International Journal of Engineering & Technology, 7 (2.7) (2018) 297-302

International Journal of Engineering & Technology

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

Research Paper

Testing distributed embedded systems through logic analyzer

Chaitanya Kilaru [1] *, Dr. JKR Sastry [2], Dr. K RajaSekhara Rao [3]

[1] Research scholar, Department of CSE, JNTU Hyderabad, Telangana
[2] Department of ECSE, KLEF deemed to be University, Vaddeswaram, Andhra Pradesh
[3] Director, Usha Rama College of Engineering and Technology, Telaprolu, Krishna District, Andhra Pradesh
**Corresponding author E-mail: kilaru.chaitanya84@gmail.com*

Abstract

Testing distributed embedded systems is complex as the individual systems connected on to the network are heterogeneous in nature. The communication system that is used for establishing the networking also varies greatly leading to different testing requirements. Testing of embedded systems can be carried using different methods that include Scaffolding, assert macros, instruction set simulators. In-circuit emulators, logic analyzers each requiring establishment of different testing environment required for undertaking actual testing. Testing of any embedded systems involves testing hardware, testing hardware dependent code, and testing hardware independent code. Logic analyzers are generally used for testing proper working of the Hardware.
In this paper, a framework is presented using which testing of hardware distributed across the distributed embedded system using logic analyzer is presented.

Keywords: *Testing Embedded Systems; Distributed Embedded Systems; Logic Analyzers; Testing Embedded Hardware*.

1. Introduction

Distributed embedded systems are frequently employed for implementing many of the application such as home automation, Automobile systems, Air surveillance and quite recently as sub nets forming an IOT (Internet of things). Distributed embedded systems are quite complex due to heterogeneity of hardware and software used to develop each of the embedded system that participates in a distributed embedded network. Communication as such is one of the complex issues. The continuous availability of such a system is quite low as failure of any of the embedded system will lead to failure of entire system. Every embedded system that gets connected with the distributed embedded network must be functioning fully for the entire embedded system to functioning fully.
Comprehensive testing of a distributed embedded system becomes most important issue that requires due attention and effort. In literature, no specific methods have been presented that deal with testing a distributed embedded system. Testing an embedded system requires testing hardware, hardware dependent code, and hardware independent code. In literature different methods have presented for testing different parts of an embedded systems. Logic analyzers are used for testing hardware, the methods that include Scaffolding, Assert Macros and Instruction set simulators are used for testing hardware independent code and In-Circuit emulators used for testing hardware independent code. While the TARGET (Embedded system Board) is needed for undertaking testing of the hardware through logic analyzer, HOST (Remote PC interfaced with the Target) is sufficient to test the hardware independent code using either methods that include Scaffolding, assert macros and Instruction set simulators. Both Target and HOST are required for testing hardware independent code using in circuit emulator and monitors.
A distributed embedded system is achieved through networking individual embedded systems using one of the communications systems that include CAN, I^2C, USB, and RS485 etc. The kind of testing required and the method to be used for testing a communication system is dependent on the type of communications system used. All the communications systems which are used for establishing distributed embedded systems largely differ. I2C, CAN, USB and RS485 are the communication systems that are frequently used for undertaking networking of distributed embedded systems. The individual embedded systems are heterogeneous and the communication interfaces supported on each of the embedded system differs, thus requiring the conversion and portability.
The methods that can be employed for testing a standalone system can be explored to find whether the same are suitable for testing a distributed embedded system. Testing of an embedded system using any of the methods requires proper environment setting and processes using which testing can be carried. Environment setting at each of the distributed location is one of the most complex issues. Testing at each of the locations simultaneously for undertaking testing considering entire distributed embedded systems is yet another complicated issue.
Logic analyzers can be effectively employed for testing to find whether everything connected with the hardware is taking place as required. Malfunctioning of the hardware must also be noticed. Testing to finding timing, sequencing and validity of the signal, response time, through put, change of content of the variables etc., can be tested using logic analyzers. Logic Analyzers as such can be effectively used for testing proper working of the hardware.
In this paper, a testing framework has been presented that can be used for undertaking testing distributed embedded systems. The framework is developed over in-famous method developed around logic analyzer that is frequently used for undertaking testing of stand-alone embedded systems. A general premise has been presented that caters for breaking a test cases that needs to be used for testing considering entire distributed embedded system as a whole. The breaking of the test cases is done in such a way that the elementary test case can be tested at a location using a specified

method. The test results obtained at each of the locations are combined and ordered and then merged to get overall status of having tested the merged test case considering entire distributed embedded system as a whole. Audit trails have been carried to find the overall reliability of distributed systems.

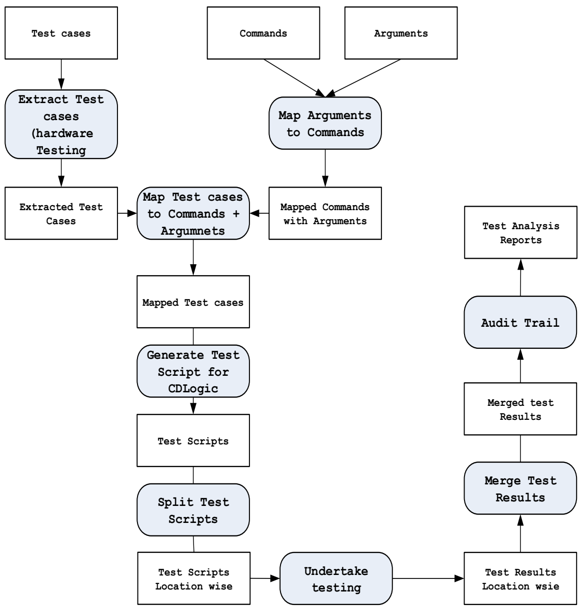
2. Related work

The Logic Analyzer can be used for sampling different signals simultaneously or can be used to change the values of the signals [1]. Logic analyzers provide larger widow for testing signals.
David E. Simon [2] has explained the way the logic analyzer can be used for undertaking testing of standalone embedded systems. He explained the kind of tests that can be carried through a logic analyzer. He explained that testing can be carried both in static and timing mode. The timing of signals related tests can be carried using the timing of the Logic Analyzer.
Textronics [3] in their white paper shown how one can undertake testing of FPGAA, Memory devices, verifying the signal integrity and the way data move between the devices and the controller.
The challenges involved in testing and debugging of embedded systems have been explained by Alex Dicson [4]. He has explained several instruments that can be used for testing of embedded systems. He has deal with the challenges that one should meet in designing the embedded systems. He has given details of various Agilent Logic Analyzers that can be used for undertaking testing of the embedded systems.
Dario et al., [5] presented a logic / protocol analyzer that help learning of asynchronous serial data communication by capturing and visualizing the real time diagrams from a laboratory unit through use of Saleae logic analyzers. He opined that use of oscilloscope or manual testing of the serial communication program does not yield proper behavior of the same.
Doug Beck [6] has presented the way to understand the triggering process implemented within logic Analyzer. They have also explained the way firmware can be de-bugged using the logic Analyzer. Sonam [7] has presented a logic analyzer that is built on ARM based controller. The logic analyzer developed by them have extensive features that will allow one to understand relationships between timing on various data elements that flow on the data bus.
Varsha Karambelkar et al., [8] have proposed a logic analyzer that helps verifying the digital circuit design. The have built the system using ARM processor which is built on RISC processor. The speed of such a processor matches the speed of processing that happens within digital logic design.
Will McGrath [9] have proposed a method of visualizing and checking the behavior of an embedded system across hardware and software. They have shown the way to verify the boundary between hardware and software. They have proposed a method to trace out the boundary between the hardware and software. The feathers and working of "vivado" logic analyzer has been explained in a technical paper posted at so-logic.net [10]. In the white paper posted on embedded.com the process to be used for trouble shooting real-time software issues using logic analyzer [11].

3. Investigations and findings

3.1. Process flow for testing distributed embedded systems through logic analyzers

Testing of an embedded system can be undertaken through several methods that include scaffolding, assert macros, instruction set simulators, In-circuit emulators, monitors and Logic analyzers. The testing of an embedded system involves testing hardware, software and both. The methods scaffolding, assert macros and instruction set simulators are used for testing hardware independent code while hardware dependent code can be tested through in-circuit emulators and monitors. The testing of hardware as can be carried through use of logic analyzers which can be fed with test cases through commands fed through a PC interfaced with Logic analyzers. Logic Analyzers after executing the commands makes available the test results to the PC where the results are stored and an audit trail is connected.
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 1.
The test cases that are related to testing hardware are selected form the test suite that is meant for testing the entire distributed embedded systems. Logic analyzers are to be fed with commands for it do the testing required and make available the result gained out of undertaking testing to HOST which actually submit the commands along with the arguments to be used by the logic analyzer for undertaking testing.
A logic analyzer is capable of undertaking different types of testing that includes presence of signals, timing of signals, occurrence of signals in a sequence validity period of a signal, throughput of the occurrence of the signal; whether a device is faulty, speed of transmission of a signal etc. The actual working of the hardware can be thoroughly tested using the logic analyzer. Using single Logic analyzer, a localized embedded hardware can be thoroughly tested. However, for sub-system to work properly on a distributed embedded system, the hardware at different locations within the distributed embedded system must be properly functioning. A test case may involve in testing proper working of the hardware at defend locations.
The embedded system can be networked using any of the networking system that are in vogue such as CAN, I2C, USB and RS485. The hardware interface provided at the individual embedded systems must be working properly for effecting the communication required. It is necessary to test whether the networking interfaces are in proper conduction while proper working of actual communication can be tested through software testing.


Test cases
Commands
Arguments
Extract Test cases (hardware Testing
Map Arguments to Commands
Extracted Test Cases
Map Test cases to Commands + Argumnets
Mapped Commands with Arguments
Test Analysis Reports
Mapped Test cases
Audit Trail
Generate Test Script for CDLogic
Merged test Results
Test Scripts
Merge Test Results
Split Test Scripts
Test Scripts Location wise
Undertake testing
Test Results Location wsie
Fig. 1: Testing Through Logic Analyzer.

THROUGHPUT	Response time of Temp-2
SIGNALING	Throughput time of Temp-1
SIGNAL-SEQUENCE	Throughput time ofTemp-2
SIGNAL-VALIDITY	Response time of Pump-1 to be ON

3.2. Environment setting for testing distributed embedded systems through logic analyzer

The settings of environment for undertaking the testing through logic analyzer include maintenance of a repository of commands, arguments and the mapping of the arguments to the commands. A command as such indicates the kind of testing that a logic analyzer must carry. The commands as such will vary from type of logic analyzer to logic analyzer. For this project CDLOGIC is used for undertaking testing.

While Table 1 shows repository of Test locations, Table 2 shows repository of commands, Table 3 shows different types of arguments that can be mapped to the commands. Table 4 shows the mapping of the arguments to the commands.

Table 1: Repository of Test Locations

S. no	Location code	Location description	Location address
1.	L1	Microcontroller based location for sensing and monitoring Temperature-1	20
2.	L2	Microcontroller based location for sensing and monitoring Temperature-2	30
3.	L3	Microcontroller based location for sensing and monitoring Pump-1	40
4.	L4	Microcontroller based location for sensing and monitoring Pump-2	50
5.	L5	Microcontroller based location for sensing and monitoring Central Monitoring system	60

Table 2: Logic Analyser - Commands Repository

Command	Command Description
RESPONSE	Compute response time

Table 3: Maintain Arguments

S. No.	Name of the Argument	Type of the Argument	Start Value	End Value
1.	P1	Char(2)	L1	
2.	P2	Char(2)	L2	
3.	P3.1	Char(2)	L3	
4.	P.3.2	Char(2)	L4	
5.	P3.3	Char(2)	L5	

Table 4: Mapping Arguments to Commands

Command	Argument- 1	Argument-2	Argument-3
RESPONSE-T1	P1	Location-1	
RESPONSE-T2	P2	Location-2	
RESPONSE –P1	P3.1	Location-3	
RESPONSE-P2	P3.2	Location-4	
RESPONSE-BZ	P3.3	Location-5	
LATENCY-1	P1	P3.1	
LATENCY-2	P2	P3.2	
LATENCY	P1	P2	P3.3
MEMCOMP-1	#1000	#1001	
MEMCOMP-2	#2000	#2001	
MEMCOMP-3	#1000	#2000	

The test cases that must be tested through logic analyzers are extracted from the master test cases and the same are mapped to the commands that can be processed by Logic Analyzers. The functional requirements of a distributed embedded system are mapped to testing requirements. The testing requirements are decomposed into elementary level testing requirements such that a decomposed test case can be tested at an embedded system that is connected on to a network. Table 5 shows the decomposition of test cases and mapping locations to the decomposed test cases.

The mapped test cases are used to generate test scripts that can be fed to the Logic Analyzer. The generated test scripts are shown in Table 6

Table 5: Test Cases for Testing Distributed Embedded System Using Logic Analyser

Functional requirement Number	Functional Requirement	Test Requirements	Test Case serial	Sub-Test case serial	Split test cases	Command	Testing Location
10	Compute response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	Test response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	13	13A	Test for Response time of temp-1	RE-SPOSNE-P1	ES-1
11	Compute response between the Reading the Temp-1 and stopping the pump-1	Test response time considering reading of Temp-1 and stopping the pump-1 when Temp-1 < Reference Temp-1	17	17A	Test for Response time of temp-1	RE-SPOSNE-P1	ES-1
12	Compute response between the Reading the Temp-1 and starting the buzzer	Test response time considering reading of Temp-1 and Temp-2 and starting the buzzer when difference between Temp-1 and Temp-2 >2	21	21A	Test for response time of temp-1	RE-SPOSNE-P1	ES-1
13	Compute response between the Reading the Temp-1 and stopping the buzzer	Test response time considering reading of Temp-1 and Temp-2 and stopping the buzzer when difference between Temp-1 and Temp-2 < 2	25	25A	Test for Response time of temp-1	RE-SPOSNE-P1	ES-1
12	Compute response between the Reading the Temp-1 and starting the buzzer	Test response time considering reading of Temp-1 and Temp-2 and starting the buzzer when difference between Temp-1 and Temp-2 >2	21	21B	Test for response time of Temp-2	RE-SPOSNE-P2	ES-2

13	Compute response between the Reading the Temp-1 and stopping the buzzer	Test response time considering reading of Temp-1 and Temp-2 and stopping the buzzer when difference between Temp-1 and Temp-2 < 2	25	25B	Test for response time of Temp-2	RE-SPOSNE-P2	ES-2
10	Compute response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	Test response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	13	13C	Test for Response time of Pump-1 when on	LA-TENCY-1	ES-3
11	Compute response between the Reading the Temp-1 and stopping the pump-1	Test response time considering reading of Temp-1 and stopping the pump-1 when Temp-1 < Reference Temp-1	17	17C	Test for Response time of Pump-1 when off	LA-TENCY-1	ES-3
10	Compute response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	Test response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	13	13B	Test if temp-1 > reference temp-1	MEM-COMP-1	ES-5
11	Compute response between the Reading the Temp-1 and stopping the pump-1	Test response time considering reading of Temp-1 and stopping the pump-1 when Temp-1 < Reference Temp-1	17	17B	Test if temp-1 < reference temp-1	MEM-COMP-1	ES-5
12	Compute response between the Reading the Temp-1 and starting the buzzer	Test response time considering reading of Temp-1 and Temp-2 and starting the buzzer when difference between Temp-1 and Temp-2 >2	21	21C	Test if absolute of Temp1 - Temp-2 > 2	MEM-COMP-3	ES-5
12	Compute response between the Reading the Temp-1 and starting the buzzer	Test response time considering reading of Temp-1 and Temp-2 and starting the buzzer when difference between Temp-1 and Temp-2 >2	21	21D	Test for Response time of BUZZER when ON	LA-TENCY-3	ES-5
13	Compute response between the Reading the Temp-1 and stopping the buzzer	Test response time considering reading of Temp-1 and Temp-2 and stopping the buzzer when difference between Temp-1 and Temp-2 < 2	25	25C	Test if absolute of Temp1 - Temp-2 < 2	MEM-COMP-3	ES-5
Functional requirement Number	Functional Requirement	Test Requirements	Test Case serial	Sub-Test case serial	Split test cases	Command	Testing Location
13	Compute response between the Reading the Temp-1 and stopping the buzzer	Test response time considering reading of Temp-1 and Temp-2 and stopping the buzzer when difference between Temp-1 and Temp-2 >2	25	25D	Test for Response time of BUZZER when OFF	LA-TENCY-3	ES-5

Table 6: Generated Test Scripts

Functional requirement Number	Test Requirements	Test Case serial	Sub-Test case serial	Split test cases	Testing Location	Test Script
10	Test response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	13	13A	Test for Response time of temp-1	ES-1	RE-SPOSNE-P1
11	Test response time considering reading of Temp-1 and stopping the pump-1 when Temp-1 < Reference Temp-1	17	17A	Test for Response time of temp-1	ES-1	RE-SPOSNE-P1
12	Test response time considering reading of Temp-1 and Temp-2 and starting the buzzer when difference between Temp-1 and Temp-2 >2	21	21A	Test for response time of temp-1	ES-1	RE-SPOSNE-P1
13	Test response time considering reading of Temp-1 and Temp-2 and stopping the buzzer when difference between Temp-1 and Temp-2 < 2	25	25A	Test for Response time of temp-1	ES-1	RE-SPOSNE-P1

3.3. Testing through logic analyzer

Testing at all locations using logic analyzer considering the test cases that must be tested at those locations is carried. There are 5 Locations within the distributed embedded system in the pilot project. Following sections shows the way testing is carried at location-1 similar process is used for testing at all the remaining locations. The test cases and the related test scripts to be used for undertaking the testing at Location-1 using Logic Analyzer are shown in Table 7.

3.4. Experimental results

The logic analyzer program on the HOST reads a test script from the script file and sends the same to the Logic Analyzer. The HOST generally sends the commands that must be executed by the logic analyzer. The Logic analyzer executes the test and sends back to the PC the test result. The test results stored at all PCs are accumulated and merged to find the overall status of the testing conducted through Logic Analyzer. The test results obtained after testing using the above process is sent to the HOST where the results are written to audit trail are shown in the Table 8.

Table 7: Test Scripts at Location-1 Using Logic Analyzer

Functional requirement Number	Test Requirements	Test Case serial	Sub-Test case serial	Split test cases	Testing Location	Test Script
10	Test response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	13	13A	Test for Response time of temp-1	ES-1	RE-SPOSNE-P1
11	Test response time considering reading of Temp-1 and stopping the pump-1 when Temp-1 < Reference Temp-1	17	17A	Test for Response time of temp-1	ES-1	RE-SPOSNE-P1
12	Test response time considering reading of Temp-1 and Temp-2 and starting the buzzer when difference between Temp-1 and Temp-2 >2	21	21A	Test for response time of temp-1	ES-1	RE-SPOSNE-P1
13	Test response time considering reading of Temp-1 and Temp-2 and stopping the buzzer when difference between Temp-1 and Temp-2 < 2	25	25A	Test for Response time of temp-1	ES-1	RE-SPOSNE-P1

Table 8: Test Results - Location-1 - Logic Analyzer

Functional Requirement Number	Test Case	Test Case serial	Sub-Test case serial	Split test cases	Command	Test output	Expected output	Test Fail/Pass
10	Test response time considering reading of Temp-1 and starting the pump-1 when Temp-1 > Reference Temp-1	13	13A	Test for Response time of temp-1	RE-SPOSNE-P1	20	20	P
11	Test response time considering reading of Temp-1 and stopping the pump-1 when Temp-1 < Reference Temp-1	17	17A	Test for Response time of temp-1	RE-SPOSNE-P1	20	20	P
12	Test response time considering reading of Temp-1 and Temp-2 and starting the buzzer when difference between Temp-1 and Temp-2 >2	21	21A	Test for response time of temp-1	RE-SPOSNE-P1	20	20	P
13	Test response time considering reading of Temp-1 and Temp-2 and stopping the buzzer when difference between Temp-1 and Temp-2 < 2	25	25A	Test for Response time of temp-1	RE-SPOSNE-P1	20	20	P

3.5 Combining test results from all locations

The test results obtained at all locations are combined and merged based on the belonging ness of a test result to a master test case and a single unified test results that could have been obtained if a unified testing procedure using logic analyzer is used. An audit trail is conducted to find the reliability of the hardware.

4. Conclusions

Testing any embedded systems involves testing hardware, hardware independent code and hardware dependent code. Hardware independent code can be tested through using methods such as Scaffolding, Assert macros and instruction-set simulators. Hardware dependent code can be tested through In-Circuit emulator and Monitors.
Hardware as such has to be tested using special gadgets such as Logic Analyzer. Tests that include testing for timing, occurrence of proper signal or occurrences of proper sequence signal, validity of signal, response time etc., could be tested by using Logic Analyzer. Intermixing of the test results produced through different methods and coming out with overall results of having carried the testing considering entire distributed system will help in finding the non-functioning all malfunctioning of the distributed embedded systems.

Acknowledgement

I acknowledge KLEF (deemed to be university) for having given the opportunity and ambiance required for undertaking the experimentation related to this work. I acknowledge Dr. A. Srinath for extending his moral support, and research related thought provocation

References

[1] Wayne Hendrix Wolf, Computers as Components: Principles of Embedded Computing System Design, Gulf Professional Publishing,2005, page no.223-225,296-304
[2] David E. Simon, An Embedded Software Premier, Pearson Publications, 1999, page no.313-319
[3] White paper, The XYZs of Logic Analysers Primer, Tektronix
[4] Alex Dickson, Embedded Debug Overcoming Design Challenges Using Agilent Logic Analysers, Oscilloscopes, and Protocol Analysis Tools Agilent Technologies
[5] Dario Schor, Witold Kinsner, and Kathryn Marcynuk, Reinforcing the design foundation of asynchronous serial data communications using logic and protocols analyzers, CEEA11, St. John's, NL, June 6-8, 2011
[6] Doug Beck, Hewlett-Packard, Understanding Logic Analyser Triggering, Software / hardware integration -Insight, issue.3, Vol.4, 1999, page no. 22-26
[7] Sonam M. Tiple, Swetansha Chauhan, Aishwarya Sharma, Prof. V. P. Mulik, Prof. A. P. Yadav, Arm Based Logic Analyzer, International Journal of Advanced Research in Computer Science and Software Engineering, issue.3, vol. 5, Mar 2015, page no.426-428

[8] Varsha Karambelkar, Prof. A. A. Shinde, Logic analyser, International Journal of Engineering Research & Technology, issue.3, vol.1, May 2012

[9] Will McGrath, Daniel Drew, Jeremy Warner, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, Bjoern Hartmann, Bifrost: Visualizing and Checking Behaviour of Embedded Systems across Hardware and Software", IJIST, Québec City, Canada, Oct 2017, page no.299-310

[10] Debugging system using "vivado" logic analyser, Basic Embedded System Design Tutorial

[11] Troubleshooting real-time software issue using a Logic analyser https://www.embedded.com/design/debug-and-optimization/4236800/1/Troubleshooting-real-time-software-issues-using-a-logic-analyzer.