The Aplication Of Testbed Tbrun In Unit Test Of Aviation Software

Yuan Li
Research institute of electronic Science and technology
University of electronic science and technology of China
Chendu, China
75460833@qq.com

Dan Liu
Research institute of electronic Science and technology
University of electronic science and technology of China
Chendu, China
349194719@qq.com

Abstract— **Based on actual project of unit test,it disscussed the application of testbed in unit test of aviation software.It introduced the configure of three general aviation software's development environment in testbed,and some modifications of code before test.It analysed steps of unit test and several common problems during steps,in addition,it introduces defect of testbed in unit test. Each function can be verified after unit test,at the same time,other test processes are guaranteed.**

Keywords- testbed tbrun;unit test;configure of the environment

I. INTRODUCTION *(HEADING 1)*

The requirements of Aviation software for reliability and security is topmost ,Either a software error are likely to cause significant economic losses and casualties, Therefore, we must to do the most rigorous and demanding test to it. Unit testing is an important part of the software testing process, it can verify the correctness of features of each function and the consistency of the detailed design description. It can provides a certain basis for reliability and security of software and reduce the overhead of software testing.

II. UNIT TEST METHOD AND TOOL SELECTION

To detect the logic error of every function is the goal of unit test, the method of unit test is divided into static test and dynamic test, static test includes document review, static analysis and code review, dynamic test generally uses white-box test, auxiliary to black-box test.

Aviation Software is embedded software, its development platform environment is Tornado, CCStudio, VisualDSP++.Before unit test, we must ensure that the function can be measured. First of all, we should ensure that the function can run independently, First, it must be isolated from other code ,using pilling; Second isolated from dependent system to resolve differences between compilation environment and platform.

Testbed developed by LDRA can solve the two problems which we face, first, it can automatically piling , second, it can compile on different development environment, to ensure the consistency with code compiled platform.

Tbrun module of it is designed to do unit testing, which enables engineers to input easily enter the input and output data, automatically generate test drive no need to write test scripts. In addition, It can automatically generate stubs, engineers simply fill in the code stubs directly according to the need of test cases. After completed, we can directly see the test coverage analysis chart. In a word, it can improve the efficiency of software test and save engineer's time and energy. Here we briefly introduce Aviation Software test process and notes.

III. EASE OF USE CONFIGURATION OF SEVERAL COMMON DEVELOPMENT PLATFORM ENVIROMENT

During the test, the diriver execution environment needs to be consistent with code compilation environment, so we need to configure compilation environment of testbed as the code compile environment before the unit test. Embedded development environment mainly has the following three categories: Tornado,CCStudio,VisualDSP++， then we make a brief introduction for three environment configuration. In the process, TBConfig is essential.

Enviroment configuration of Tornado:

1) Open the Tornado configuration interface in TBConfig, select testbed and Tornado's installation path, fill in the host name, keep the default vaule,then click OK;

2) In the testbed installation path, open Vxsim_build.bat in Vxwoks file, fill the path of header files in it,and plus "PAUSE" at the end of file, such we can easily view error message.

Enviroment configuration of CCS:

1) Open the CCS configuration interface in TBConfig, choose testbed and CCS's installation path, keep the default vaule, then click OK;

2) In Setup CCStudio, set CPU for device simulator which code run on, and delete the value of GEL File;

3) In the testbed installation path, open Tic28xx_Compile.bat in Tic28xx file, fill the path of header files in it, and plus "PAUSE" at the end of file, such we can easily view error message.

Enviroment configuration of Visual DSP++:

1) Open the CCS configuration interface in TBConfig, choose testbed and Visual DSP++'s installation path, specially attention to Build option, fill in the full path of cc21k,keep the default vaule, then click OK;

2) Open Visual DSP++ development environment, create a new session, which be consisitent with development model;

3) Increase the environment variable path, if Visual DSP++ is installed on the D drive, path=" D:\ VisualDSP++\System;D:\ VisualDSP++";

4) In the Testbed installation path, open the Ad21060tcl.tem in Visual DSP file, if the session is ADSP-

21060, ill the path of header files in it, such we can easily view error message.

IV. STEPS AND PRECAUTIONS OF UNIT TEST USED TBRUN

Before test, judge whether the test code has infinite loop,such as while(1),while(TRUE).if it has infinite loop, we must modify it, otherwise, test cases will always perform and can not quit, we generally use "for" loop statement to instead, design cycle numbers according to the actual situation of the code.

In embedded code, we often encountered that the code get the value from the hardware, generally, we redefine a same type variable instead of it, then assign value to it ,for example:

```
#define SCICTL2 ((unsigned int *)          0x7054)
    Fun()
{      ……
while((*SCICTL2&0x80)!=0x80);
……
}
```

For *SCICTL2,we need to redefine a variable such as unsigned int SCICTL2_test;the code is modified accordingly:

```
Fun()
{      ……
while((SCICTL2_test)!=0x80);
    ……
  }
```

After the code is modified, we can select a file to start test, then select the file storage path during testing process, and select the corresponding header file path of tested code, choose "unit test only" to enter Tbrun interface, following steps can be roughly summarized as the following steps, as shown in Figure 1:

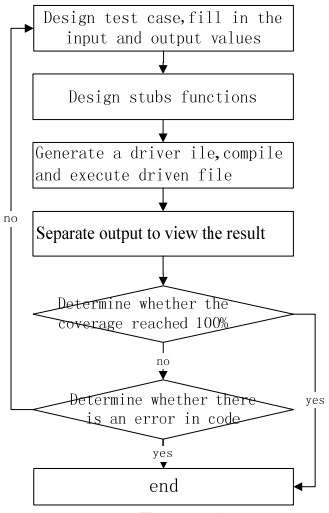

Figure 1

A. *Design test cases*

Before unit test, we understand the function of each function by reading the software detailed design document. On the basis of understanding the I/O conditions and logical structure, we design test cases which can sufficiently find hidden error as little as possible. Each example is a "profile" which the program run at a certain moment, we fill the input and output values of cases according to the program running path, we need to pay special attention to the design of values of array and pointer. Array usually has a lot of numbers, it's a waste of time that we input the values manually, so, when we click the right button of mouse to create a new test case, we generally choose "Remove all elements from the test case", such as Figure 2, and use "for" loop statement to initialize arrays.

Figure 2

When we design the initial value of pointer, we can't directly assign an address to it, but redefined same type global variables, then assign the address of the variable to it, then we assign values to global variables. For example:

```
void SubmenuProc(INT32 No, BP_D10_ST* BP_D10_pst)
                {          ……
        if(BP_D10_pst->NotInSubmenuOrApplication == 1)
          ……
                }
```

BP_D10_ST* BP_D10_pst is a parameter of the function, there is an input parameter: BP_D10_pst, which TBrun generates, because it is not global variables, we need to redefined a same type parameter, such as: BP_D10_ST g_BP_D10_pst,then we assign &g_BP_D10_pst to BP_D10_pst,and assign value to g_BP_D10_pst,for example:g_BP_D10_pst.NotInSubmenuOrApplication = 1;

B. *Set stubs for code*

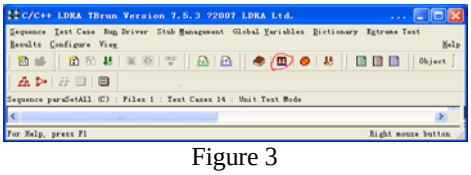

Figure 3

Just click the button in Figure 3, it can pile all stub functions, if some stubs need some special treatment, click the right button of mouse, then choose "set code segment" to design stubs.

C. *Generate,compile and execute drives files*

In unit test process, if there is an error in compile and execute process, we usually open sequence work directory to

deal with failure, for example, in Figure 4,open the .dyn file, view the running print message. It is the last message where the error usually appears, then we can quickly locate the error. After, we add some print message according our analysis in it, save it and recompile it in Testbed, then view the contents of .dyn files and find the problems, so we can find solutions.

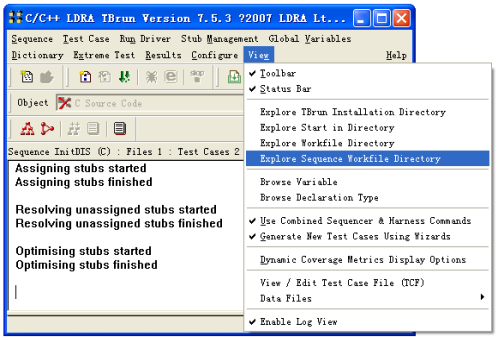

Figure 4

For example:

```
DIS_51zzqqzz (     3 );
{
  InitSerialPort ( 2 ) ;
 }
InitVolumeValue ( DISNo ) ;
memset ( & s_BPUAgreeDISSubmenu [ DISNo ] , 0 , sizeof ( BPDIS_CMD11_ST ) ) ;
InitDISRecvProc ( DISNo ) ;
semTake( SEM_Q_PRIORITY , 5 ) ;
DIS_51zzqqzz (     4 ) ;
```

Its .dyn file content is as follows:

```
ldra S N 1 64 1 0 V
ldra S I V Z DISNo
ldra S E 1
ldra EXH DIS_64.exh
ldra     1
ldra     3
```

semTake(SEM_Q_PRIORITY , 5) is likely to be the location of error, add print statement in the front and back of it, such as: DIS_51zzqqzz (33333). When the print statement is in the front, recompile and execute, if it is printed in .dyn file, and it is not printed when the print statement is in the back, so we can determine the error is caused by semTake(SEM_Q_PRIORITY , 5).

D. Separate output and analyze the results

Comparing the actual output value and the expected output value, if the test cases fail, we check the program execution path combined with the coverage maps to find the error. Click the node in the graph, we can see the executed code and see the front and back code of the node by clicking “previous” and “next”, so we can find the position of error. Figure 5 is a part of coverage graph.

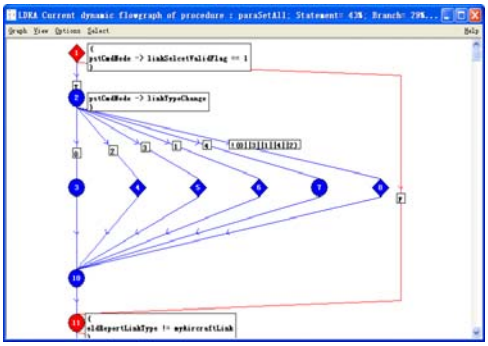

Figure 5

E. Repeate the previous four steps

Repeat the previous four steps until the combined coverage of branch and statement is 100% unless the code itself is abnormal. In addition to it, it is a disadvantage of Testbed that Testbed simulate some code to run, such as : interrupt problem,which may make the coverage is less than 100%.For example, the following code:

```
While(1)
{
if(count_LED_period>250)
        {
                ……
                count_LED_period=0;
                if(count_delay<3)
                        count_delay++;
                else if(flag_delay==0)

                        flag_delay=1;
        }
}
```

The value of count_delay can be added only once ostensibly when if branch is true, as the value of count_LED_period satisfies the conditions, but he value of count_LED_period will be changed by interrupt procedures because count_LED_period is a global variable, so that the if branch may be true again. Testbed does not allow infinite loop so the value of count_LED_period will not be changed by interrupt procedures.

V. CONCLUSION

Testbed greatly improve the efficiency of Aviation software’s unit test and save a lot of consumption, at the same time, it save a lot of time on document work, so engineers can quickly and accurately locate errors.

ACKNOWLEDGMENT

I am very grateful to Ningbo major scientific and technological research: The development and application of system monitoring internet public opinion , which funded me. It’s number is 2011C51007.

Funded project: The development and application of system monitoring internet public opinion, Number: 2011C51007

REFERENCES

[1] Shuang Cao ,Research on Automatic Generation Method of Test Case for Aviation Software, Nanjing University of Aeronantics and Astronautics Master's degree thesis,2010.

[2] Liu Yuanyuan, Implementation of aircraft control software's unit test, Beijing University of Posts and Telecommunications Master's degree thesis,2010.

[3] Xu Ke, The research of embedded software test, University of Electronic Science and Technology of China Master's degree thesis,2006.

[4] Wang yu, He yongjun, The application of Testbed/Tbrun in embedded software unit test, Acoustics and Electronics Engineering,2006.

[5] Zou Huirong, Coverage test of aircraft central maintenance system based on LDRA Testbed, Aeronautical Computing Technique,2010.

[6] Li Zhongping, Yue Hai, Xue Jing,The application of Testbed in Aerospace software, Aerospace Control,2007.

Published by Atlantis Press, Paris, France.