SSC19-VIII-03

Utilization of The Software Communication Architecture for Software Defined Radio Waveform Development

Noah Acosta University of Hawaii at Manoa
nmacosta@hawaii.edu

Faculty Advisor: Miguel Nunes
University of Hawaii at Manoa

ABSTRACT

The Software Communication Architecture (SCA) is a method developed for the U.S military's Joint Tactical Radio Systems that aims to standardized the way in which software defined radios for the U.S armed forces are to be configured and built. Most developmental software defined radios use proprietary software such as GNURadio Companion (GRC) or Gqrx SDR to construct waveforms for RF applications. Here we utilize a suite of tools called eCo Architect and eCo Inspector developed by NordiaSoft, which conforms the end-to-end waveform to satisfy the SCA requirements. In this paper, we present an evaluation on waveform development using the SCA in a lab environment of integrating two waveforms using NordiaSoft's eCo Suite in conjunction with an Ettus E310 Software Defined Radio (SDR). A comparison in data packet recovery between GRC and eCo Suite was also performed. The same waveforms were used in both GRC and eCo Suite with differences in component algorithm construction.

Introduction

Since the development of configurable receivers in the 1980's, waveform development for radio communications has become significantly more versatile than its traditional counterpart which has limited cross-functionality and requires physical hardware intervention in order to adjust its capabilities for signal and data processing. Radios that are software defined, improve on production cost and flexibility for handling multiple waveform standards. The context of a waveform for our application is related to the lower layers (network, data-link, and physical) of the ISO-OSI model. Efforts to facilitate the flexibility of SDRs and the re-use of waveforms across multiple radio sets lead to the initiative program Joint Tactical Radio Systems (JTRS). Due to the lack of interoperability across the spectrum and insufficient bandwidth that was required for communications, the intentions of JTRS was to provide the warfighter with software-programmable radios that could aid in an advantage in voice and data communications across the battlespace.

The need for an open framework that the Department of Defense (DoD) could utilize without having to seek outside vendor support for software updates and fixes led to the design and implementation of the SCA. The SCA provides its users with a transparency to how hardware and software components must work together. The SCA was originally developed to accommodate software requirements for the DoD because of the limitations on current tactical communication systems. These limitations evolved from traditional systems being designed to meet service and mission specific requirements. An area of programmable systems that benefit greatly from the SCA are SDRs. Although the SCA's intent was to be a standard for programmable and integratable systems, it has not quite been fully adopted by many developers within the DoD. Various groups using SDRs still rely on proprietary softwares such as GNURadio Companion (GRC) for waveform development mainly because of its open source free access and large community of like developers. The waveforms that are described were originally developed within GRC and transitioned into NordiaSoft's eCo Suite. Both GRC waveform flow diagrams were constructed by collaborators at the Naval Post Graduate School (NPS) and the Hawaii Space Flight Laboratory (HSFL) at the University of Hawaii at Manoa. In Section 3A and 3B, we take a look at the end-to-end waveform characteristics and properties for two waveforms PropCube and HiakaSat respectively.

The Software Communication Architecture

Being a component based development architecture, the SCA is independent of the application domains; meaning, it is not strictly confined for SDRs, but it can be applied to other various systems given a set of well-defined API's. This is the reason the SCA has such versatility across tactical systems. Groups that have presented work using the SCA, developed applications within its older version 2.2.2. The work presented in this paper demonstrates end-to-end packet recovery using the SCA 4.1 architecture. Unattractive

aspects of using the SCA 2.2.2 included aspects such as monolithic interfaces. The SCA 4.1 provides a baseline set of capabilities that are structured to enhance security, improving infrastructure performance, maximizing waveform portability and providing a wide range of flexibility and extensibiltiy. The SCA's software components are divided into three function groups; radio management, devices, and applications. These functional groups take part in the framework of the SCA as seen in Figure 1. The middleware that connects these groups with the operating system (OS) is the Common Object Request Broker Architecture (CORBA). CORBA is often referred to as a "software bus" since it acts as a software interface that controls the locating and accessing of information between objects. Application components can communicate with one another no matter where they are located or who designed them. This makes development with CORBA a location and language independent interface.

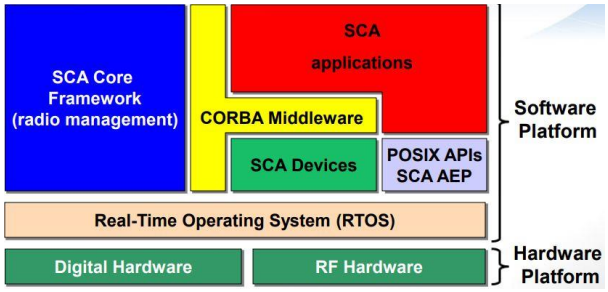


Figure 1: The SCA framework and its integration with hardware devices

The Radio Management (RM) is the core framework that takes care of both manager clients; Domain and Device. it also consists of the ApplicationFactory and Application. RM is responsible for all the actions from installing the applications of interest to configuring and deploying those applications. The second functional group, Devices, provides link access between the executables and hardware components. These devices can be loadable devices or executable devices. Lastly, the third group is applications. To understand the application group, it is important to understand how things graphically look in an SCA radio as shown in figure 2. There will be one Domain Manager. To the left of the domain manager in figure 2, the handling of the hardware is taken care of by the device managers. Each node will have a single device manager and a device manager can handle as many devices as needed. To the right of the domain manager, application factories enable the instantiation of the waveform application. The application group is arguably the most important group of the SCA because the SCA is geared towards making the applications portable between platforms. An application is composed of resources. Here, a resource is a software component that will consist of an input and output port allowing it to connect to other components or rather other resources. Resources can have multiple properties that alter the way in which it behaves. Resources can be modified to interact with other components in various ways, such as developing software to handle complex I/Q or bytes of data. Because a resource provides access to particular API's to a given port of another resource, it is considered to be a port supplier.

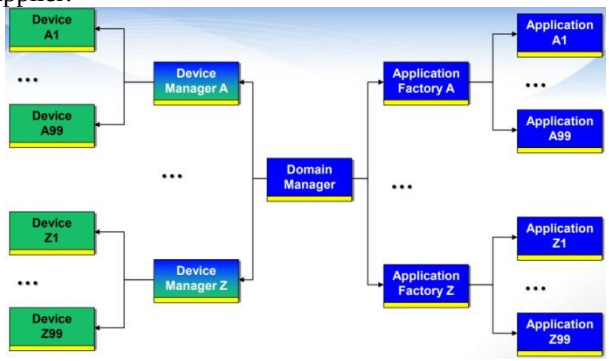


Figure 2: Graphical representation of an application structure with a hardware device

An SCA API can be seen in figure 3. Here, we have a resource that inherits behaviors from other API's. This component based development consists of individual boxes that get connected in assembly. For example, in figure 3, the API associated with operation getPort() will allow a port to be obtained by that given resource. This will enable the possibility for the resource to establish connection with other components. Because a component requires a control, there will be operations that initialize or release an object of a component. This is a result of the specification of the RM to enforce the life cycle of the component. Other operations such as configure() and query() allows a resource to access the internal values of various properties for any SCA component. A given resource must also be able to invoke tests that can be started and stopped during the run of an assembly.

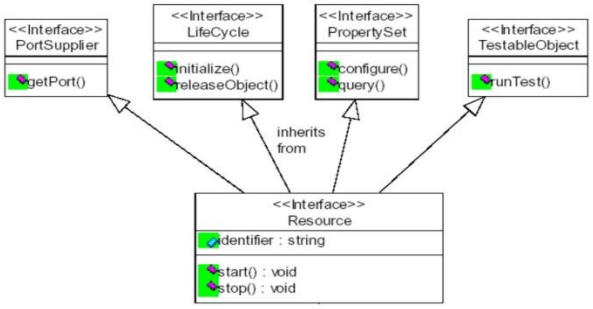


Figure 3: A resource API inherits other API allowing components to be connected together in assembly for component based development

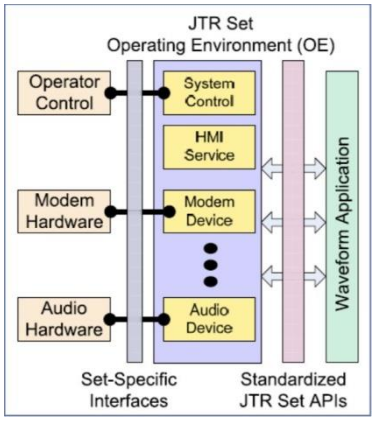


Figure 4: Waveform applications will interact with hardware components

The Device Manager boots the radio nodes and launches the proxies associated with the physical hardware.

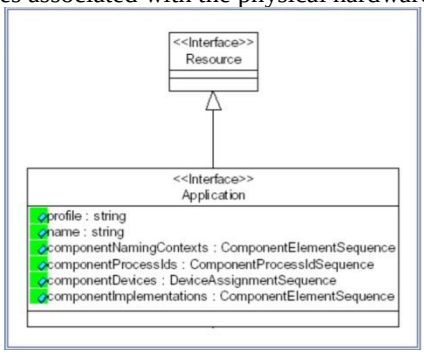


Figure 5: Application API used for storing information about a given application

Using the SCA on Radios

The application group and its resources that are individually constructed with the logic and specifications previously described can be applied to any domain that can be used for component based development. The research described in this paper focuses only on radio development. To put this more into perspective, waveform applications eventually interact with hardware devices. Unfortunately, waveform applications cannot talk directly with hardware devices, so a proxy representation of the hardware devices is needed to map waveforms to the hardware. The SCA defines three types of devices: device, loadable device, and an executable device. A device can be any generic device such as a modem or audio device. Loadable devices act as a proxy to devices such as FPGA's and executable devices generally act as a proxy to a general purpose processor (GPP). Figure 4 illustrates the idea that waveform applications are forbidden to interact with hardware components on their own. For waveform applications to interact with hardware devices, a proxy representation of the three pieces of hardware is needed; devices, loadable devices, and executable devices.

The core framework, or rather the RM group is the supporting infrastructure that allows a developed waveform application to be generated. The RM group consists of four components: Domain Manager, Application Factory, Application, and Device Manager. The Domain Manager, is the heart of the given specification that has defined functionality to enable the radio to operate and applications to be instantiated. The Application Factory invokes the creation of a waveform application. The application box shown in figure 5 is a place holder for information to be stored about the application that is being deployed. A radio consists of a variable amount of nodes. Radio nodes in this context can be considered any device capable of running CORBA enabled code when that device is powered on.

Waveform for PropCube SmallSat

A waveform structure developed by NPS is currently being used to collect data and telemetry from low earth orbit (LEO) satellite constellations called PropCube. PropCube is a series of three CubeSat LEOs used by the Naval Research Laboratory (NRL). The mission of interest for PropCube is for basic research to demonstrate the capability of beacon CubeSats for determining potential effects the ionosphere may have on radio systems, the ability to track and communicate with multiple CubeSats immediately after launch, and providing the value of radio beacons complementing ground based soundings of the ionosphere [2]. The Space Systems Academic Group (SSAG) of NPS delegates efforts to improving upon hardware systems such as SDRs to provide cost-efficient, flexible, and robust software solutions. In collaboration with NPS, we took upon the efforts to transition their existing layout that was designed within GRC and make it SCA compatible. The waveform discussed here utilizes the functionality of a Parsing AX.25 data link protocol decoding scheme.The command and control uplink and downlink frequencies for PropCube are f_{up} = 449.775 MHz and f_{down} = 914.00 MHz respectively. For our test purposes between two SDRs, we ran tests using both frequencies from the transmit and receiving applications. The signal modulation is based off the Gaussian Minimum Shift Keying (GMSK) scheme with a data rate of r = 9600baud and sample rate of f_{SR} = 400 kHz. The raw data that is sourced for the PropCube waveform consists of a pseudo data packet that abides by the KISS protocol. The KISS protocol frames the packet with a leading 0xC0 0x10 and a trailing 0x10. Figure 13 in the appendix displays the cascading of transmit and receiving components used for the PropCube waveform, respectively.

The sent packet is un-KISSED and appended with the appropriate packet protocol taken care of by the HDLC Encoder allowing for the data to be placed on a carrier for modulation, re-sampling, and filtering to the

frequency bandwidth of interest. The receiving signal frequency is up-sampled from 240kHz to 400kHz and passed through a low-pass filter with a cutoff frequency f_c = 5kHz. The signal is demodulated to baseband and uncompressed from 1-bit per byte to 8-bits per byte before passing through the decoding components. The decoding aspect of the waveform takes care of descrambling, differential decoding, and parsing. Scrambling and non-return to zero (NRZ) transformation of the data message was performed by the HDLC Encoder, thus descrambling and differential decoding is needed on the receiving end.

Waveform for HiakaSat CubeSat

The second waveform that is analyzed in this paper is structured very similar to that of PropCube with the exception of a few new components that were developed as shown in figure 6. Certain parameters that were designated for HiakaSat's components also differ from PropCube's. The HiakaSat waveform was developed by HSFL and both transmit and receive waveforms are shown in the appendix in figure 14 respectively. HSFL is a mission developmental group at the University of Hawaii at Manoa which aims to develop, launch and operate small space crafts from the Hawaiian Islands. HiakaSat was a cubesat that HSFL had launched in the past using a Li-1 SDR with its corresponding software developed within GRC. Because the biggest take away from using SDRs is its versatility, one of the milestones set for this paper was to evaluate two waveforms using an Ettus E310 SDR to demonstrate the effectiveness of SDRs in both lab environments as well as during real-time satellite passes.

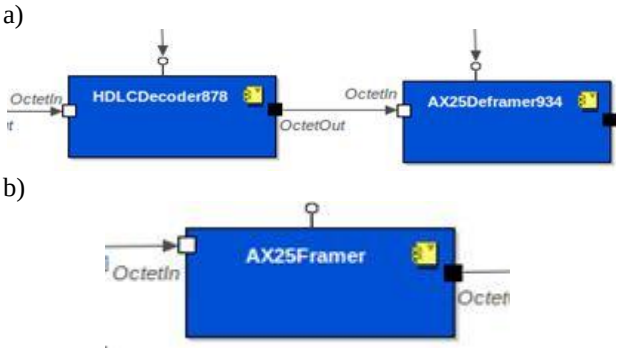


Figure 6: Application API used for storing information about a given application

The uplink and downlink frequencies used for HiakaSat are f_{up} = 440.030 MHz f_{down} = 440.030 MHz respectively. HiakaSat is also based off of a GMSK modulation scheme. The sampling rate was set to f_{SR} = 450.2 MHz. Two components that were added to the HiakaSat waveform that will also be added to PropCube's waveform during testing are the AX25 Framer and Deframer. Licensed operators are assigned an alphabetized regional-based call sign, it is proper practice to append the call sign of the operator who is transmitting as well as the call sign of the receiving operator to the signal. The appended call signs are taken care of by the Framer on the transmitting side of the waveform and the Deframer removes the call signs during the decoding stage on the receiving side of the waveform. The ParseAX.25 protocol is no longer used for this waveform. However, similar functions such as CRC checking are utilized in the HDLC Decoder. The Decoder encompasses the certain functionality that of the descrambler, differential decoder, and parse ax.25. The execution of the decoder's algorithm however, is not equivalent to placing all three components together in assembly.

Methods

A series of tests were performed within a lab environment to observe the robustness of both waveforms as well as the software platforms in which they were tested with. All tests were conducted at the the Hawaii Space Flight Laboratory at the University of Hawaii at Manoa. A total of twelve test cases were performed with PropCube and HiakaSat. Six cases were set-up with NordiaSoft transmitting and receiving applications; three hardline and three over-the-air cases were done. The other Six cases were carried out using GRC as the transmit and receive platform. A hardline test using GRC as the transmit and receive software was used as a baseline test because of its ability to decode over 90% of the incoming packets.Two variations of the PropCube waveform was carried out as well as the HiakaSat waveform. The two variations of the PropCube waveform were demonstrated because the waveform going up from the ground station to the satellite and the waveform coming down from the satellite back to the ground station have slightly altered parameters such as the sync word and frequency.

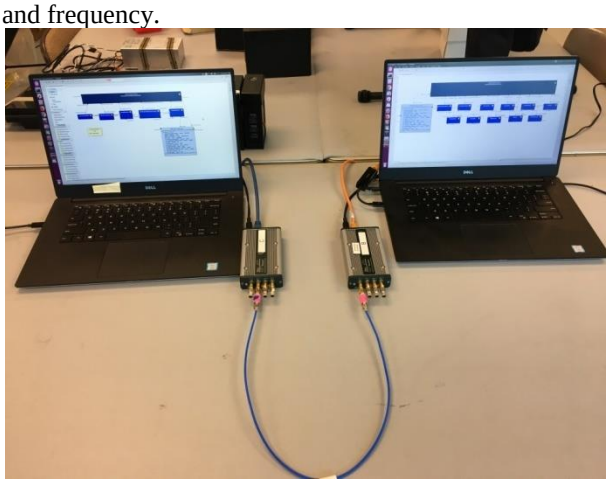


Figure 7: Lab set up with two E310 SDRs connected by an RF capable between the Tx and Rx ports

Here, we utilized the network mode capability of the Ettus E310 for narrow band signal observation and evaluation. The NordiaSoft license used for development and testing can only be used on x86 processors. Because the E310 uses an ARM based processor on board, we were unable to deploy the waveforms onto the radios themselves, thus requiring network mode. The RF cables were calibrated before testing using a Aniritsu vector network analyzer (VNA). A 30dB attenuator was placed at both transmit and receive ports of the radios for the hardline tests. For each test case, seven runs were done incrementally.

Figure 8: Lab set up with two E310 SDRs separated by a distance of 3ft with antennas at the Tx and Rx ports

The transmit signal gain was increased from 65dB to 100dB in increments of 5dB for all eight test cases. A 30dB attenuator was placed only at the transmitting port for the over-the-air tests due to the effect of space path loss on the signal gain. The obtained results comparing SCA with GRC are shown in figure 9 and 10. The presented data demonstrates the lack of robustness and stability in the waveforms. Figure 11 displays the wide range of the number of packets that were recovered between each run. There was no clear linear relationship between the increasing gain transmit levels and the total number of packets recovered. It was observed that the combining weight of the attenuators and RF cable relative to the size of the E310 caused the cables to bend downward at the connecting port preventing a snug connection between cable and port. A trend of packet recovery for the different sample rates that were tested with is shown in figure 12. To the right of the sample rate displayed on the plot is the data buffer sizes that were used on the receiving end of the test.

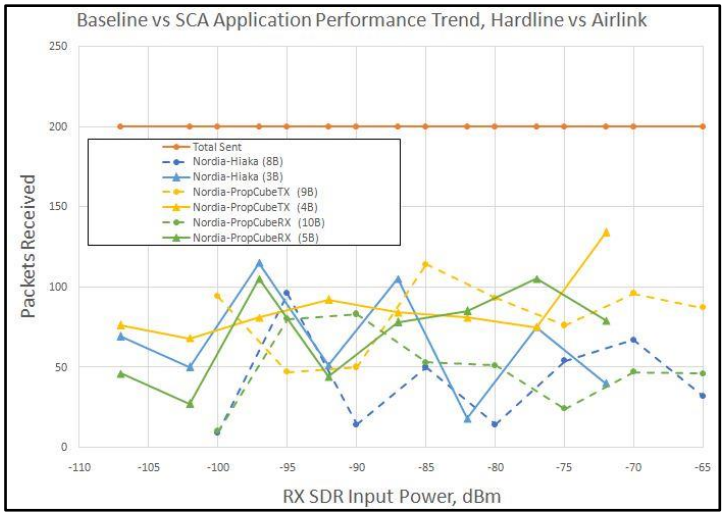


Figure 9: Number of packets received over the various different receive input power levels. Hardline and over-the-air tests were compared

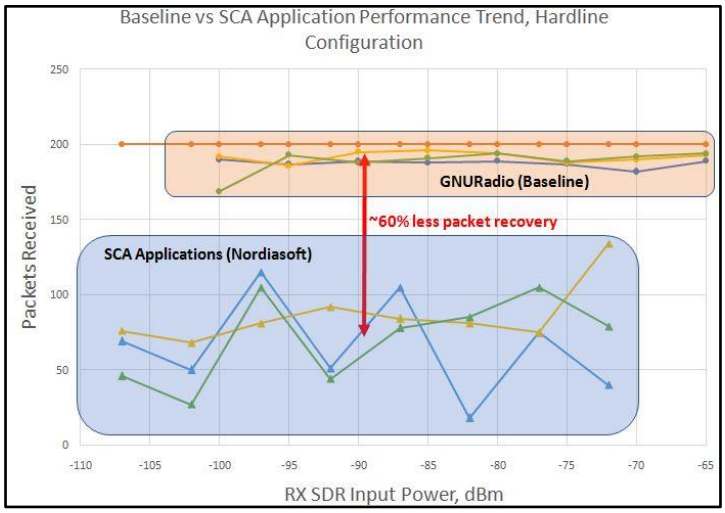


Figure 10: Number of packets received over the various different receive input power levels. Hardline and over-the-air tests were compared

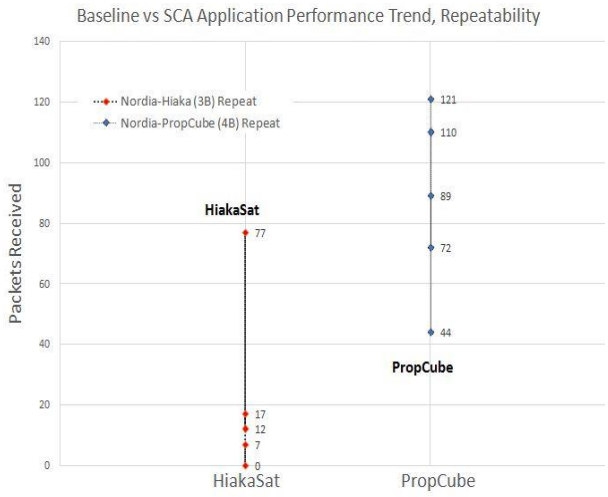


Figure 11: Comparison in stability of packet recovery for each test run between GRC and NordiaSoft

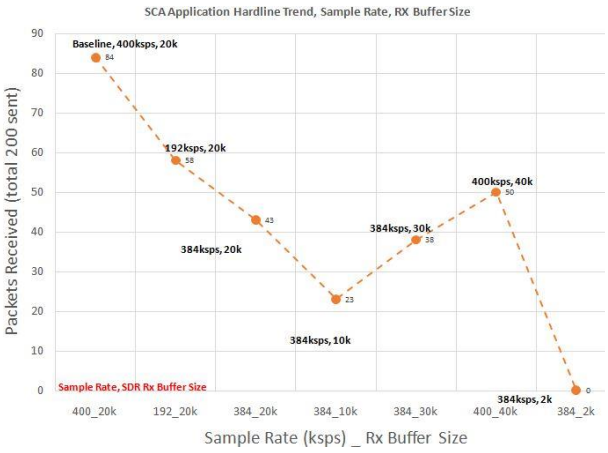


Figure 12: Number of packets received out of 200 packets sent over the various sample rates used for the hardline tests

Future Work

Our teams efforts to deploy our developed SCA waveform into an existing Mobile Cubesat Command & Control (MC3) ground station to collect real-time satellite pass data is currently in progress. All materials needed to integrate our current E310 USRP radio into an MC3 have been acquired. New sets of variables such as doppler shifting will need to be accounted for in the waveform application in order to communicate with LEO's. Our team has been proficiently putting together a test plan as well as the necessary link budget parameters for space telecommunications. Collaborative efforts with HSFL to design an end-user GUI application is also currently being worked on alongside the SDR integration efforts. The intentions of creating a graphical interface for the end-user is so our waveform applications can eventually become independent from NordiaSoft's eCo Suite tool kit. With an independent system, applications can be loaded and executed from the SDR without the need for a license commercial software. The eCo Suite licensed software that our team is currently using is also only SCA compliant for x86 systems.This license allows for development and testing applications within a lab environment. However, because the SDR's that we are developing with have ARM processor's on board, we will eventually need to convert over to an ARM eCo Suite license for deployment. The lab environment tests that were presented in this paper will also be validated using other SDRs such as the Ettus N300. We would like to demonstrate the cross-platform functionality of the SCA by loading and executing both waveforms on two separate SDRs to confirm the flexibility aspect of the SCA

Conclusion

In this paper we presented a lab environment test set-up between two SDRs. Direct connection tests and over-the-air tests were performed to juxtapose the quality and stability of the waveforms developed on the transmitting and receiving end. Our findings display some inconsistencies in data packet recovery for a given value in transmitting gain. Our teams efforts aim to narrow down any instabilities among the waveform with intentions of having a more robust application. Alongside these efforts, our group has begun work on integrating the Ettus E310 SDR into an existing MC3 ground station at the University of Hawaii. The integration of hardware into the MC3 has been started and after further validation of our waveform, a real-time satellite pass will be monitored and recorded. Once our waveform application has been fine tuned sort-of speak, our team aims to deploy the waveform application onto the E310's ARM processor for further space communication testing.

Acknowledgments

The work presented in this paper is funded on behalf of the Naval Information Warfare Center (NIWC) Pacific. Collaborative efforts from Members Giovanni Minelli from NPS and members Miguel Nunes, Eric Pilger, Isaac Rodrigues, and Kacey Hagi of HSFL made the research and development for this project a much faster process. Appreciationll members have contributed valuable time to make the results and future work for this project possible.

References

1. Kopitzki, J. "Development and implementation of a communication scheme for software defined radios." *Naval Post Graduate School* (2014)

2. Roehrig, J.M. "development of a versatile groundstation utilizing software defined radios." Naval Post Graduate School

3. Jondral, Friedrich K. "Software-defined radio: basics and evolution to cognitive radio." *EURASIP journal on wireless communications and networking* 2005, no. 3 (2005): 275-283

4. Bernier, Steve, Martin Phisel, and David Hagood. "INCREASING PERFORMANCES OF SCA APPLICATIONS THAT USE OPENCL."

5. Springer, Jonathan, Steve Bernier, James Ezick, Juan Pablo Zapata Zamora, and Janice McMahon. "Accelerating SCA compliance testing with advanced development tools." Analog Integrated Circuits and Signal Processing (2015): 1-13.

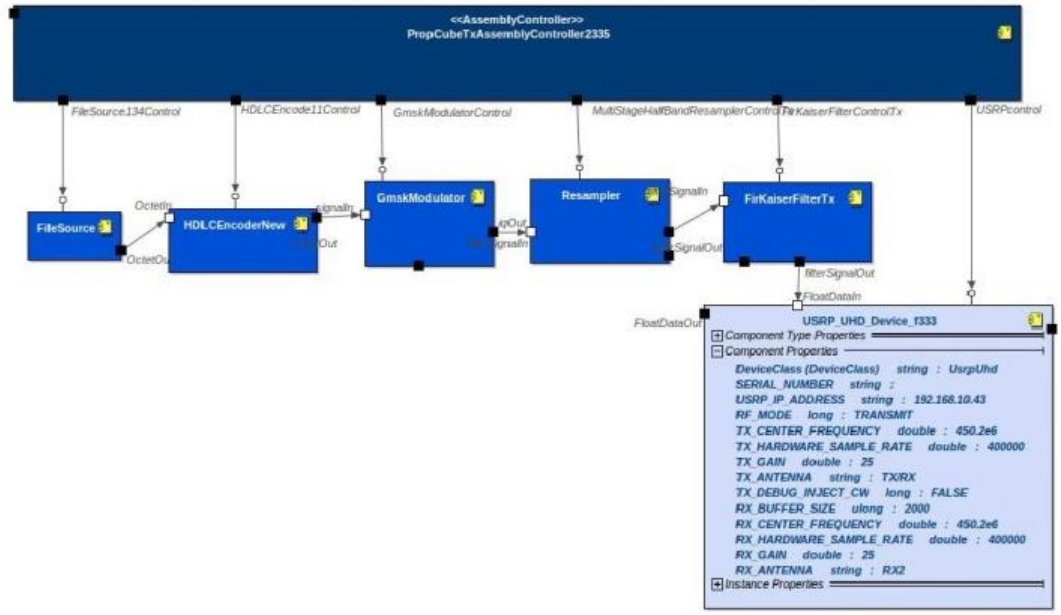
6. Ulversoy, Tore. "Software defined radio: Challenges and opportunities." IEEE Communications Surveys & Tutorials 12, no. 4 (2010): 531-550.

7. Bard, John, and Vincent J. Kovarik Jr. Software defined radio: the software communications architecture. Vol. 6. John Wiley & Sons, 2007.

8. Quinn, Todd, and Thomas Kacpura. "Strategic adaptation of SCA for STRS." (2007).

Appendix

a)



b)

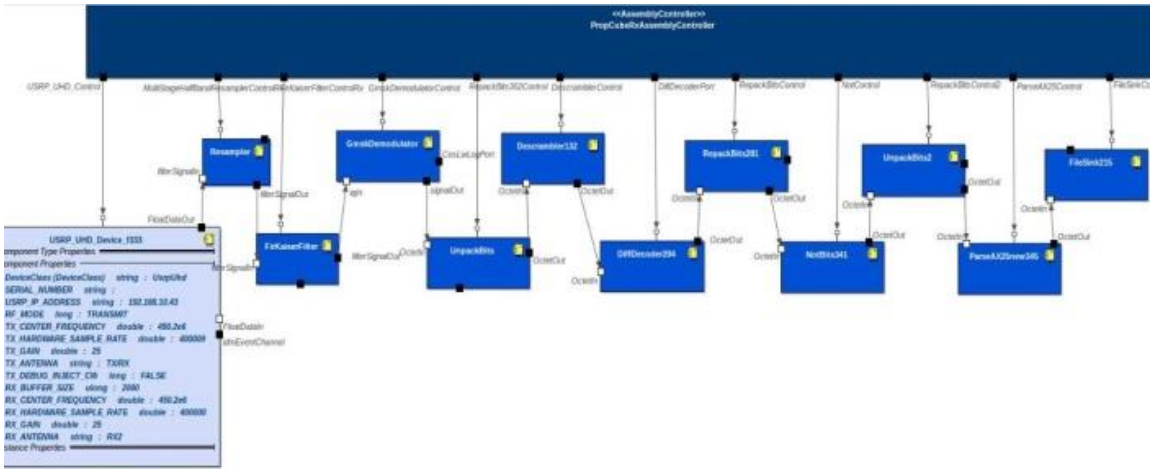
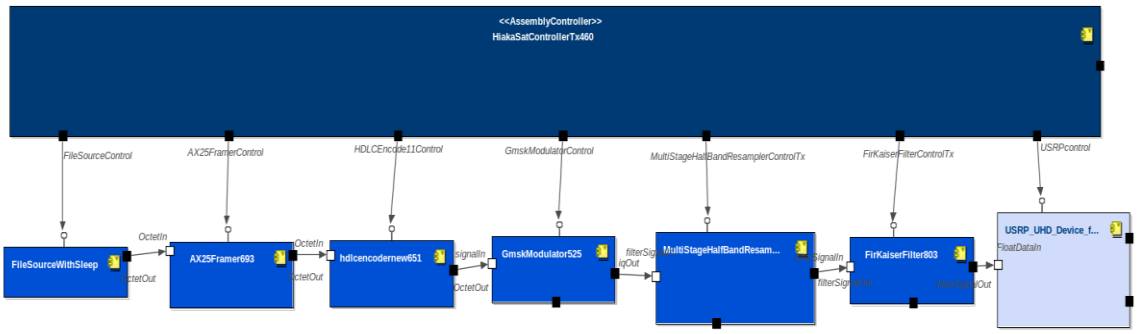


Figure 13: a) Transmitting and b) receiving block diagram for the PropCube waveform



b)

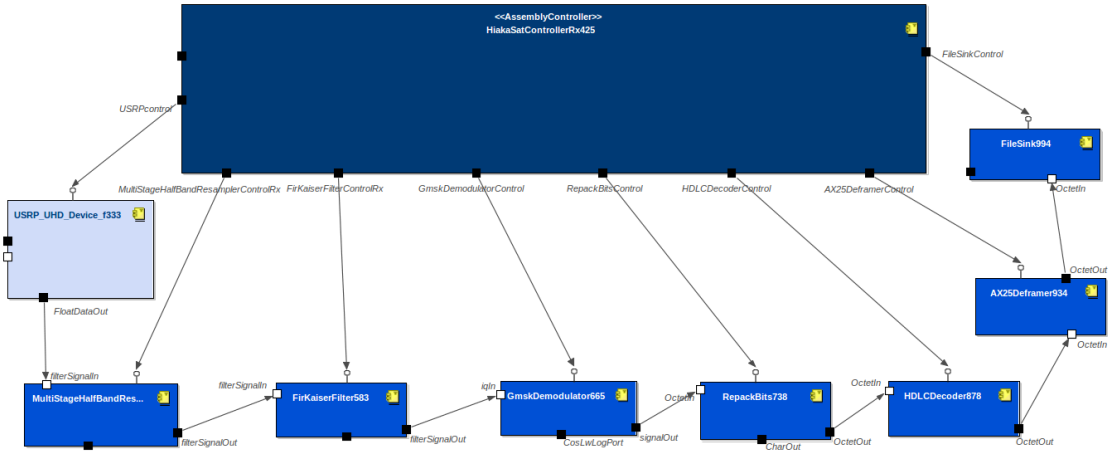


Figure 14: a) Transmitting and b) receiving block diagram for the HiakaSat waveform