NEXT GENERATION TEST LIBRARY FOR RF SOC ON ATE

Ping Wang[1], Haocheng Yuan[1], Vincent Lin[1], Haixia Guo[1], Goh Frank[1]*
[1]Global Application and Development Center of
ADVANTEST Shanghai 201203, China
*Corresponding Author's Email: ping.wang@advantest.com

ABSTRACT

Current RF SOC devices are highly complex with many modules with different functions and wireless standards. This complexity results in thousands of tests needed even for production testing. Characterization and engineering test flow will demand even more tests.

It takes a lot of time to develop, debug and bring the program into production. This is especially true for new engineers which can take months or years to develop on their own. Typical solution would be to develop a test library to generate all the tests. However, not all test library are build the same. And typical "1st generation" are done to get the job done. They can be difficult to debug and may need hack codes for special or customized test. In addition, some libraries have grown so complex that it takes a long time to learn how to use them.

With experiences gained from development of many projects and Java coding, a new better next generation test library was developed. This paper described a new improved version of such a test library which provides more features & benefits, more user friendly and easy to learn.

The paper first describes the need for test library and typical issues with existing test libraries. The benefits and new features of the next generation test library will be introduced.

The structure and use model of this user-friendly RF SOC test library will be described. How and why development time and cost will be reduced. Complicated hacking codes can be eliminated. Future-proofing the library, it is designed to allow new wireless standards and tests to be added. Debugging and setup features will also be described.

Efficient and ease of importing user input, registers setup and design pattern will help even new users create test program efficiently and quickly.

This library is also designed to ease learning curve when using it. A beginner can develop a new kind of RF SOC offline program within one month. Framework to allow easy usage allows users to stay at a higher level helps them to focus on test strategies.

Keywords—RF SOC; Library; ATE; efficiency;

INTRODUCTION

Development and Challenge

With fast development of broadband wireless technique in the past years, it stressed huge pressure for requirements of radio frequency (RF) devices with higher performance and lower power consumption. There will be more IPs involved in RF Integrated Circuit (RFIC). Traditional mixture with digital and analog circuits could not meet market needs. More complex RF SOC devices with different functions and wireless standards will flow into market place.

At present, current RF SOC products are used for multiple domain, such as 5G/LTE wireless application, millimeter-wave radar and smart Internet of Things (IOT) application. All of them need test new features and cover more design concept from device.

Then in order to production smoothly, test engineer need develop complex automatic test program which contains tens of thousands of tests. In addition, characterization flow will be also involved. If user still use obsolete way to create method one by one, it will take months or years to finish the whole program. Surely more and more test libraries enter into test engineer's view. But some of library is hard to debug, some of them shows low execution efficiency and some of them is difficult to be understood by primary engineers. Building one powerful RF SOC library will be most important thing in IC automatic test field.

TEST LIBRARY

Introduction

Traditional test program developed for each RF SOC device contains a mass of complex main test methods which is hard to understand and debug for user. This paper will bring new concept that user could break this obsolete mode and create new frame combined with classic design pattern, such as strategy pattern, observer pattern and template mode.

In current smart design pattern, we borrow advanced concept and rule from java structure applied on internet. In order to make user configure device setup, such as register setting, we create many scalable interface to meet different application include Digital RF version and traditional IQ based version. User could build device setup efficiently and change setting quickly. Then as for tester setup, we built one flexible library which contains all instrument setup, like RF instrument setup, analog instrument setup...etc. User just call relevant setup function to trigger tester instrument. In addition, test input parameter will be described in the tabular sheet, all test parameters are controlled by open instance, so it is convenient for primary engineer to maintain or develop.

2021 China Semiconductor Technology International Conference (CSTIC) | 978-1-6654-4945-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSTIC52283.2021.9461580

If you use SmarTest 8 structure on V93000, only one command could help execute all setups in main test method. With command named Release Tester, that means java virtual machine will release relevant threads and turn to next measurement, hardware execution threads will follow defined sequence, but calculation threads could be executed in parallel. In the Figure 1, you could see multiple threads behind virtual machine. This feature could be used for improving execution efficiency and keep correct execution sequence.

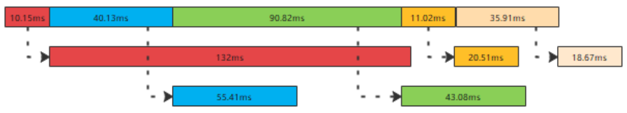


Figure 1: Mutli-Thread Pool

In data process, this library build two types of packages, which are continuous wave measurement and modulation wave measurement used for transmitter and receiver (TRX), Analog-Digital and Digital-Analog (ADDA) test items. Then user could import different class and load correct function to finish data process, such as power, evm, spectrum mask, analog signal analysis …etc.

So user could find correct standard method to finish relevant tests or create other specific tests on base of library. They only focus device setup process and fill in input parameters with tabular format among similar products. Even for one primary engineer with less test experience, it could be available to develop, debug and maintain test program independently. As for execution efficiency, there is no need to spend much time on test time reduction (TTR) because of optimized design structure.

With the library, users stand at the highest level of test method, only follow the structure of template method and combine setups and calculation selectively according to the test items. There is no need to focus on code details.

Test Frame

Figure 2, Figure3 and Figure4 show the sample frame of test library with sequential view. Clear task-based code and flexible user-defined interface or function could meet quick development of complex RF SOC test program. Meanwhile, smart design pattern and software structure provide better performance than tedious single main test method.

In setup block as shown in Figure2, user just focus on device setup, then could call different tester setup from library directly. All test sequence setup instances which are common boilerplate code are moved up the inheritance hierarchy. So it is convenient for user to view and debug because of single entry code. As for input parameters, it solved issue of complex parameters management and support smart burst mode which is a combination of test items in one main test method. It could save test time and improve test efficiency. Meanwhile, we changed tabular format in order to make it fill in parameters quickly, which is one specific feature for RF SOC test program.

In the update block as shown in Figure 3, user could change their primary setup, such as update loss. Before that, there is no extra test time consumption. Then flexible and powerful interface could be used for data process after executing previous setup. Surely user could also defined specific function by themselves on base of scalable structure.

In the Figure4, it created different child class to describe algorithmic details. There is one important feature which is multi-thread distribution shown in Figure 1. In addition, we use Log4j frame to manage program debug log which is popular in java field.

In order to get better usage, this library added easy and practical script and user-interface frame. As for instance, user could generate scientific test limitation automatically, achieve data monitor in production and loss calibration with UI.

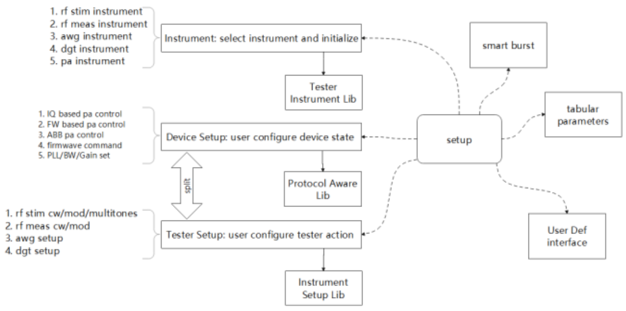


Figure 2: Test Frame of RF SOC Lib – setup

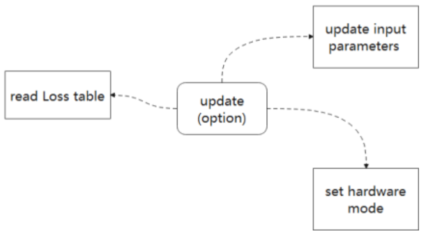


Figure 3: Test Frame of RF SOC Lib – update

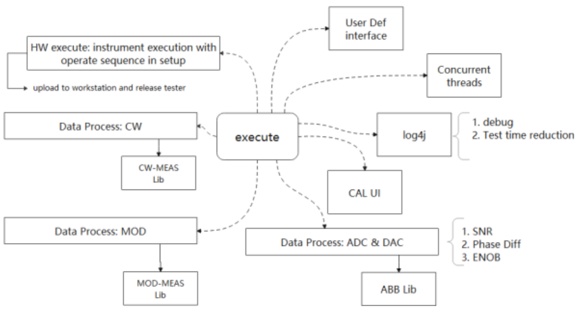


Figure 4: Test Frame of RF SOC Lib – execute

Thus far, we got so many hits from engineers about library application. They could get rid of coding pain and devote to study test strategies. And in Figure5, we summarized top six outstanding features from engineers and make comparison with traditional library. If you use this lib to develop test program, you could get these benefits, decrease development period and developer resource, decrease test time and get low test cost, better for debug and support kinds of product.

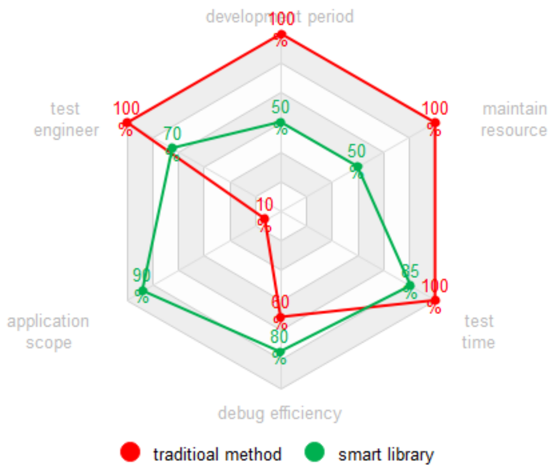


Figure 5: Outstanding Features

Application

In order to have clear understanding, this paper will show a real case with this library. As we can see in Figure 6, it shows RF TRX parts in RF SOC products. Band configuration, Phase Locked Loop (PLL) configuration and gain setting are most important for tens of thousands of TRX tests. We choose one simple RX Gain test as example.

From test strategy for gain, RF stimulus instrument transmit input power and continuous RF frequency to RF port in device. Then test engineer need configure device state with protocol interface. After finishing these setups, hardware is executed and upload waveform data to workstation. Finally test engineer need think about gain algorithm to get correct test value.

In the Figure 7, it shows tabular format which is used to set different parameters. User could use any of these two way which is shown in highlight two parts to configure parameters. For the yellow row, it is one smart way for multiple bands and multiple channels, user only configure one line to generate many tests. Surely user could also use another way directly. So as for RX Gain, it defines RF frequency, RF input power, band width, which belongs to device setup. Then for tester setup, it defines ADC sample rate and sample size, surely it supports more parameters if you configure more columns.

In Figure8, user could call different function or instance from defined library, such as tester instrument setup, RX gain setup instance and gain calculation. As an instance in Figure 9, you could define or call tester setup or device setup function in RX gain setup instance. It is clear for reader show how to execute sequence defined in setup block. Then for parameters shown in spread sheet, all of them stored in TestParameterLink instance shown in Figure 9.

After finishing test method development, user could define test suite, test flow and test program with SmarTest 8 from Advantest. Then it turn to be available to execute all test methods with automatic test frame.

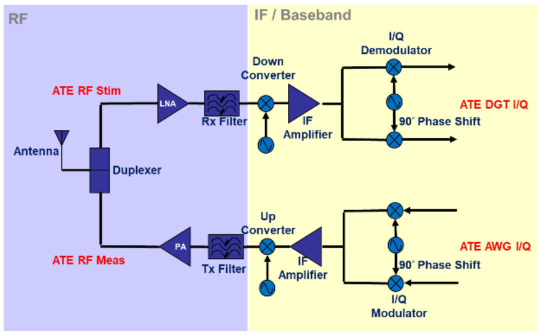
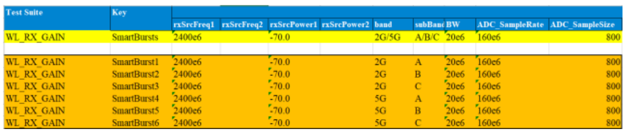
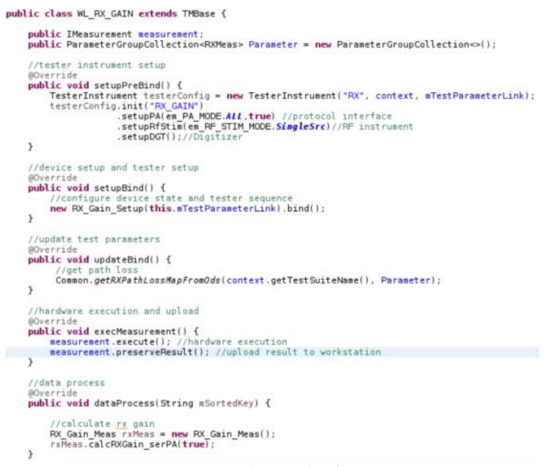


Figure 6: simple module for TRX in RF SOC



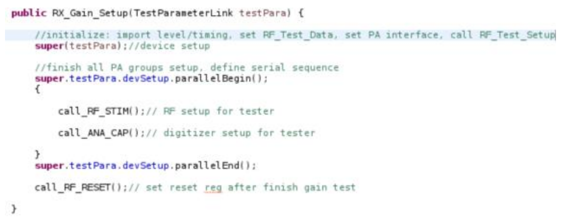
Test Suite	Key	rxSrcFreq1	rxSrcFreq2	rxSrcPower1	rxSrcPower2	band	subBand	BW	ADC_SampleRate	ADC_SampleSize
WL_RX_GAIN	SmartBursts	2400e6		-70.0		2G/5G	A/B/C	20e6	160e6	800
WL_RX_GAIN	SmartBurst1	2400e6		-70.0		2G	A	20e6	160e6	800
WL_RX_GAIN	SmartBurst2	2400e6		-70.0		2G	B	20e6	160e6	800
WL_RX_GAIN	SmartBurst3	2400e6		-70.0		2G	C	20e6	160e6	800
WL_RX_GAIN	SmartBurst4	2400e6		-70.0		5G	A	20e6	160e6	800
WL_RX_GAIN	SmartBurst5	2400e6		-70.0		5G	B	20e6	160e6	800
WL_RX_GAIN	SmartBurst6	2400e6		-70.0		5G	C	20e6	160e6	800

Figure 7: smart burst tabular format



```
public class WL_RX_GAIN extends TMBase {

    public IMeasurement measurement;
    public ParameterGroupCollection<RXMeas> Parameter = new ParameterGroupCollection<>();

    //tester instrument setup
    @Override
    public void setupPreBind() {
        TesterInstrument testerConfig = new TesterInstrument("RX", context, mTestParameterLink);
        testerConfig.init("RX_GAIN")
                    .setupPA(em_PA_MODE.ALL,true) //protocol interface
                    .setupRfStim(em_RF_STIM_MODE.SingleSrc)//RF instrument
                    .setupDGT();//Digitizer
    }

    //device setup and tester setup
    @Override
    public void setupBind() {
        //configure device state and tester sequence
        new RX_Gain_Setup(this.mTestParameterLink).bind();
    }

    //update test parameters
    @Override
    public void updateBind() {
         //get path loss
         Common.getRXPathLossMapFromOds(context.getTestSuiteName(), Parameter);
    }

    //hardware execution and upload
    @Override
    public void execMeasurement() {
        measurement.execute(); //hardware execution
        measurement.preserveResult(); //upload result to workstation
    }

    //data process
    @Override
    public void dataProcess(String mSortedKey) {

        //calculate rx gain
        RX_Gain_Meas rxMeas = new RX_Gain_Meas();
        rxMeas.calcRXGain_serPA(true);
    }
```

Figure 8: simple code for RX Gain



```
public RX_Gain_Setup(TestParameterLink testPara) {

    //initialize: import level/timing, set RF_Test_Data, set PA interface, call RF_Test_Setup
    super(testPara);//device setup

    //finish all PA groups setup, define serial sequence
    super.testPara.devSetup.parallelBegin();
    {

        call_RF_STIM();// RF setup for tester

        call_ANA_CAP();// digitizer setup for tester

    }
    super.testPara.devSetup.parallelEnd();

    call_RF_RESET();// set reset reg after finish gain test

}
```

Figure 9: simple code for RX Gain setup

Until the end of 2020, this library has been widely applied on many RF-SOC projects, such as Wi-Fi 6E products and base station transceivers. We got good feedback from different customers of all over the world. And benefit from the flexible framework, this library will be optimized constantly as per the cutting-edge technology.

CONCLUSION

The paper showed the architecture of a flexible library which could be used in any RF SOC device test program. This library will save most of development time and release engineers from coding debug. Flexibility is the most significant characteristic of this library. On the one hand, it allows users to stay at the highest level and develop program quickly and efficiently. On the other hand, it eases learning curve, so that test engineers will get more benefits if they understand library framework. Obviously, increasing new parameters, creating new setups, adding new RF feature calculations are easier for deep-learning engineers when they are developing next generation product program.

ACKNOWLEDGEMENTS

The author wishes to thanks Advantest for the support on the project.

REFERENCES

[1] IEEE 802, “IEEE P802.11ac/D1.1”, Institute of Electronic Engineers, Aug 18, 2011.
[2] V93000 SOC WSMX Analog Card User Manual, Advantest.
[3] V93000 SOC WSRF RF Card User Manual, Advantest.