A case for Simulation Software

Overview

Embedded Safety critical airborne applications have to be tested thoroughly for the functionality and coverage per industry guidelines based on the criticality level given to the application as per DO-178B guidelines. During verification phase, Hardware Software Integration Testing is one important phase. This paper provides a solution where high level testing can be done with the use of simulation software.

Hence forth the term "Application" used in this paper means it is Embedded Safety critical airborne application.

External simulation using Automated Test Equipment (ATE)

To test high level requirements, we typically use a Test equipment to drive external inputs. For instance it is common that we use the following type of signals during High Level Requirement testing.

For Discrete IOs, we normally use Relays or manual switches with LED(Light Emitting Diode) indicators.

For speed inputs, we use Frequency generator boards.

For sensor inputs like RTDs (Resistance Temperature Detectors)s we use voltage inputs.

To measure any Analog output signals we use a high profile Oscilloscope, DMM, or Frequency analyzer etc.

Any manual measurement is laborious and prone to errors. So we build Automated Test Equipment (ATE) which hosts specific Analog, Frequency, and Discrete boards and Software to drive these boards.

A typical ATE consists of a good number of Analog and Discrete boards including ARINC and CAN modules and a host of Software layers like Matlab, Simulink or LabVIEW/LabWindows or any other Script based tools like Python etc.

We see that maintaining the Software versions or Hardware revisions, the high cost of procurement and incompatibility between the Hardware and Software often cause delays and change in design while developing ATEs.

Also we come across many limitations while testing high level requirements. Fault insertion tests, verification of displayed output on a LCD screen, inspecting intermediate updates to a global variables etc may pose problems while testing. Sometimes design team may also require to measure task execution time, iteration rates accurately independent of the Target Software.

Test Simulation System

Here is a solution. We can build a test simulation system. A simple test simulation System can be part of an actual Unit under Test and can simulate inputs and send outputs periodically to the outside world (peek and poke functionalities).

A basic Test Simulation System consists of the following:

- Desktop with test custom made Script Software
- Interface cable between Desktop and Unit Under Test (UUT)/Target
- Target System
- Input Simulation Module
- Output transmission Module

To understand the design, let us understand our process of testing. First and foremost we write test cases against the requirements. A typical test case contains the input to drive and the output expected for a given requirement. Then there is a script that reads the test cases and drives inputs to the Unit Under test. The script collects the outputs send by the Output transmission module residing on UUT, compares them with expected outputs and declares results. Finally Script creates a Test result file as an output that contains the log of test execution and pass/fail status for each test case.

Designing Test Simulation Software

The Test Script Software on the Desktop can be used to read the test cases that are written. However the script running on Desktop has to communicate to the target and provide required inputs to the Application that is being tested. For this a Hardware interface is required between the target and the desktop.

Usually,targets provide a simple RS232/422 interface. This can be used as an interface between the UUT and the desktop (Figure 1).

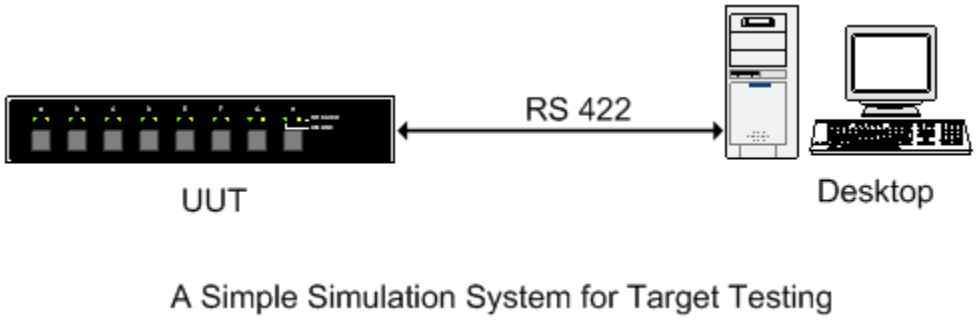

Figure 1: Simulation System setup

Now, how do we feed the inputs to the Application software running on the target (UUT). There are many ways to do this. This white paper provides one of the methods.

An embedded Application (UUT) is usually divided into three Software programs,

- Boot
- Operational Flight Program (OFP) and
- Hardware Abstraction Layer (HAL) or Device Drivers.

Usually, HAL, after reading the data from Hardware (like ADC, ARINC, Discrete IO, FPGA, CPLD etc), place the data in a shared area in the raw format so that OFP picks it up periodically. The HAL provides the data to the application as and when it reads them from Hardware. The shared area can be usually a big chunk of RAM which can be accessed sequentially (Figure 2).

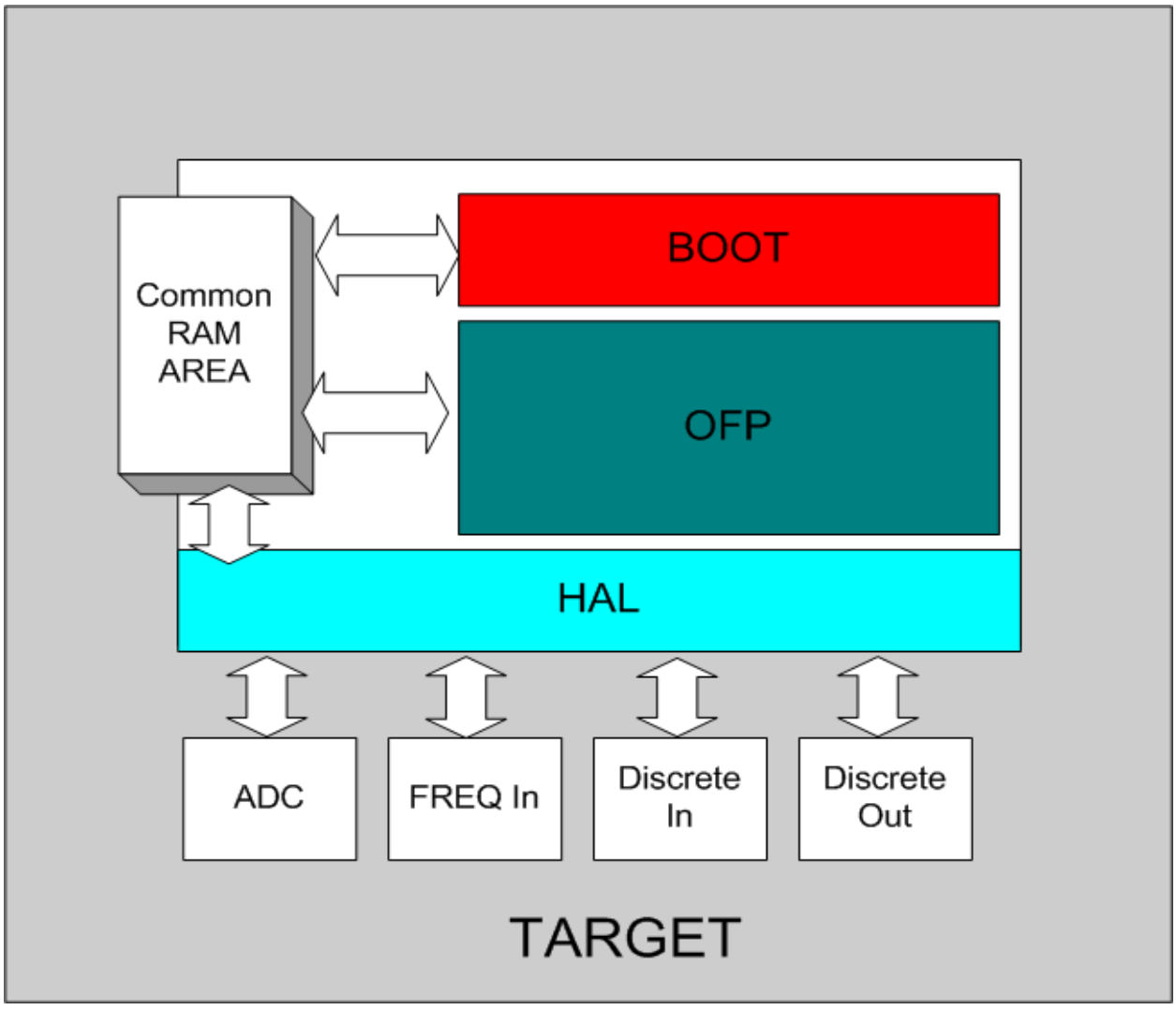

Figure 2: Block Diagram of Application Layer architecture

This kind of design enhances re-usability as, when we change the target board, we do not have to re-write our application but replace only the HAL for the new target.

Our aim is to use this facility of shared RAM, so that in place of HAL, the Simulation Software can place received test inputs into the shared RAM location and also read Outputs of the OFP from this RAM location (Figure 3).

Let us get into little more detail on our Simulation System now.

The Simulation System requires minimum following Hardware, Software and interface.

- A Suitable Desktop with at least RS232 or RS422 ports

The software components can be as follows:

- Test Script Software on the Desktop
- Test Simulation Software on the target
- Test Case Database
- RS 232/422 Protocol
- Target Board and Target Application (OFP) for testing

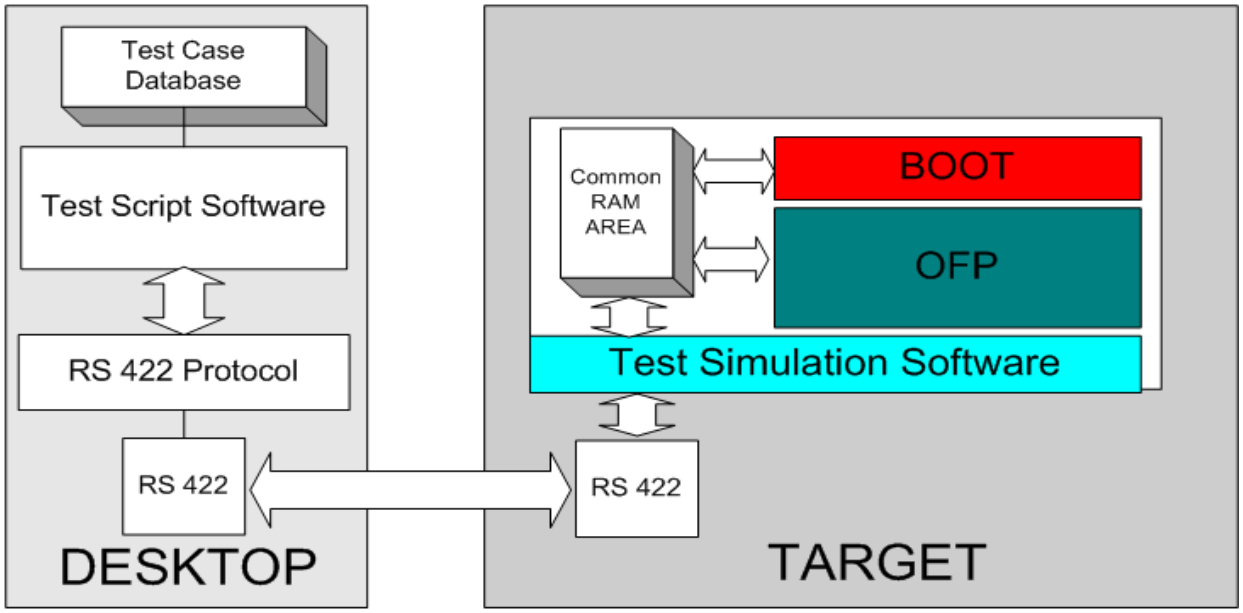

Figure 3 HW and SW Components of a Simulation System

Test Script Software

The Test Script Software is a GUI application software that runs on a PC or a desktop. These applications can be developed either in C, LabVIEW or Python or MatLab.

The primary task of the Test Script Software is to read the test cases from the Test Case Database.

Once the test cases records are read, It has to prepare packets for RS232/422 protocol to dispatch them to the Test Simulation Software.

The Test Script Software can also collect the Target Outputs from the Test Simulation Software via serial interface and compare the expected output with actual Outputs and prepare a Test Result file.

It is also advisable to have a log file generated for all the events that happen between the Desktop and Test Simulation Software.

Once the basic Test Script Software is ready and working, many more features can be added. Building intelligence into the software to pick up all scripts that are available for a particular build and storing the related test results in a specific area during the execution. A simple module to browse all test result files that shows up all the failed test cases. This will help to perform analysis quickly to ascertain failure ratio. Automate the process so that regression runs and run for score can be performed with least manual intervention.

Another automation area can be, a provision to switch on and switch off the target using script commands which can drive electronic relays.

Test Case Database

A simple Test case database can be done using Xlsheets or plain ASCII text files. For more automation one can go in for SQL servers. A typical Test case record will have following fields.

Test Case Number: Unique Number to identify test cases.

Test Input/Output Name : A column that holds the signal names

Data type : Determines the data type of the Test input/output field. An id to indicate Integer, Char, Float, signed/unsigned etc., can be stored in this field.

Test Input/Output Value: based on the data type this field can contain a value to drive or value to expect.

Data Attribute: indicates if this is an input signal or output signal.

But there are many ways to represent this info in an xlsheet.

RS 232/422 Protocol

A robust protocol is required to send and receive messages between the Desktop and the target. The messages will carry commands and data.

The commands can be “Receive Inputs”, “Send Output”, “Resend” etc.

The data packets contain actual test inputs to Target or output from targets.

Test Script Software controls the Protocol and initiates communication and declare results based on the received data.

It is usually the Protocol Software in the desktop considered as a Master and the Target side protocol software is considered as slave.

Test Simulation Software On the Target

Actual Test simulation Software on the Target has two software components. One is the RS232/422 communication protocol or the slave protocol and other one is the background process that keeps updating the shared RAM based on the input it received from the Test script Software and also processing of other commands.

The Test simulation Software sends required output from the shared RAM Area based on the request from the Desktop (Test Script Software)

One more important function of this software is to convert the raw data into engineering data when it has to send the output back to the Desktop.

However this can be the function of Test Script Software as well.

How do we simulate ARINC 429 or CAN bus data.

The best practice is to use ARINC analyzers with actual ARINC 429 modules for digital buses (and so is the case for CAN bus). The reason being, these software modules are tried and tested and generally are available in the company labs.

Simulation of ARINC 429 or CAN bus is possible using RS422, provided once is ready to invest some time and learn the working of these digital buses.

Advantages and Disadvantages of Test Simulation System

It all depends on what kind of embedded application we are testing. We have seen the cost benefits when these are designed well in time and with almost all functionalities built into it. The biggest advantage is the automation of manual testing and also performing regression runs with least manpower support.

All these systems require qualification. This may take time, but for that matter any tool used in safety critical applications required to be qualified.

Timing requirements may not be tested accurately. May be we should isolate such requirements and have a separate test set-up for such things.