Elysium Journal of Engineering Research & Management

P-ISSN: 2347-4408
E-ISSN: 2347-4734

DEVELOPMENT OF TARGET DEPLOYMENT PORT TO COLLECT CODE COVERAGE REPORT FOR PROCESSOR IN THE LOOP TESTING

Mr. Srinath Palani[1]

[1]*School of Information Technology, VIT University,*

Vellore – 632 014, Tamilnadu, India.
E-mail: srinathmsvit08@gmail.com

Abstract
Embedded computer systems are used with electronic system in order to increase performance, functionality, flexibility and accuracy of the system. This combination of computer and electronic chipset leads to several risks of failure which cannot be found out using traditional fault tolerance techniques. This scenario especially occurs in "Safety critical systems" where its failure could result damage to the property, environment or even loss of life. TMS320F28335is a processor with features like 32x32, 16x16 MAC operations and 16x16 DUAL MAC operations. The response time taken is very less which is desirable in case of safety critical systems. This paper comes up with a better tool for the verification and validation of such systems.

Keywords: Target Deployment port (TDP), Processor in the Loop Testing (PIL), Rational Test Real Time (RTRT), Code Composer Studio (CCSV4), Coverage Report, TMS320F28335.

1. INTRODUCTION

A well-known safety critical system is an aircraft using wire control, here the pilot uses a joystick to input commands and controls the computer, where in turn the computer manages the actual aircraft controls. Proper functioning of such a system is required as the aircraft carries hundreds of passengers in it. Therefore it is to be ensured that any implemented safety critical systems are completely safe.

This system aims at coming up with a better tool for the verification and validation of such systems, more specifically to Processor In the Loop (PIL) testing. PIL concept is applicable for Model Based Testing, where the Simulink models form the customer requirement and has to be implemented in a real-time system. As shown in the figure, the Developed Model (from the customer) generates the expected results and actual results are captured from the Designers Program (target system for the same test vectors. The discrepancies in the results show that the system is not safe to fly. The current proposal considers C code running on TMS320F28335 processor for implementation.

The system includes:

1. Enhance the PIL concept to work with TMS320F28335 based developer kit.

2. Come up with a Target Deployment Port (TDP) between IBM Rational Test Real Time (RTRT) and Code Composer Studio (CCS) IDE for TMS320F28335 target.

3. Use TDP to collect structural coverage metrics and let IBM RTRT generate a structural coverage report. Structural code coverage helps to identify dead code, unreachable code and uncovered decisions and conditions etc. This has to done in the PIL environment where Model and flight code are running in parallel with the test input.

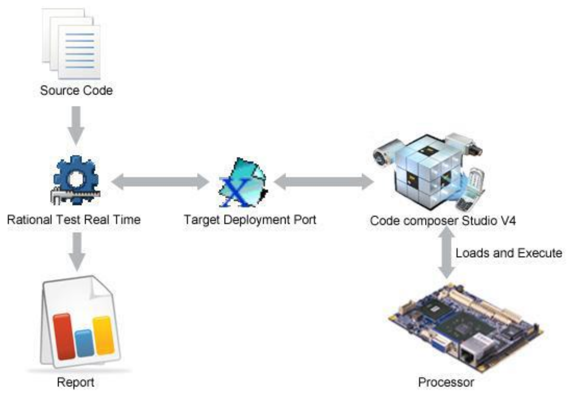


Fig. 1 The working diagram of the system.

2. TARGET DEPLOYMENT PORT EDITOR

Target Deployment Technology, all tests are fully target independent. Each cross-development environment - that is, every combination of compiler, linker, and debugger - has its own Target Deployment Port. Target Deployment Port (TDP) technology is constructed to accommodate your compiler, linker, debugger, and target architecture. Tests are independent of the TDP, so tests don't change when the environment does. Test script deployment, execution and reporting remain easy to use.

3. PROCESSOR IN THE LOOP TESTING

PIL (Processor in loop) testing environment works by exchanging data between the simulated host and the object code executing on the simulator. Pil simulation provides with functionality where the models verify the algorithms on target platform without manual creation of embedded test harness. Fig 2. Explains an environment where PIL tests are likely close to real time but they don't run on it as the execution of PIL code on target processor is done by Simulink controls. When the data is transferred to the target processor simulation halts at every sample period. Till the object codes execute on the processor the host simulation is paused, after its process is done data will be transferred to host. By this approach, the functional differences between model and compiled object code are checked and potential deviations in the performance are identified.

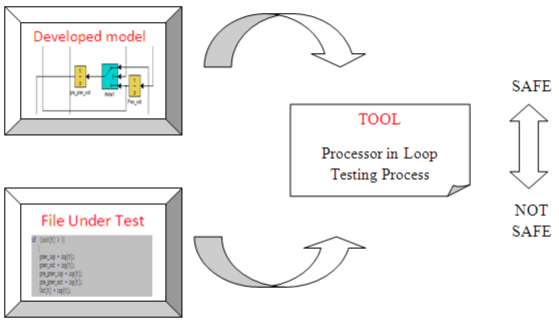


Fig 2. Working Model of PIL.

4. DESIGN

4.1 TARGET DEPLOYMENT PORT EDITOR

This tool allows you to create a new Target Deployment Port or edit an existing one. Editor is divided into four blocks which constitutes to BASIC, BUILD, LIBRARY, and PARSER SETTINGS as shown in fig 3.

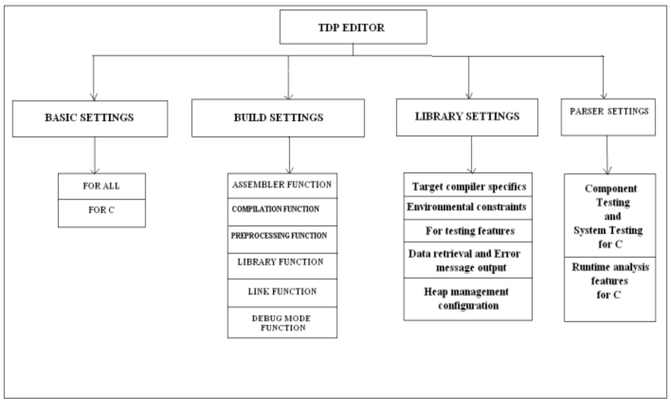


Fig 3. The modular design of various section of TDP.

1) Basic Settings: This section is to specify default file extensions, default flags, environment variables and custom variables required for the target architecture. The general syntax is: FIELDNAME=‖Value‖. The contents of any variable defined within this section may be referenced by the Perl functions in the Build Settings section using the Perl syntax Ini{'FIELDNAME'}. Common settings for all languages can be located in ―For All‖. This avoids the duplication.

The following fields are REQUIRED by the TDP being developed. The value is dependent on target environment, except for LANGUAGE, but if these fields are not set properly, the process will fail later. These fields are set in ―For c‖ block. The values set on RHS of equal to are the extensions and flags required by our processor TMS320F28335.

OBJEXT=obj

This field should contain the default object extension for object files. It may be something required by the compiler.

SRCEXT=c

This field represents the source extension that will be used to generate test drivers.

EXEEXT=out

This field contains the default extension for the executable test program.

ASMEXT=asm

This field contains the default extensions for assembler files.

LIBEXT=lib

This field contains the default extension for the library files.

DLLEXT=dll

This field contains the default extension for the Dynamic Link Library files.

DEFAULT_DEFINES=_DEBUG,__TMS320C28X__

The defines listed here will be set by default for all nodes referring to this Target Deployment Port in the Test Real Time user interface. This information may be modified through the Test Real Time user interface. These defines will be given to the compilation and pre-processing Perl function.

Standard includes, environmental paths and library paths are also set with respect to the target.

2) Build Settings: This section is used to configure the functions required for the integrated build process. This section can also be used to include all additional files needed by the TDP. All additional files are stored in the cmd subdirectory of the TDP. The subdirectory may be specified if needed. It has six functions for which the commands are given which contains their task.

2.1) Assembler Function: This function will be called to assemble $src. The generated object file is referenced by $out. $cflags are the assembler flags, $Defines contains defines, while include search paths are stored in $Includes. Uses cl2000 as assembler along with –g and –q as flags.

2.2) Compilation Function: This function will be called to compile $src. The generated object file is referenced by $out. $cflags are compilation flags, $Defines contains defines, while include search paths are stored in $Includes. Uses cl2000 as compiler along with -v28 --float support=fpu32 -g -pdr -ml –mt as flags.

2.3) Preprocessing Function: This function will be called to preprocess a source file. The preprocessed file name is stored in $out. $cflags are preprocessing flags, $Defines
Contains defines and include search paths are stored in $Includes. Uses cl2000 as preprocessor along with -ppl as flag.

2.4) Library Function: This function will be called to link the object files contained in the Perl array pointed to by $Objects and generate a libray file name as defined by the $exe variable. $ldflags are linker flags, $LibPath contains library search paths, while additional object files or libraries are stored in $libs. The link function should launch the linker with the appropriate options.

2.5) Link Function: This function will be called to link the object files contained in the Perl array pointed to by $Objects. The generated executable file name is stored in $exe. $ldflags are linker flags, $LibPath contains library search paths, while additional object files or libraries are stored in $libs. The link function should launch the linker with the appropriate options. Uses cl2000 as linker along with -q -heap0x100 -stack0x1000 –x as flag.

P-ISSN: 2347-4408
E-ISSN: 2347-4734

2.6) Debug mode Function: This Perl function should launch the target system debugger. The goal is to enable the user to interactively debug the test program or the instrumented application. This function is optional, and it is usually very difficult, unless in a Standard-Mode, to generate the test report while debugging interactively at the same time. If the method used to retrieve the value of atl_buffer is based on a debugger script executed in an automated way, user-defined breakpoints may break this automation. This is not a problem as long as the differences between an interactive debugging session and a batch execution of a test program or of an instrumented application are clear.

2.7) E3xecution Function: This function should execute the program whose name is given as $exe parameter. When returning, the Intermediate Report should have been generated. The name of the Intermediate Report file is given the parameter $out. This routine should block until the Intermediate Report is generated. It should launch the test program or should launch some process that will execute the program (a shell window, simulator or a debugger) with the proper options or command script.

3) *Library Settings*: In this section, all the library configuration settings are accessible. For each setting, the Scope field specifies the applicable Test Real Time feature. All the compiler-dependent settings and compiler dependent data types must be configured.

4) *Parser Settings*: This section enables modification of the parser configuration files. The parser configuration files are used to modify the default behavior of the parser to address. The parser setting is made of options, #pragmas and #defines that you might need to parse specific keyword used by your compiler.

4.1) Component Testing and System Testing for C: Parser modifications are necessary to support the Component Testing and System Testing features for the C language.

4.2) Runtime analysis features for C: This section is used to configure the parser settings for the runtime observation features of Rational Test RealTime. The content of this section is written into the ana/atct.def file and only used by the attolcc1 instrumentor.

5. HIGH LEVEL DESIGN

5.1 Data Flow diagram

The data flow takes place between two interface namely Rational Test Real Time and Code Composer Studio as shown in fig 4.

6. SIMULATION RESULT

In the first step the TDP Editor is taken in which required settings are made. On selecting save and generate option from file menu the tdp is saved. It is saved in the following path
C:\Program Files\Rational\TestRealTime\targets\xml
The target directory generated from the xdp(TDP) contains
the required libraries and the commands to be followed while testing the code against the processor. It is generated in C:\Program Files\Rational\TestRealTime\targets

Fig.5. shows the TDP editor which is being instructed ―save and generate‖ which generates the files and folders to the above given paths.

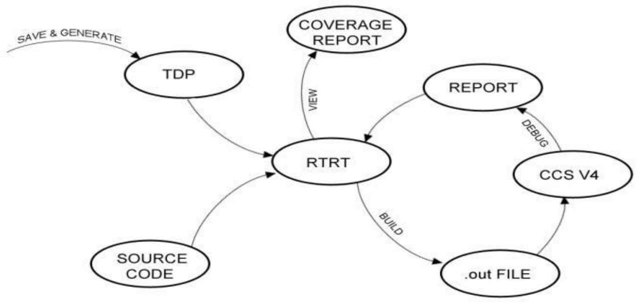
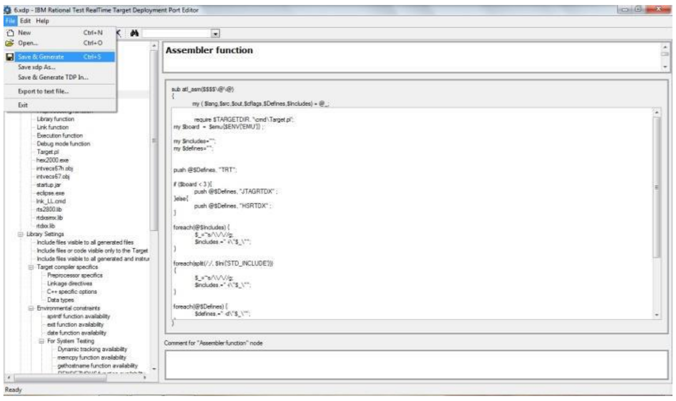


Fig 4. Data flow diagram of the entire system.

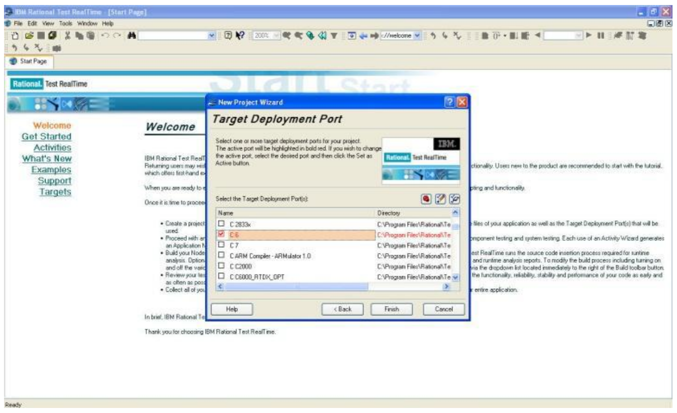


```
sub atl_asm($$$$\@\@)
{
        my ( $lang,$src,$out,$cflags,$Defines,$Includes) = @_;

        require $TARGETDIR. "\cmd\Target.pl";
my $board = $emu{$ENV{'EMU'}} ;

my $Includes="";
my $defines="";

push @$Defines, "TRT";

if ($board < 3 ){
        push @$Defines, "JTAGRTDX" ;
}else{
        push @$Defines, "HSRTDX" ;
}

foreach(@$Includes) {
        $_=~s/\\/\//g;
        $Includes .=" -I\"$_\"";
}

foreach(split(/;/, $ini{'STD_INCLUDE'}))
{
        $_=~s/\\/\//g;
        $Includes .=" -I\"$_\"";
}

foreach(@$Defines) {
        $defines .=" -d\"$_\"";
}
```

Comment for "Assembler function" node

Fig 5. Target Deployment port Editor.

In RTRT user interface the tdp is selected from list of TDP's available in xml folder while creating a project for
testing. The required c code .c files along with the extension files like .ptu/.mat/.mdl/.csv are fed up to the project.



Select one or more target deployment ports for your project.

Select the Target Deployment Port(s)

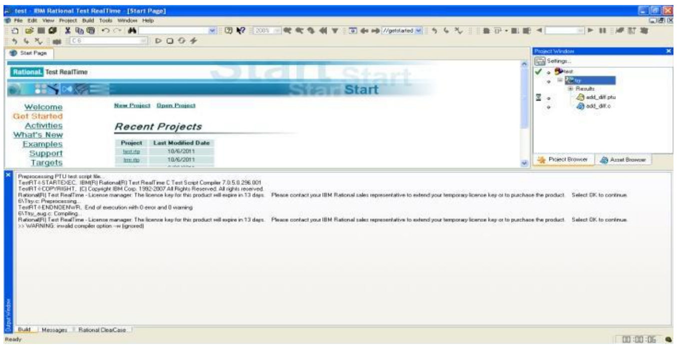
Name	Directory
C 2833x	C:\Program Files\Rational\Te
C 6	C:\Program Files\Rational\Te
C 7	C:\Program Files\Rational\Te
C ARM Compiler - ARMulator 1.0	C:\Program Files\Rational\Te
C C2000	C:\Program Files\Rational\Te
C C6000_RTDX_DPT	C:\Program Files\Rational\Te

Thank you for choosing IBM Rational Test RealTime.

Fig 6. RTRT user interface.

After adding the source codes .c and .ptu's (file extensions) the project is built using BUILD option and the .out file is generated.

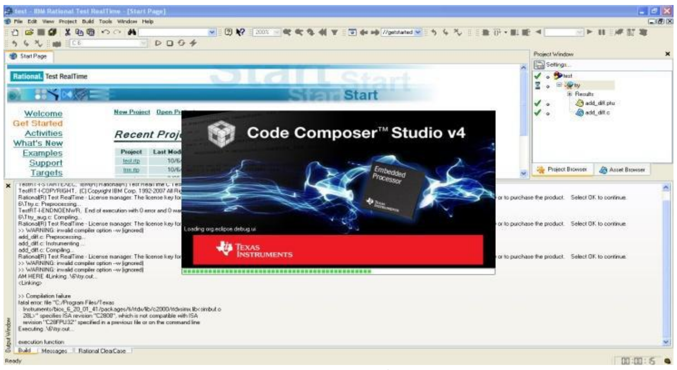
P-ISSN: 2347-4408
E-ISSN: 2347-4734



```
Preprocessing PTU test script file...
TestRT-I-STARTEXEC.  IBM(R) Rational(R) Test RealTime C Test Script Compiler 7.0.5.0.296.001
TestRT-I-COPYRIGHT.  (C) Copyright IBM Corp. 1992-2007 All Rights Reserved. All rights reserved.
Rational(R) Test RealTime - License manager: The licence key for this product will expire in 13 days.    Please contact your IBM Rational sales representative to extend your temporary licence key or to purchase the product.    Select OK to continue.
6\Ttry.c: Preprocessing...
TestRT-I-ENDNOENWR.  End of execution with 0 error and 0 warning
6\Ttry_aug.c: Compiling...
Rational(R) Test RealTime - Licence manager: The licence key for this product will expire in 13 days.    Please contact your IBM Rational sales representative to extend your temporary licence key or to purchase the product.    Select OK to continue.
>> WARNING: invalid compiler option -w (ignored)
```

Fig 7.Building project in RTRT.

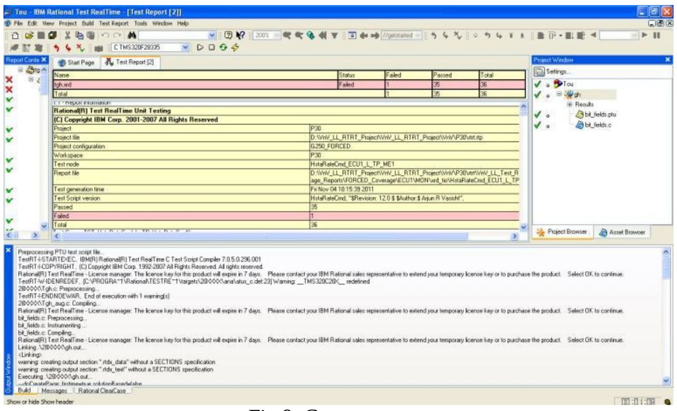
The generated .out file is automatically fed to the CCS as soon as it is invoked.



```
TestRT-I-COPYRIGHT.  (C) Copyright IBM Corp. 1992-2007 All Rig
Rational(R) Test RealTime - License manager: The license key for                  or to purchase the product.    Select OK to continue.
6\Ttry.c: Preprocessing...
TestRT-I-ENDNOENWR.  End of execution with 0 error and 0 wa
6\Ttry_aug.c: Compiling...
Rational(R) Test RealTime - License manager: The license key for                  or to purchase the product.    Select OK to continue.
>> WARNING: invalid compiler option -w (ignored)
add_diff.c: Preprocessing...
add_diff.c: Instrumenting ...
add_diff.c: Compiling...
Rational(R) Test RealTime - License manager: The license key for                  or to purchase the product.    Select OK to continue.
>> WARNING: invalid compiler option -w (ignored)
>> WARNING: invalid compiler option -w (ignored)
AM HERE 4Linking .\6\try.out...
<Linking>

>> Compilation failure
fatal error: file "C:/Program Files/Texas
   Instruments/bios_6_20_01_41/packages/ti/rtdx/lib/c2000/rtdxsim.lib<simbuf.o
   28L>" specifies ISA revision "C2800", which is not compatible with ISA
   revision "C28FPU32" specified in a previous file or on the command line
Executing .\6\try.out...

execution function
```

Fig 8.Invoking of CCS V4.

In CCS the debug uses the .out file generated by RTRT and generates the coverage report for the given code which can be viewed in RTRT interface window.



Name	Status	Failed	Passed	Total
tgh.xrd	Failed	1	35	36
Total		1	35	36

Rational(R) Test RealTime Unit Testing	
(C) Copyright IBM Corp. 2001-2007 All Rights Reserved	
Project	P30
Project file	D:\VnV_LL_RTRT_Project\VnV_LL_RTRT_Project\VnV\P30\rtrt.rtp
Project configuration	G250_FORCED
Workspace	P30
Test node	HstaRateCmd_ECU1_L_TP_ME1
Report file	D:\VnV_LL_RTRT_Project\VnV_LL_RTRT_Project\VnV\P30\rtrt\VnV_LL_Test_R age_Reports\FORCED_Coverage\ECU1\MON\xrd_tio\HstaRateCmd_ECU1_L_TP
Test generation time	Fri Nov 04 10:15:39 2011
Test Script version	HstaRateCmd, "$Revision: 12.0 $ $Author:$ Arjun R Vasisht".
Passed	35
Failed	1
Total	36

```
Preprocessing PTU test script file...
TestRT-I-STARTEXEC.  IBM(R) Rational(R) Test RealTime C Test Script Compiler 7.0.5.0.296.001
TestRT-I-COPYRIGHT.  (C) Copyright IBM Corp. 1992-2007 All Rights Reserved. All rights reserved.
Rational(R) Test RealTime - License manager: The license key for this product will expire in 7 days.    Please contact your IBM Rational sales representative to extend your temporary license key or to purchase the product.    Select OK to continue.
TestRT-W-IDENREDEF.  [C:\PROGRA~1\Rational\TESTRE~1\targets\28XXXX\ana\atus_c.def:23] Warning: __TMS320C28X__ redefined
28XXXX\Tgh.c: Preprocessing...
TestRT-I-ENDNOEWAR.  End of execution with 1 warning(s)
28XXXX\Tgh_aug.c: Compiling...
Rational(R) Test RealTime - License manager: The license key for this product will expire in 7 days.    Please contact your IBM Rational sales representative to extend your temporary license key or to purchase the product.    Select OK to continue.
bit_fields.c: Preprocessing...
bit_fields.c: Instrumenting ...
bit_fields.c: Compiling...
Rational(R) Test RealTime - License manager: The license key for this product will expire in 7 days.    Please contact your IBM Rational sales representative to extend your temporary license key or to purchase the product.    Select OK to continue.
Linking .\28XXXX\gh.out...
<Linking>
warning: creating output section ".rtdx_data" without a SECTIONS specification
warning: creating output section ".rtdx_text" without a SECTIONS specification
Executing .\28XXXX\gh.out...
```

Fig 9. Coverage report.

The global coverage of the code is also provided as in.

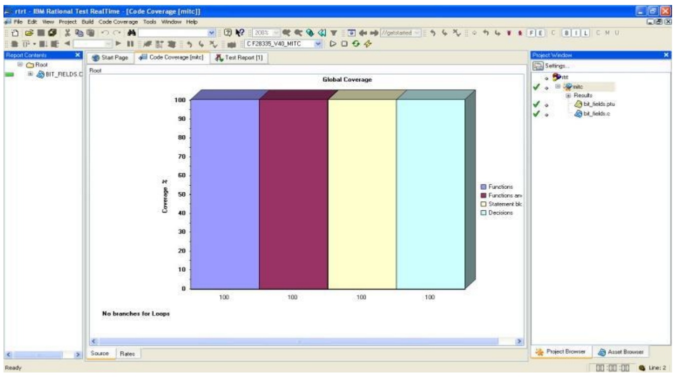


Fig 10. Global Coverage

7. CONCLUSION

An interface between RATIONAL TEST REAL TIME (RTRT) software and CODE COMPOSER STUDIO (CCSV4) can be developed to collect the code coverage report for the processor TMS320F28335 by using the Processor In the Loop (PIL) Testing.

REFERENCES

[1] Embedded software design for safety critical systems, Knowledge transfer network electronics, UK, Nov. 2009.

[2] RTRT-TDP Guide by Rational Software Corporation. Architecture of the TMS320F28335 by Frank Bormann.

[3] D.S. Loubach, J.C.S. Nobre, A. M. da Cunha, L.A.V. Dias, M. R. Nascimento, W. A. Dos Santos, "Testing Critical Software: A Case Study for an Aerospace Application" *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*.

[4] S. Qureshi "Book Review [review of "Embedded Image Processing on the TMS320C6000 DSP: Examples in Code Composer Studio and MATLAB," in *Signal Processing Magazine, IEEE, 2007*.

[5] P. Meena, K.U. Rao, D. Ravishankar, "Real-time detection and analysis of PQ disturbances with DSP using matlab embedded link to code composer studio," *IEEE Conference Publications, 2009*.

[6] H. Nagaishi, M. Fukui, Asakura, Hisao, Sugimoto, Aritoshi, "Defect reduction in Cu dual damascene process using short-loop test structures", *Semiconductor Manufacturing, IEEE Transactions on Volume:16, Issue: 3, 2003*.

AUTHORS

Mr. Srinath Palani received his M.S (Integrated) in Software Engineering from VIT University, Vellore in 2013. He is certified professional by Microsoft and SAP AG. He is presently working as SAP Consultant in leading IT sector. He is also part of Innovation Team in his organization and working on solutions for various challenging problems.