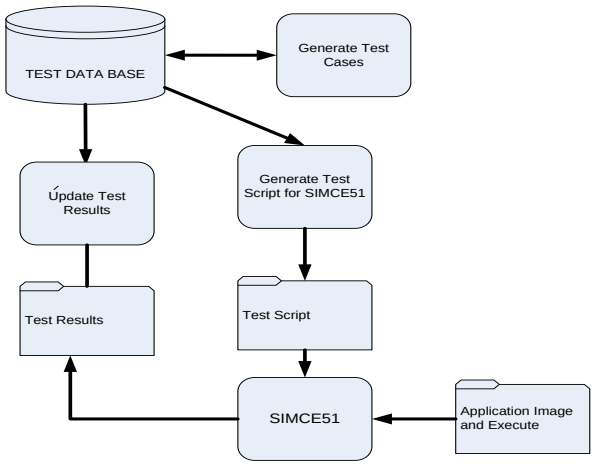

Fig. 6: Testing through Simulators

4.4. Testing through Logic Analyzers

The test cases that should be used to test the hardware are generated and stored in the central repository. These test cases stored in the database are read one after the other and converted into commands and command line arguments understandable by the Logic analyzer. The commands along with command line arguments are stored in a separate file within the database are read one after the other and submitted to the Logic analyzer through an interface developed within the testing process application which is resident at the HOST. The test outcomes are transmitted back by the logic analyzer back to the HOST based application which stores the test results in the audit trail data file.

The probes of the logic analyzer are connected to the hardware at strategic locations so that the commands submitted to the logic analyzers are successfully executed and the results obtained for transmission of the same to the HOST. The process flow related to undertaking the testing of the hardware through Logic analyzer through test process installed at the HOST is shown in the Figure 7 and the test results obtained out of testing undertaken is shown in the Table VI.

4.5. Testing through In Circuit Emulators

The test cases that should be tested using the in-circuit emulator are generated and stored in a database at the HOST. The micro controller in the target is replaced by an in-circuit emulator which emulates the execution of the code as if the code is executed by the originally installed Micro controller. The in circuit emulator has a separate overlay memory at which trace of the application execution and the emulation software are stored which can be used for debugging even if the rest of the Hardware is broken down.

The test cases meant for testing through in-circuit emulator are read one after the other through a separate process and a script file containing the commands and the command line arguments which are understandable by the processes resident at the TARGET and the in-circuit emulator is created and stored in the database.

The test process initiates the execution of the commands contained in the script file and communicates the same to a process resident at the HOST. The process at the TARGET execute the commands received from the HOST in emulation mode and the results obtained due to execution of the test cases through command execution are sent through a communication interface implemented at both ends of the TARGET and the HOST. The test results received at the HOST are stored in an audit trail database file.

The target machine can be made to work in emulation or non-emulation mode by setting the mode switch provided on the

Target Board or the mode can also be set by way of asserting the ALE signals of the controller.

The communication between the Emulator and the HOST based program can be achieved through RS232C interface. Emulators are defined with specified address range at the time of relocating the code. All the memory variables, register and the CPU status and control registers are mapped to the internal memory of the emulator. Emulator can communicate with the HOST even in the case of failure of target. The process flow used for undertaking the testing through in-circuit emulator is shown in the Figure 8.

The test results stored in the audit trail are shown in the Table VII.

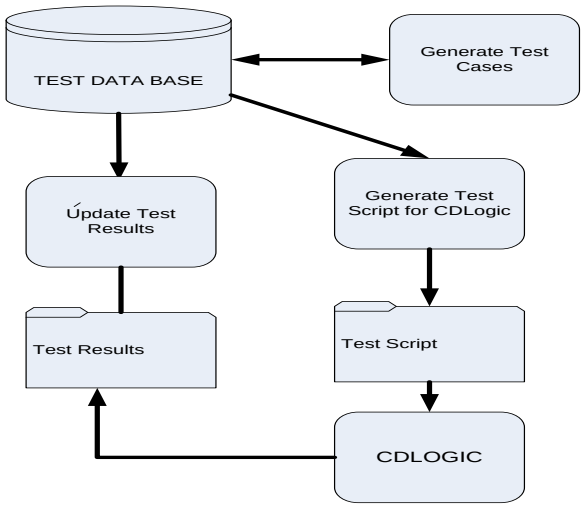

Fig.7: Testing through Logic Analyzer

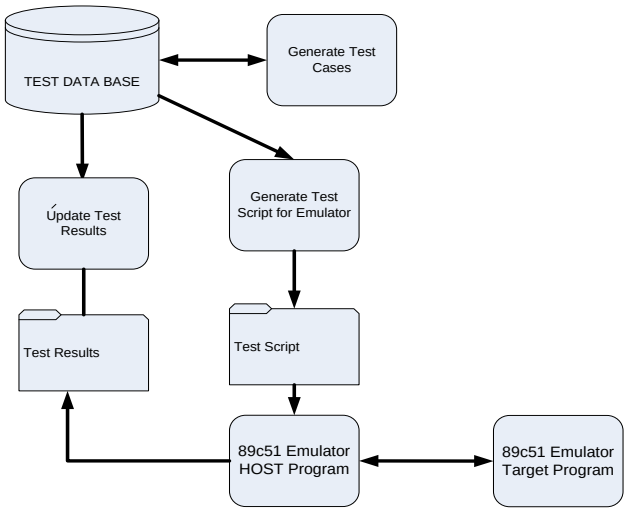

Fig. 8: Testing through In-Circuit Emulator

5. Conclusion

Combinatorial methods help generating test cases considering either input domain, output domain or input-output domain. Test cases must be generated considering testing of embedded systems which include hardware, software and both. Several methods must be sued for undertaking the testing of the embedded system. The test cases generated must be generated considering the method used for undertaking testing of the embedded system. Different methods that can be used for undertaking testing of embedded systems and the way testing is carried, the process flows of which have been presented in this chapter. It could be concluded that while test can be generated by using any of the method/model, separate process is required for undertaking actual testing of the embedded systems.

References

[1] Tsai W. T. , R Mojdehbakhsh and F.Zhu (1998) "Ensuring Systems and Software Reliability in the Safety-Critical Systems*," IEEE ASET 98*, Dallas, Texas , March, pp.no.48- 53.

[2] L. Elliot , R.Mojdehbakhsh and W.T.Tsai (1994) "A Process for developing Safe Software," *Proceedings of the 7th Annual IEEE Symposium on Computer-based Medical Systems IEEE CS Press*, Los Alamitos, California.

[3] F.Zhu. (2002) "A Requirement oriented verification Framework for Real-Time Embedded Systems," *Ph.D. dissertation, Dept. Of Computer Science and Engineering*, University of Minnesota, Minneapolis, MN.

[4] J.K.R Sastry, Rajasekhara Rao K., and Sasi Bhanu J.(2007), "Comprehensive requirements specification of a Cost Effective Testing Tool," *Proceedings of CSI National Conference on Software Engineering*, NCSOFT-2007,pp.73- 85.

[5] Sastry J.K.R., K. Rajasekhara Rao, and J. Sasi Bhanu (2007) "An efficient Architectural framework for a Comprehensive embedded Testing Tool," *Journal of Institute of Engineers-Computer Science*.

[6] Thomas Bauer, Frank Bohr,Dennis Landmann,Taras Beletski,Robert Eschbach and Jesse Poore (2007) "From Requirements to Statistical Testing of Embedded Systems," *IEEE Fourth International Workshop on Software Engineering for Automotive Systems*.

[7] Yi ,Lin Fan, Zeng Wenhua, Chen Guowu (2009) "The Embedded Product Testing Using Cleanroom Statistical Method," *2009 World Congress on Computer Science and Information Engineering*.

[8] K. Rajasekhara Rao (2009) "Architectural framework and models for testing embedded systems", *Thesis submitted to Acharya Nagarjua University*, Guntur for award of PhD degree.

[9] V. Chandra Prakash (2012) "Developing Embedded systems through Clean Room Software Engineering," *Thesis submitted to Acharya Nagarjua University*, Guntur for award of PhD degree.

[10] D.B.K Kamesh (2014) "Testing Embedded systems through clean room software engineering methodology," *Thesis submitted to Shri Venkateswara University*, Gajroula, Utter Pradesh.

Table 2: Mapping Test Cases to Test Methods

Serial	Function Serial	Function to be tested	Test case Serial	Type of testing to be carried	Method to be used for undertaking testing
1	1	Write Initial message to LCD	1A	Test whether LCD is working	Logic Analyser
2		Write Initial message to LCD	1B	Test whether initial message is Displayed Properly	Scaffolding
3	2	Write enter password message on LCD	2A	Test whether LCD is working properly	Logic Analyser
4		Write enter password message on LCD	2B	Test whether the message to enter password is displayed properly	Scaffolding
5	3	Read Key1 and write to LCD	3A	Test whether LCD is working properly	Logic Analyser
6		Read Key1 and write to LCD	3B	Test whether entered Key1 is displayed properly on LCD	Scaffolding
7	4	Read Key2 and write to LCD	4A	Test whether LCD is working properly	Logic Analyser
8		Read Key2 and write to LCD	4B	Test whether entered Key2 is displayed properly on LCD	Scaffolding
9	5	Read Key3 and write to LCD	5A	Test whether LCD is working properly	Logic Analyser
10		Read Key3 and write to LCD	5B	Test whether entered Key3 is displayed properly on LCD	Scaffolding
11	6	Read Key4 and write to LCD	6A	Test whether LCD is working properly	Logic Analyser
12		Read Key4 and write to LCD	6B	Test whether entered Key4 is displayed properly on LCD	Scaffolding
13	7	Read Key5 and write to LCD	7A	Test whether LCD is working properly	Logic Analyser
14		Read Key5 and write to LCD	7B	Test whether entered Key5 is displayed properly on LCD	Scaffolding
15	8	Compare password and write password mismatch on to LCD	8A	Test whether LCD is working properly	Logic Analyser
16		Compare password and write password mismatch on to LCD	8B	Test for password Mismatch	Scaffolding
17		Compare password and write password mismatch on to LCD	8C	Test for password Match	Scaffolding
18	9	Read Ref1 Temperature and write to LCD	9A	Test whether LCD is working properly	Logic Analyser
19		Read Ref1 Temperature and write to LCD	9B	Test for proper reading of Ref1 temperature	Scaffolding
20		Read Ref1 Temperature and write to LCD	9C	Test for proper display of Ref1 Temperature on to LCD	Scaffolding
21	10	Read Ref2 Temperature and write to LCD	10A	Test whether LCD is working properly	Logic Analyser
22		Read Ref2 Temperature and write to LCD	10B	Test for proper reading of Ref2 temperature	Scaffolding
23		Read Ref2 Temperature and write to LCD	10C	Test for proper display of Ref2 Temperature on to LCD	Scaffolding

24	11	Read Temp1 and write to LCD	11A	Test whether LCD is working properly	Logic Analyser
25		Read Temp1 and write to LCD	11B	Test whether temp1 is read properly	Scaffolding
26		Read Temp1 and write to LCD	11C	Test whether the temperature 1 is displayed properly on LCD	Scaffolding
27		Read Temp1 and write to LCD	11D	Test for throughput	Instruction set Simulator
28		Read Temp1 and write to LCD	11E	Test whether the TEMP1 read is within the Range	Assert Macro
29	12	Read Temp1 and send to HOST	12A	Test whether LCD is working properly	Logic Analyser
30		Read Temp1 and send to HOST	12B	Test whether Temp1 is read properly	Scaffolding
31		Read Temp1 and send to HOST	12C	Test whether the temperature 1 is sent to HOST properly	Instruction set Simulator
32		Read Temp1 and send to HOST	12D	Test for throughput	Instruction set Simulator
33	13	Compare Temp1 with Ref1 Temperature and set Pump1 ON	13A	Testing for proper functioning of PUMP-1	Logic Analyser
34		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13B	Test the value of temp-1 to be in a Range	Assert Macro
35		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13C	Test the value of Ref-1 to be in a range	Assert Macro
36		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13D	Test whether pump-1 is on when Temp-1 > Ref1	Scaffolding
37		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13E	Test for response time	Scaffolding
38		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13F	Test for Throughput	Instruction set Simulator
39		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13G	Testing for Timing of the signals	Logic Analyser
40		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13H	Testing for Timing of the signals	Logic Analyser
41		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13I	Testing for occurrence of signals in a proper sequence	Logic Analyser
42		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13J	Testing for occurrence of signals in a proper sequence	Logic Analyser
43		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13K	Testing for validity of the signal	Logic Analyser
44		Compare Temp1 with Ref1 Temperature and set Pump1 ON	13L	Testing for validity of the signal	Logic Analyser
45	14	Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14A	Testing for proper functioning of the PUMP-1	Logic Analyser
46		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14B	Test the value of temp-1 to be in a Range	Assert Macro
47		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14C	Test the value of Ref-1 to be in a range	Assert Macro
48		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14D	Test whether pump-1 is off when Temp-1 < Ref1	Scaffolding
Serial		Function to be tested	Test case Serial	Type of testing to be carried	Method to be used for undertaking testing
49		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14E	Test for response time	Scaffolding
50		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14F	Test for Throughput	Instruction set Simulator
51		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14G	Testing for Timing of the signals	Logic Analyser
52		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14H	Testing for Timing of the signals	Logic Analyser
53		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14I	Testing for occurrence of signals in a proper sequence	Logic Analyser

54		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14J	Testing for occurrence of signals in a proper sequence	Logic Analyser
55		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14K	Testing for validity of the signal	Logic Analyser
56		Compare Temp1 with Ref1 Temperature and set Pump1 OFF	14L	Testing for validity of the signal	Logic Analyser
57	15	Read Temp2 and write to LCD	15A	Test whether LCD is working properly	Logic Analyser
58		Read Temp2 and write to LCD	15B	Test whether temp2 is read properly	Scaffolding
59		Read Temp2 and write to LCD	15C	Test whether the temperature 2 is displayed properly on LCD	Scaffolding
60		Read Temp2 and write to LCD	15D	Test for throughput	Instruction set Simulator
61		Read Temp2 and write to LCD	15E	Test whether the TEMP1 read is within the Range	Assert Macro
62	16	Read Temp2 and send to HOST	16A	Test whether LCD is working properly	Logic Analyser
63		Read Temp2 and send to HOST	16B	Test whether temp2 is read properly	Scaffolding
64		Read Temp2 and send to HOST	16C	Test whether the temperature 2 is sent to HOST properly	Instruction set Simulator
65		Read Temp2 and send to HOST	16D	Test for throughput	Instruction set Simulator
66	17	Compare Temp2 with Ref2 Temperature and set Pump2 ON	17A	Testing for proper functioning of the PUMP-2	Logic Analyser
67		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17B	Test the value of temp-2 to be in a Range	Assert Macro
68		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17C	Test the value of Ref-2 to be in a range	Assert Macro
69		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17D	Test whether pump-2 is on when Temp-2 > Ref2	Scaffolding
70		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17E	Test for response time	Scaffolding
71		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17F	Test for Throughput	Instruction set Simulator
72		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17G	Testing for Timing of the signals	Logic Analyser
73		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17H	Testing for Timing of the signals	Logic Analyser
74		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17I	Testing for occurrence of signals in a proper sequence	Logic Analyser
75		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17J	Testing for occurrence of signals in a proper sequence	Logic Analyser
76		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17K	Testing for validity of the signal	Logic Analyser
77		Compare Temp2 with Ref2 Temperature and set Pump2 ON	17L	Testing for validity of the signal	Logic Analyser
Serial	**Function Serial**	**Function to be tested**	**Test case Serial**	**Type of testing to be carried**	**Method to be used for undertaking testing**
78	18	Compare Temp2 with Ref2 Temperature and set Pump2 OFF	18A	Testing for proper functioning of the PUMP-2	Logic Analyser
79		Compare Temp2 with Ref2 Temperature and set Pump2 OFF	18B	Test the value of temp-2 to be in a Range	Assert Macro
80		Compare Temp2 with Ref2 Temperature and set Pump2 OFF	18C	Test the value of Ref-2 to be in a range	Assert Macro
81		Compare Temp2 with Ref2 Temperature and set Pump2 OFF	18D	Test whether pump-2 is off when Temp-2 <Ref2	Scaffolding
82		Compare Temp2 with Ref2 Temperature and set Pump2 OFF	18E	Test for response time	Scaffolding
83		Compare Temp2 with Ref2 Temperature and set Pump2 OFF	18F	Test for Throughput	Instruction set Simulator
84		Compare Temp2 with Ref2 Temperature and set Pump2 OFF	18G	Testing for Timing of the signals	Logic Analyser

85		**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18H**	**Testing for Timing of the signals**	**Logic Analyser**
86		**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18J**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
87		**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18K**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
88		**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18L**	**Testing for validity of the signal**	**Logic Analyser**
89		**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18M**	**Testing for validity of the signal**	**Logic Analyser**
90	**19**	**Compare Temp1 and Temp2 and set Buzzer ON**	**19A**	**Test the value of temp-1 to be within the range**	**Assert Macro**
91		**Compare Temp1 and Temp2 and set Buzzer ON**	**19B**	**Test the value of temp-2 to be within the range**	**Assert Macro**
92		**Compare Temp1 and Temp2 and set Buzzer ON**	**19C**	**Test whether Buzzer is ON if ABS (Temp1 – Temp20)> 2**	**Scaffolding**
93		**Compare Temp1 and Temp2 and set Buzzer ON**	**19D**	**Testing for response time**	**Scaffolding**
94		**Compare Temp1 and Temp2 and set Buzzer ON**	**19E**	**Testing Throughput**	**Instruction set Simulator**
95		**Compare Temp1 and Temp2 and set Buzzer ON**	**19F**	**Testing for Timing of the signals**	**Logic Analyser**
96		**Compare Temp1 and Temp2 and set Buzzer ON**	**19G**	**Testing for Timing of the signals**	**Logic Analyser**
97		**Compare Temp1 and Temp2 and set Buzzer ON**	**19H**	**Testing for Timing of the signals**	**Logic Analyser**
98		**Compare Temp1 and Temp2 and set Buzzer ON**	**19I**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
99		**Compare Temp1 and Temp2 and set Buzzer ON**	**19J**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
100		**Compare Temp1 and Temp2 and set Buzzer ON**	**19K**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
101		**Compare Temp1 and Temp2 and set Buzzer ON**	**19L**	**Testing for validity of the signal**	**Logic Analyser**
102		**Compare Temp1 and Temp2 and set Buzzer ON**	19M	**Testing for validity of the signal**	**Logic Analyser**
103		**Compare Temp1 and Temp2 and set Buzzer ON**	19N	**Testing for validity of the signal**	**Logic Analyser**
104	**20**	**Compare Temp1 and Temp2 and set Buzzer OFF**	**20A**	**Test the value of temp-1 to be within the range**	**Assert Macro**
105		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20B**	**Test the value of temp-2 to be within the range**	**Assert Macro**
106		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20C**	**Test whether Buzzer is OFF if ABS (Temp1 – Temp20)< 2**	**Scaffolding**
107		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20D**	**Testing for response time**	**Scaffolding**
108		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20E**	**Testing Throughput**	**Instruction set Simulator**
109		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20F**	**Testing for Timing of the signals**	**Logic Analyser**
110		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20G**	**Testing for Timing of the signals**	**Logic Analyser**
111		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20H**	**Testing for Timing of the signals**	**Logic Analyser**
112		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20G**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
113		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20H**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
114		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20I**	**Testing for occurrence of signals in a proper sequence**	**Logic Analyser**
115		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20J**	**Testing for validity of the signal**	**Logic Analyser**

116		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20K**	**Testing for validity of the signal**	**Logic Analyser**
117		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20L**	**Testing for validity of the signal**	**Logic Analyser**

Table 3: Test cases and Results for Testing- Scaffolding

Serial	Function Serial	Function to be tested	Test case Serial	Type of testing to be carried	Input Vatiable-1/ Command	Input Vatiable-1 Value	Input Vatiable-2	Input Vatiable-2	Input Vatiable-3	Input Vatiable-3	Output Variable-1	Output Vatiable-1 Value	Expected output Value	Pass / Fail
2	**1**	**Write Initial message to LCD**	**01B**	**Test whether initial message is Displayed Properly**	INIT-MESAGGE	"ABC"					LCD-WRITE	"ABC"	"ABC"	P
4	**2**	**Write enter password message on LCD**	**02B**	**Test whether the message to enter password is displayed properly**	MEASSE-FOR-PASSWD-ENTRY	"Enter Password"					LCD-WRITE	"Enter Password'	"Enter Password'	P
6	**3**	**Read Key1 and write to LCD**	**03B**	**Test whether entered Key1 is displayed properly on LCD**	KEY-1	"K1"					LCD-WRITE	"K1"	"K1"	P
8	**4**	**Read Key2 and write to LCD**	**04B**	**Test whether entered Key2 is displayed properly on LCD**	KEY-2	"K2"					LCD-WRITE	"K2"	"K2"	P
10	**5**	**Read Key3 and write to LCD**	**05B**	**Test whether entered Key3 is displayed properly on LCD**	KEY-3	"K3"					LCD-WRITE	"K3"	"K3"	P
12	**6**	**Read Key4 and write to LCD**	**06B**	**Test whether entered Key4 is displayed properly on LCD**	KEY-4	"K4"					LCD-WRITE	"K4"	"K4"	P
14	**7**	**Read Key5 and write to LCD**	**07B**	**Test whether entered Key5 is displayed properly on LCD**	KEY-5	"K5"					LCD-WRITE	"K5"	"K5"	P
16	**8**	**Compare password and write password mismatch on to LCD**	**08B**	**Test for password Mismatch**	PASS-WD	"Mudra"					LCD-WRITE	"Miss Match Password'	"Miss Match Password'	P
17		**Compare password and write password mismatch on to LCD**	**08C**	**Test for password Match**	PASS-WD	"jkrs2009"					LCD-WRITE	"Password Match"	"Password Match"	P
19	**9**	**Read Ref1 Temperature and write to LCD**	**09B**	**Test for proper reading of Ref1 temperature**	REF1	30					LCD-WRITE	30	30	P
20		**Read Ref1 Temperature and write to LCD**	**09C**	**Test for proper display of Ref1 Temperature on to LCD**	REF1	30					LCD-WRITE	30	30	P

Table 4: Test cases and Results for Testing- Assert Macros

Serial	Function Serial	Function to be tested	Test case Serial	Type of testing to be carried	Input Vatiable-1/ Command	Input Vatiable-1 value	Input Vatiable-2	Input Vatiable-2 value	Input Vatiable-3	Input Vatiable-3 value	Output Variable-1	Output Vatiable-1	Expected output value	Pass / Fail
28	**11**	**Read Temp1 and write to LCD**	**11E**	**Test whether the TEMP1 read is within the Range**	TEMP1	35	Start-val	1	End-val	255	TEMP1-STA	35	35	P
34	**13**	**Compare Temp1 with Ref1 Temperature and set Pump1 ON**	**13B**	**Test the value of temp-1 to be in a Range**	TEMP1	35	Start-val	1	End-val	255	T1-RANGE-STA	Y	Y	P
35		**Compare Temp1 with**	**13C**	**Test the value of Ref-**	REF1	35	Start-val	1	End-val	255	REF1-RANGE-	Y	Y	P

		Ref1 Temperature and set Pump1 ON		**1 to be in a range**						STA			
46	**14**	**Compare Temp1 with Ref1 Temperature and set Pump1 OFF**	**14B**	**Test the value of temp-1 to be in a Range**	TEMP1	35	Start-val	1	End-val	255	T1-RANGE-STA	Y	Y	P
47		**Compare Temp1 with Ref1 Temperature and set Pump1 OFF**	**14C**	**Test the value of Ref-1 to be in a range**	REF1	35	Start-val	1	End-val	255	REF1-RANGE-STA	Y	Y	P
61	**15**	**Read Temp2 and write to LCD**	**15E**	**Test whether the TEMP1 read is within the Range**	TEMP2	35	Start-val	1	End-val	255	TEMP2-STA	35	35	P
67	**17**	**Compare Temp2 with Ref2 Temperature and set Pump2 ON**	**17B**	**Test the value of temp-2 to be in a Range**	TEMP2	35	Start-val	1	End-val	255	T2-RANGE-STA	Y	Y	P
68		**Compare Temp2 with Ref2 Temperature and set Pump2 ON**	**17C**	**Test the value of Ref-2 to be in a range**	REF2	35	Start-val	1	End-val	255	REF2-RANGE-STA	Y	Y	P
79	**18**	**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18B**	**Test the value of temp-2 to be in a Range**	TEMP2	35	Start-val	1	End-val	255	T2-RANGE-STA	Y	Y	P
80		**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18C**	**Test the value of Ref-2 to be in a range**	REF2	35	Start-val	1	End-val	255	REF2-RANGE-STA	Y	Y	P
90	**19**	**Compare Temp1 and Temp2 and set Buzzer ON**	**19A**	**Test the value of temp-1 to be within the range**	TEMP1	35	Start-val	1	End-val	255	T1-RANGE-STA	Y	Y	P
91		**Compare Temp1 and Temp2 and set Buzzer ON**	**19B**	**Test the value of temp-2 to be within the range**	TEMP2	35	Start-val	1	End-val	255	T2-RANGE-STA	Y	Y	P
104	**20**	**Compare Temp1 and Temp2 and set Buzzer OFF**	**20A**	**Test the value of temp-1 to be within the range**	TEMP1	35	Start-val	1	End-val	255	T1-RANGE-STA	Y	Y	P
105		**Compare Temp1 and Temp2 and set Buzzer OFF**	**20B**	**Test the value of temp-2 to be within the range**	TEMP2	35	Start-val	1	End-val	255	T2-RANGE-STA	Y	Y	P

Serial	**Function Serial**	**Function to be tested**	**Test case Serial**	**Type of testing to be carried**	**Input Vatiable-1/ Command**	**Input Vatiable-1 value**	**Input Vatiable-2**	**Input Vatiable-2 value**	**Input Vatiable-3**	**Input Vatiable-3 value**	**Output Variable-1**	**Output Vatiable-1 value**	**Expected output value**	**Pass / Fail**

27	**11**	**Read Temp1 and write to LCD**	**11D**	**Test for throughput**	T1-THRU-TIME	10					T1-THRU	10	10	P
31	**12**	**Read Temp1 and send to HOST**	**12C**	**Test whether the temperature 1 is sent to HOST properly**	TEMP1	30					TX	30	30	P
32	**12**	**Read Temp1 and send to HOST**	**12D**	**Test for throughput**	T1-THRU-TIME	10					T1-THRU	10	10	P
38	**13**	**Compare Temp1 with Ref1 Temperature and set Pump1 ON**	**13F**	**Test for Throughput**	T1-THRU-TIME	10					T1-THRU	10	10	P
50	**14**	**Compare Temp1 with Ref1 Temperature and set Pump1 OFF**	**14F**	**Test for Throughput**	T1-THRU-TIME	10					T1-THRU	10	10	P
60	**15**	**Read Temp2 and write to LCD**	**15D**	**Test for throughput**	T2-THRU-TIME	10					T2-THRU	10	10	P
64	**16**	**Read Temp2 and send to HOST**	**16C**	**Test whether the temperature 2 is sent to HOST properly**	TEMP2	30					TX	30	30	P
65		**Read Temp2 and send to HOST**	**16D**	**Test for throughput**	T2-THRU-TIME	10					T2-THRU	10	10	P
71	**17**	**Compare Temp2 with Ref2 Temperature and set Pump2 ON**	**17F**	**Test for Throughput**	T2-THRU-TIME	10					T2-THRU	10	10	P
83	**18**	**Compare Temp2 with Ref2 Temperature and set Pump2 OFF**	**18F**	**Test for Throughput**	T2-THRU-TIME	10					T2-THRU	10	10	P
94	**19**	**Compare Temp1 and Temp2 and set Buzzer ON**	**19E**	**Testing Throughput**	BUZZER-THRU-TIME	10					BUZZER-THRU	10	10	P
108	**20**	**Compare Temp1 and Temp2 and set Buzzer OFF**	**20E**	**Testing Throughput**	BUZZER-THRU-TIME	10					BUZZER-THRU	10	10	P

Table 5: Test cases and Results for Testing- Instruction Set Simulators

Table 6: Test cases and Results for Testing- LOGIC Analyzers

Serial	**Function Serial**	**Function to be tested**	**Test case Serial**	**Type of testing to be carried**	**Input Vatiable-1/ Command**	**Input Vatiable-1 value**	**Input Vatiable-2**	**Input Vatiable-2 value**	**Input Vatiable-3**	**Input Vatiable-3 value**	**Output Variable-1**	**Output Vatiable-1 val-**	**Expected output value**	**Pass / Fail**
3	**2**	**Write enter password message on LCD**	**02A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P

5	**3**	**Read Key1 and write to LCD**	**03A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
7	**4**	**Read Key2 and write to LCD**	**04A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
9	**5**	**Read Key3 and write to LCD**	**05A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
11	**6**	**Read Key4 and write to LCD**	**06A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
13	**7**	**Read Key5 and write to LCD**	**07A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
15	**8**	**Compare password and write password mismatch on to LCD**	**08A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
18	**9**	**Read Ref1 Temperature and write to LCD**	**09A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
21	**10**	**Read Ref2 Temperature and write to LCD**	**10A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
24	**11**	**Read Temp1 and write to LCD**	**11A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
29	**12**	**Read Temp1 and send to HOST**	**12A**	**Test whether LCD is working properly**	TEST-PORT	P1					LCD-STAT	Y	Y	P
33	**13**	**Compare Temp1 with Ref1 Temperature and set Pump1 ON**	**13A**	**Testing for proper functioning of PUMP-1**	TEST-PORT	P2.2					PUMP 1-STA	ON	ON	P
39	**13**	**Compare Temp1 with Ref1 Temperature and set Pump1 ON**	**13G**	**Testing for Timing of the signals**	TEMP1	P1	PUMP 1	P2.2			TEMP 1-SENSE - TIME	10	10	P
40	**13**	**Compare Temp1 with Ref1 Temperature and set Pump1 ON**	**13H**	**Testing for Timing of the signals**	TEMP2	P2	PUMP 2	P2.3			PUMP 1-SENSE -TIME	15	15	P
41	**13**	**Compare Temp1 with Ref1 Temperature and set Pump1 ON**	**13I**	**Testing for occurrence of signals in a proper sequence**	TEMP1	P1	PUMP 1	P2.2			TEMP 1-SENSE - TIME	10	10	P

Table 7: Test cases and Results for Testing- In circuit emulators

Serial	**Function Serial**	**Function to be tested**	**Test case Serial**	**Type of testing to be carried**	**Input Vatiable-1/ Command**	**Input Vatiable-1 value**	**Input Vatiable-2**	**Input Vatiable-2 value**	**Input Vatiable-3**	**Input Vatiable-3 value**	**Output Variable-1**	**Output Vatiable-1 value**	**Expected output value**	**Pass / Fail**
118	**21**	**Testing for changes in Data**	**21A**	**Testing for changes in Data at specified memory locations**	**#1000**						**TEMP1**	**0-255**	**0-255**	**P**
119	**22**	**Testing for system Bring up**	**22A**	**Testing for system Bring up**	**#0001**						**STARTUP -STA**	**1**	**1**	**P**