Using Protocol Aware Tools to Simplify Program Development for RF SOC Test

By Ronald Burke
Teradyne Inc.

Developing an ATE production test solution for today's complex RFSIP/SOC devices has been an increasingly difficult task

Developing an ATE production test solution for today's complex RFSIP/SOC devices has been an increasingly difficult task. Most of the typical RFSIP/SOC devices are multi-band and in most cases multi-function. Test requirements have moved away from simple RF parametric tests (like: Gain, Noise Figure, S-Parameters) and have moved toward system level testing including modulation. With new enhanced device capability and system level test requirements, today's typical RF ATE test program is substantially more challenging and is more time consuming to develop. Test engineers are always under great pressure to deliver a solution on time with overall Cost-of-Test (COT) that is better than the previous device.

So what is the major challenge here? The communication protocols are! As previously mentioned we need to know how to code test solutions for extremely complex RF modulation protocols for both transmitter and receiver testing. But it doesn't stop there, since most of these devices use some sort of digital interface like SPI, MDIO, I²C, PCIe or JTAG, device setup and communication requires an understanding of digital protocols:

Looking at two examples:

• 4G cellular transceiver: Today's typical transceiver supports both yesterday's and today's RF Protocols (including the modulation schemes: GSM, EDGE, WCDMA, HSPA, LTE) and will have anywhere from 6 to 16 RF ports. Digital interfacing may include SPI, DigRF or other.

• Connectivity device: Usually includes support for BlueTooth along with 802.11 A/B/G/N (may also include MIMO support) with a minimum of 5 RF ports. Digital interfacing done via I2C, JTAG, PCIe, SPI, USB.

For the typical connectivity device what are the interfaces/protocols necessary for testing:

• Device setup is typically accomplished using an interface like SPI or JTAG.

• Samples captured by the device's receiver are transferred to the ATE using an interface like I²C, PCIe or JTAG. The samples need to be extracted from the digital transfer protocol and reassembled for further measurement analysis.

• Connectivity RF protocols may include BlueTooth and 802.11 A/B/G/N

• RF source signals compliant with the above standards are required to test the device receiver.

• The ATE system must be able to capture and process the RF protocol compliant signals from the device's transmitter.

Test Program Development

The most difficult programming tasks center on the development of the protocols needed for the ATE system to communicate with the device for both the digital and RF sections of the device. So why is this difficult? Well, test engineers must often hand craft vector patterns for each new device and for each new test. These vector patterns must include the digital register read/write setups (encoded to conform to the protocol) along

with the source and capture information for analog instruments. Further more, the typical test program for an RFSIP/SOC is usually DSP intensive for both transmit and receive and will require creation of complex modulated waveforms and demodulation DSP algorithms required to test the DUT.

Digital Protocol

The large majority of today's SIP/SOC devices use a digital bus (I^2C, JTAG or similar) for both programming and data signal (both register data and capture data) transfer. What are the challenges to testing a device with one of these interfaces? It's the amount of time it takes to develop the digital side of the test program to conform to the protocol. Transmitting to the device is manageable, but time consuming and very cumbersome, because in order to communicate via the digital interface each register change and or data bit transmitted must be converted to conform to the protocol and then coded into a vector pattern.

Thankfully due to the fact that the tester is driving the device input clock and digital information is sent via pattern, both the tester and the device are aligned in frequency and phase, so the device can easily receive the incoming data. The larger problem is measurement or retrieval of captured data from the device. It's not as simple as transmit, because the phase timing of the device's digital output edge may be shifted and or the location of data words maybe asynchronous. So it becomes common practice to implement a matched loop in order to search for the timing of the output edges and then capture the transmitted data to be processed later via DSP in order to extract the retrieved register values or captured signal data. To retrieve the data words from the encoded protocol, DSP code must be written to locate and then decode the captured data words from the measured signal.

Design engineers are familiar with and work directly with register settings. When the device moves from design into test, most test engineers are given an extensive set of hexadecimal based register tables that have to be used in order to program and test the device. Each new test has the distinct possibility of requiring a new series of register settings. For speed it is best practice to code the register settings into vector patterns for the digital instrument. To do this each register setting is converted from hexadecimal to binary, wrapped in the protocol and coded into vectors. There will be patterns for device mode setup, for device transmit (possibly requiring a pattern for the mode change and another for the transmit data stream) and patterns required to retrieve captured data from the device's receiver. The number of patterns can add up very quickly into hundreds of patterns in a single program. Debug becomes a difficult and time consuming task to locate and change register values that have been encoded into long vector patterns, especially when your primary help (the design engineer) is not familiar with vector patterns and has requested multiple register changes/updates. In the end, there is great risk during debug and silicon revision updates that many of these patterns will require modification in order to achieve and keep proper correlation. Program development, debug and maintenance become very complex and time consuming, one update could take several person-weeks of effort.

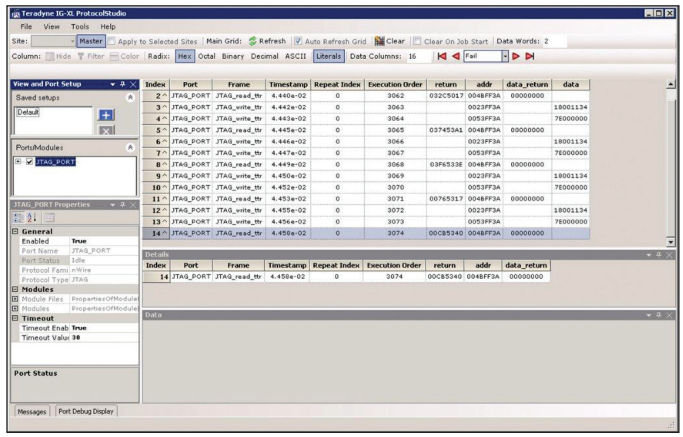
There has to be a better way and there is: digital Protocol Aware (PA) hardware. An ATE system with local digital PA engines eliminates the above problems and more. Data words are encoded and decoded compliant to the desired protocol locally in the PA engine; this removes the need to develop large complex patterns and DSP algorithms. Also a hardware based protocol engine will detect and align to the device's timing and manage the measurement acquisition, returning the desired data words. Therefore instead of writing hundreds of patterns, test engineers should be able to write simple calls to functions like:

JTAG(Write, RegisterAddress, RegisterData)

Or

Data=JTAG(Read, RegisterAddress).

These simplified digital calls are then easier to code, debug and maintain over the course of the production run. In fact this simplifies ATE digital protocol programming to be similar to bench equipment, by directly using hexadecimal values to set register addresses and data. Now, without the need to develop all of the encoding/decoding for the digital protocols, the test engineer is free to focus his programming efforts where he should, on device setup and measurement. PA's greatly simplified programming format also streamlines the communica-



Digital Protocol Aware Debug Tool

tion with design. As stated before, design engineers are typically not familiar with ATE patterns, so there has been a gap in knowledge and terminology between the design engineer and the test engineer. PA bridges that communication gap, greatly shortening the debug cycle. Therefore program development and debug time should be reduced with the end result: faster time to market!

RF Protocol

Until a few years ago it was possible to test the typical transceiver using simple sine waves. SIP/SOC device integration and the rapid evolution of RF communication protocols, with increased bandwidths and more complex modulation requirements, are driving test lists that include system level testing like Error Vector Magnitude (EVM), Bit Error Rate (BER), Packet Error Rate (PER), etc. Development of the RF protocol solution for a single SIP/SOC, including both modulation and demodulation, typically requires several months' worth of effort to read through the device standard in order to decipher the protocol's requirements before coding can even begin. The effort required to complete this task could take an additional five to nine months to develop a robust and reusable algorithm that can cover the permutations of the standard necessary to test the device.

For example, a modulated source signal is created by building a mathematical I & Q segment, which is frequency up converted and sourced to a device. To be compliant with the protocol, code development for each source signal may involve coding the following: Header coding function, Data coding (Convolution encoder or other), Interleaving function and Mapping function.

Today's bench synthesizers have reduced this task to just a point and click operation. ATE RF Protocol source solutions should be the same way. The desired ATE solution will contain a tool that allows test engineers to simply choose a source signal from a set of predefined waveforms for any of today's complex communication protocols. This will reduce code development of the modulated waveform to just minutes.

The demodulation algorithm will require building DSP functions to perform the following: Bits comparator function, Demapping function, Deinterleaving function, Data decoder (Viterbi decoder or other), Header decoder function and Mapping function.

Once the signal has been demodulated and the data bits have been recovered, they must be processed further in order to get to a result. Code development continues with the creation of measurement functions to calculate: EVM, Phase Error, BER, time mask, spectral mask, Adjacent Channel Leakage Power Ratio (ACPR), and some simple spectrum analyzer-like functionality. Again this activity may require many person-months of code development.

Vector Signal Analyzer Debug/Measurement Tool

Bench spectrum analyzers have reduced this task to just a point and click operation. You select the appropriate RF protocol and turn on the spectrum analyzer's vector signal analyzer (VSA) in order to demodulate the incoming signal and then measure EVM, etc.. ATE RF Protocol measurement solutions should include a VSA solution that allows test engineers to quickly setup a measurement with just a few simple clicks. This will reduce DSP code development for complex communication protocols to minutes.

Conclusion

With the continued advances in RF SIP/SOC integration, test program development has continued to become more complex. Today's ATE system must include hardware and software tools to allow fast program development for complex protocols that can stimulate, measure and interpret captured signals for both the digital and RF sections of the device. With the right tools the typical time to market for test programs that contain system level tests can be reduced by many person-months.

Author Information

Ronald Burke is employed by Teradyne Inc. as Senior Factory Applications Engineer, specializing in Wireless/RF. He has more than 13 years of RF/microwave ATE experience. In his 11-plus years at Teradyne Ron has worked in the following positions: Factory Applications Engineer, Factory Applications Manager and Field Product Specialist. He has worked on Teradyne's A575, Catalyst, FLEX and UltraFLEX testers, as well as various wireless devices from GSM through LTE. He graduated from the University of Massachusetts - Lowell with a B.S.E.E.