

# Tools for Software Testing in ARM<sup>®</sup> IP-based Embedded Systems

By Magnus Unemyr, VP Sales/Marketing and Mark Moran, US General Manager Atollic

Releasing a product with bugs is potentially very expensive when costs of field upgrades, recalls, repairs, etc. are considered. Less quantifiable, but equally important is the diminution of a company's reputation and loss of good will. Yet, many products based on embedded systems are released without all of the testing which is necessary and/or desirable to minimize these problems.

For those developers of embedded projects who choose to employ automated in-target software testing, the challenge is to find tools that are well integrated into their embedded development environment.

Most existing software testing tools can only execute the tests on a PC. This is of limited use when testing an embedded application because the actual system will have different hardware interfaces, different timing issues, memory constraints, target specific #pragmas, inline assembly, incompatible linker configuration files and other differences. Consequently, it is of considerable importance to run as many software tests as possible on the actual embedded hardware so as to minimize inter-platform differences.

New tools for embedded test automation emerge on the market, such as Atollic TrueVERIFIER<sup>™</sup> that analyzes the source code of the application to test, auto-generates a test suite, and run it automatically in an ARM IP-based target board.

Once testing is completed, a test report showing 100% test success may not be the reason for celebration that one might expect. This is because the tests that were run might be considered successful, but the test procedure itself might not have been applied to more than a small fraction of the code. Understanding the quality of your test procedures thus become critical in judging whether or not the product is tested well enough before release. Dynamic execution flow analysis can be used to perform code coverage analysis, which is a means of measuring test quality. Atollic TrueANALYZER<sup>®</sup> handles the most stringent types of code coverage analysis with almost no extra effort from the developer or tester.

Both the Atollic TrueVERIFIER<sup>™</sup> and Atollic TrueANALYZER tools are deeply integrated into the Atollic TrueSTUDIO<sup>®</sup> C/C++ development and debugger IDE, thus creating an efficient environment for developers of ARM IP-based embedded systems.

## Unit Testing

Most embedded applications are not the subject of a formal test methodology, although more and more teams are looking into the use of this methodology. Unit tests are those tests which exercise your C functions, by calling them with different combinations of input parameter values, to drive the code through different execution paths of the function.

Writing unit tests by hand is labor intensive, costly, tedious, and still leaves the possibility that many of the possible execution paths are not tested. To guarantee that as many as possible of the important execution paths throughout a function would be exercised by a test suite, requires a detailed analysis of how the input parameters drive the code in various directions. This is very time consuming and difficult or impossible for anything but trivial functions.

Maintaining the synchronization of unit tests with code under development is another challenge that development teams face. This problem is often exacerbated by the fact that schedules are tight and frequently become compressed as the project progresses. If source code and unit tests thus get out-of-synch, the unit tests will become less useful, especially at the time that they are most needed.

The unit test tools available to PC developers are less useful to embedded developers as they rarely manage compilation, downloading and execution of the test suites in embedded boards. Tools that create the unit tests automatically, build them into the embedded application and allow you to run them on the target via a convenient pathway such as a JTAG debugger will result in optimal usage of time and maximum productivity.

An effective approach is to use a full embedded test automation system that are integrated right into the C/C++ development IDE so that synchronization is easy to maintain, and ancillary tools such as JTAG debug probes can be used to full effect. Such tools have not been common in the embedded industry previously, but new tools like Atollic TrueVERIFIER now meet these demanding criteria and bring very powerful in-target test automation capabilities to ARM software developers.

### Automatic generation of unit tests

The new breed of professional embedded systems test automation tools can analyze the source code and generate and execute suitable test suites automatically in an ARM® target board.

Atollic TrueVERIFIER auto-generates a test suite (implemented in C source code), that makes many function calls with different combinations of input parameters, thus driving a large number of different execution paths in the functions to be tested.

The illustration below shows how a trivial C function can be tested, by calling it many times with different input parameters. The selection of input parameter values (the test data) is selected

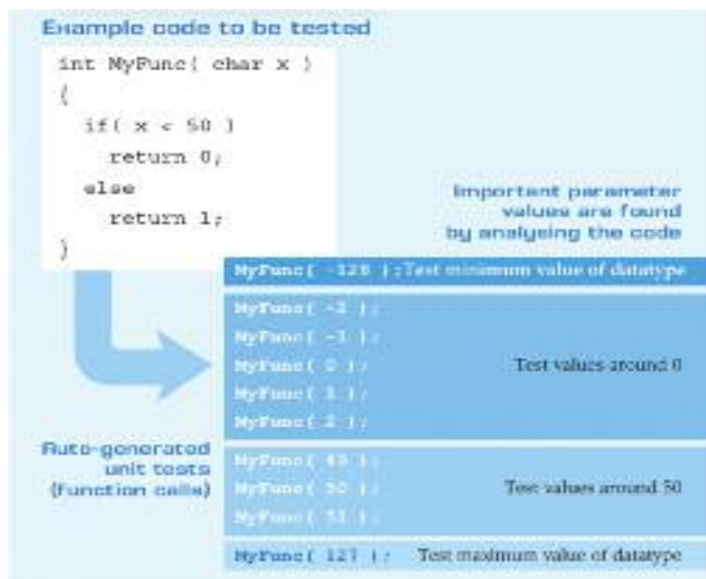


Figure 1: Automatic generation of unit tests.

and generated automatically by the testing tool. Atollic TrueVERIFIER is a powerful tool for embedded test automation. It analyzes the source code of the application and automatically generates unit tests with important combinations of input parameter values, in order to drive as many important combinations of execution paths in the function as possible. To maintain full flexibility in case special cases need to be addressed, it is also possible to create test cases manually with custom sets of input parameter values.

### Automatic execution of unit tests

Once the test suite has been generated (as C source code), it must be compiled, linked and executed on the target system.

Many low-cost unit test tools exist that only run unit tests on a Windows® PC. The inability of these tools to integrate with the embedded tool development environment, results in additional inefficiency of not being able to execute the tests on the target board.

Atollic TrueVERIFIER integrates seamlessly into a professional embedded IDE, enabling easy synchronization of test with code development and takes advantage of integration with ancillary tools such as JTAG debug probe.

TrueVERIFIER integrates directly into Atollic TrueSTUDIO, a modern state-of-the-art IDE designed explicitly to serve the needs of professional developers, especially those operating in teams. Once the unit tests have been generated in C source code, they are compiled automatically using the integrated embedded build tools. They are then downloaded to the target board using the same JTAG probe that works with the IDE that is being used for normal debugging. Finally, execution is performed in the target board with dynamic execution flow analysis to measure the achieved code coverage.

The testing tool system is shown in Figure 2.

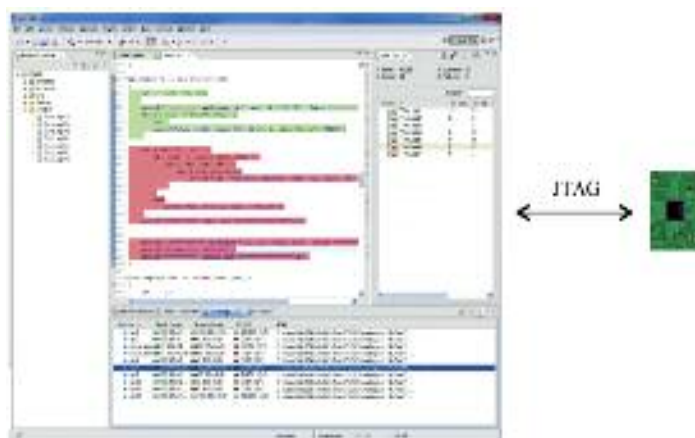


Figure 2: Atollic TrueVERIFIER executes in-target testing with a JTAG probe.

Once the test suite is completed, test results and code coverage information is uploaded to the IDE. Atollic TrueVERIFIER measures code coverage on MC/DC level (see the next section for details), the test quality level required for flight control system software. In effect, a professional embedded systems test automation tool like Atollic TrueVERIFIER automates the following tasks:

- Analyzing the source code to find out what combinations of input parameter values affect the execution flow.
- Generation of test suites (in C source code) that calls the functions to be tested many times with different combinations of input parameter values, thus driving different execution paths in the functions being tested.
- Building (compilation and linking) of the test suite source code.
- Downloading the built test suite into the target board, using the same JTAG probe as is used for debugging.

- Execution of test suites in the target board with execution flow monitoring, thus enabling advanced code coverage analysis to measure the achieved test quality.
- Uploading and visualization of test results and achieved code coverage.

**Measuring test quality**

Once code is developed and tested, the focus shifts to quantitatively understanding what actually happened during the testing. It is of no use to conclude that testing is completed successfully if only a small part of the code was tested.

Code coverage measurement, which is performed using dynamic execution flow analysis, is commonly used to study what parts of the code have been tested. This therefore is a direct measure of the test quality. There are many different types of code coverage analysis, from very simple analysis up to very stringent types. It is a very common error among software testers to believe that simple analysis suffices, when in fact the more stringent analysis methods are often required to reveal if the product has been properly tested or not.

Code coverage analysis has been classified formally. The more advanced types of code coverage analysis (such as MC/DC described below) are often used for measuring test quality of safety critical software.

As an example, RTCA DO-178B (a standard for development of flight safety critical software) requires MC/DC testing of software on "Level-A criticality", the most critical part of airborne software, where a software error can lead to a catastrophic situation with loss of aircraft or human lives.

Many projects also outside the aerospace industry would benefit from better control of what has been tested. In particular this is valid for companies with high production volumes, or products that are difficult or expensive to upgrade in the field. The same goes for products where the supplier wants to keep its good reputation and where loss of goodwill can be costly for the company.

**Different types of code coverage analysis**

Before elaborating on the different types of code coverage analysis that can be done, consider the code example in Figure 3.

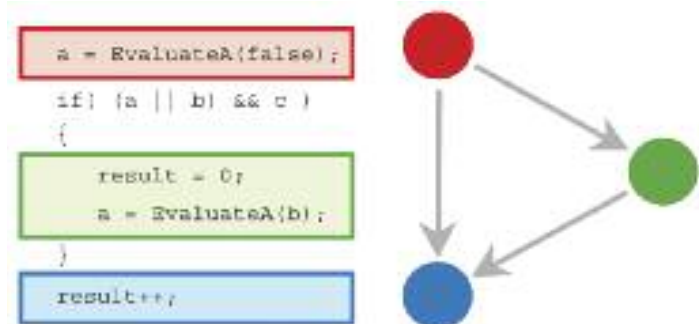


Figure 3: Also trivial code sections are difficult to test rigorously.

The trivial code section above contains three code blocks; a red code block that is always executed, a green code block that is sometimes executed dependent on the branch decision made in the if-statement, and a blue code block that is always executed. The code section above can be visualized as an execution flow diagram. We can see that this code section yields two potential execution paths, one directly from the red to the blue code block, and another one that also passes through the green code block. The branch decision taken in the if-statement will drive the selection of which of the two execution paths will be taken.

**Statement or block coverage**

Statement or block coverage only measures how many of the C-statements or code blocks have been executed during a test session. It does not measure how branches in the execution flow affect which statements or code blocks become executed.

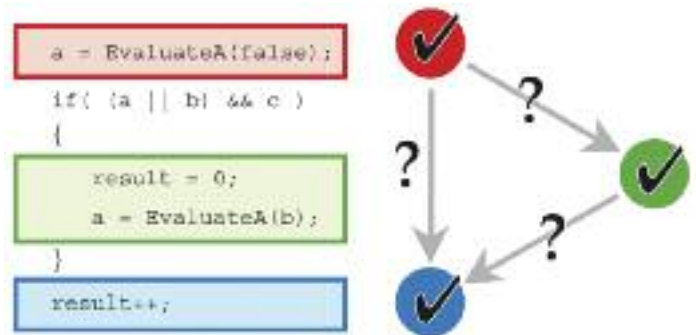


Figure 4: Statement coverage and Block coverage.

**Function coverage**

Function coverage only measures which or how many of the C-functions have been called during a test session. It does not measure which or how many of the function calls in a code section is actually executed, or the quality of the testing of the function itself.

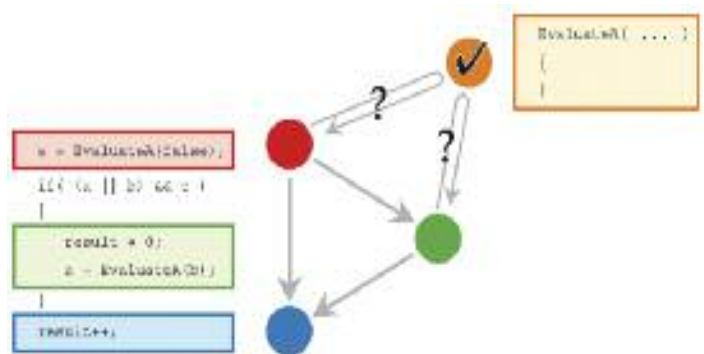


Figure 5: Function coverage.

**Function call coverage**

Function call coverage measures which or how many of the function calls in a code section have actually been called during a test session.

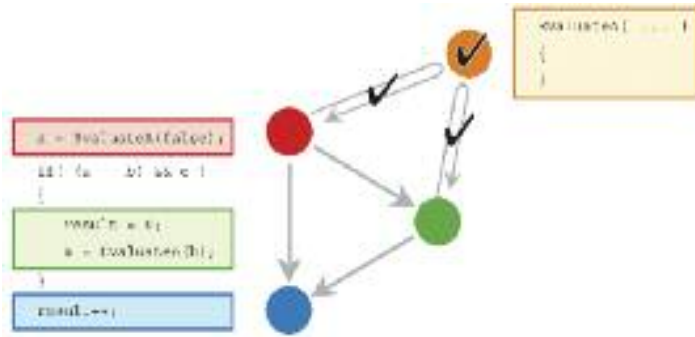


Figure 6: Function call coverage.

**Branch coverage**

Branch coverage measures whether all code blocks and alternate branch paths have been executed in a code section (such as both the if- and the else- part in an if-else statement).

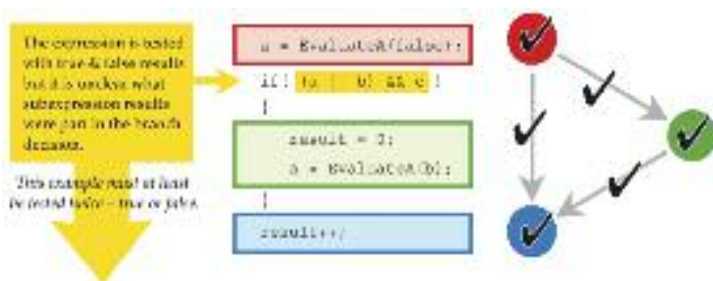


Figure 7: Branch coverage.

Branch coverage typically requires a code section to be executed a sufficient number of times, so that all alternative branch directions will be tested. As all branch paths must be executed, all corresponding code blocks are executed as well.

**Modified condition/decision coverage**

Modified condition/decision coverage (MC/DC) is a very advanced type of code coverage analysis. This kind of code coverage is applied to applications of which the highest reliability is expected. It extends Branch coverage with the additional requirement that all sub-expressions in complex decisions (such as in a complex if-statement) must drive the branch decision independently of the other sub-expressions.

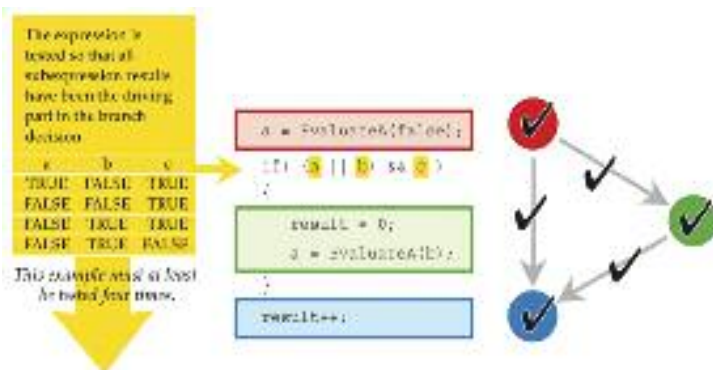


Figure 8: Modified condition/decision coverage (MC/DC coverage).

As can be seen from the illustration below, the code section must be executed several times with different values of sub-expressions a, b and c, to ensure that all code blocks and branch paths have been executed, and to ensure that all sub-expressions (a, b and c) have been the driving force in the overall branch decision independently of the other sub-expressions.

Modified condition/decision coverage (MC/DC) is a very stringent type of code coverage analysis, and is in fact required for some types of safety critical software. Flight control system software testing must for example fulfill MC/DC coverage.

**Tools for test quality measurement**

Up until now, it has been difficult to find tools for measurement of test quality in embedded systems software. The few tools that have existed have been limited by one or more of the following problems:

- Only testing using weak types of code coverage analysis is used
- Only testing in PC environments and not on the embedded target
- Very expensive
- Difficult to use
- Lacking integration with other embedded development tools

Atollic TrueANALYZER is a highly integrated new tool that changes the situation dramatically. Atollic TrueANALYZER supports a wide range of code coverage analysis types as shown below, including MC/DC.

One of its most powerful features is the fact that the analysis is conducted right on the embedded target. Atollic TrueANALYZER is truly easy to use with the following set of test quality measurements conducted in-target using only two mouse clicks:

- Statement/block coverage
- Function coverage
- Function call coverage
- Branch coverage
- Modified condition/Decision coverage (MC/DC)

The illustration below visualizes how Atollic TrueANALYZER detects all different types of coverage scenarios described above.

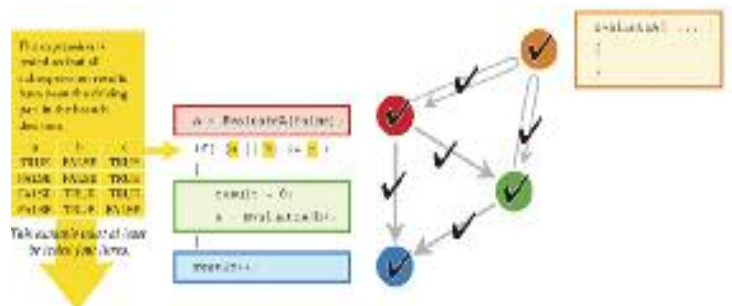


Figure 9: Atollic TrueANALYZER measure test quality rigorously.

Atollic TrueANALYZER connects to the target board via the same JTAG probe used by the C/C++ debugger. The embedded application is run directly on the target board with the code coverage information recorded for later inspection and quantitative analysis.

*Continued on page 42*

# Unlock the potential of your ARM-Powered® designs.

Join the community defining the future.

Chip Design Day  
**October 25**

Software & Systems Design Days  
**October 26-27**

The Santa Clara Convention Center,  
Santa Clara, CA

## Subscriber Discount

Receive a 25% Savings on Conference  
Registration Use Promo Code **IQSUBS**.

**FREE ViewSonic® gTablet** when  
you purchase a conference  
all access pass



**ARMtechcon.com**

# JOIN US IN DEFINING THE FUTURE!

FREE ViewSonic® gTablet when you purchase a conference  
All Access Pass

## HIGHLIGHTS

### EXHIBIT HOURS

OCTOBER 25, 10:15 AM-7:00 PM

OCTOBER 26, 10:30 AM-6:30 PM

OCTOBER 27, 10:30 AM-4:00 PM

### KEYNOTES

#### Advanced Process Technology Roadmap

**Dr. Shang-Yi Chiang**, Senior Vice President,  
Research & Development, TSMC

#### Measuring and Enhancing Product Differentiation

**Walden C. Rhines**, Chairman and Chief  
Executive Officer, Mentor Graphics

#### 2020 in 26 Easy Steps

**Mike Muller**,

Chief Technology Officer, ARM

### INDUSTRY ADDRESS

**Chi-ping Hsu**, Senior Vice President, Research  
and Development, Silicon Realization Group,  
Cadence Design Systems, Inc.

### FIRESIDE CHAT

Hosted by **Lip-Bu Tan**, President and Chief  
Executive Officer, Cadence Design Systems,  
Inc. and **Simon Segars**, EVP and General  
Manager, Physical IP Division, ARM

### ARM Theatre

Featuring Android Speed Training, Teardown  
& Giveaway: ViewSonic® gTablet, Teardown:  
Exploring the technology behind "Intelligent  
Play" and more!

### FREE EDUCATIONAL SESSIONS

Sponsored sessions by ARM, Cadence, ENEA,  
Freescale, Magma, Marvell, Mentor Graphics,  
NXP, STMicroelectronics, Synopsys, TI, TSMC,  
UBUNTU and more.

### ARM TECHCON CONFERENCE DATES

#### Chip Design Day

Tuesday, October 25

#### Systems and Software Design Days

Wednesday & Thursday, October 26 & 27

### HANDS-ON SESSIONS & TUTORIALS

In-depth hands-on solutions to your specific  
design challenges

### FUN & GIVEAWAYS

ARM Tool Kit Giveaways, The NEW ARM  
Techcon SOLVE-IT Puzzle Challenge

Chill Out Lounge and Evening  
Receptions!

### REGISTER NOW

Use promo code IQSUBS to Register for your  
FREE expo pass and to save 25% off  
any conference package.

*Discount cannot be used in  
combination with other  
discount offers.*



Join the community defining the future



[www.ARMtechcon.com](http://www.ARMtechcon.com)

IN THE EYE OF  
EVERY ENGINEER...



...THERE IS A NEW EMBEDDED SOFTWARE SOLUTION.™

One that is integrated and optimized – simplifying what was once a complex problem.

Don't miss the unveiling at ARM TechCon™ 2011

Register today – [www.armtechcon.com](http://www.armtechcon.com)

Brought to you by

