# RAPID PORTING OF AN SCA-COMPLIANT FM3TR WAVEFORM

Per Johansson, Zhongren Cao, William Hodgkiss (California Institute for Telecommunications and Information Technology – Calit2, University of California San Diego, La Jolla, CA 92093; pjohansson@soe.ucsd.edu, zcao@soe.ucsd.edu, whodgkiss@ucsd.edu)

## ABSTRACT

In this paper, we describe the software architecture and our experience in porting an existing (voice) implementation of the FM3TR waveform to a new software defined radio (SDR) platform – the SDR-4000 from Spectrum Signal Processing. Data communications was not part of the original code base and was included as part of the project. Initially, analog communications was demonstrated at SDR-4000 IF (20 MHz) which included frequency hopping over a 3.2 MHz bandwidth. Then a National Instruments RF up-converter and down-converter was added to each SDR that enabled over-the-air communications between three SDR-4000 units. The entire project was completed in 9 months. The design and partition of the FM3TR waveform into SCA components across the GPP, DSP, and FPGA hardware of the SDR-4000 is discussed.

## 1. INTRODUCTION

The Joint Tactical Radio System (JTRS) program was established to provide next generation radio systems for US military forces [1]. By adopting software defined radio (SDR) technology and leveraging on the inherent programmability using software, the JTRS program aims to provide affordable, high-capacity, and flexible waveforms for rapid field deployment. The cornerstone for achieving this objective is the use of a standardized open architecture – the software communication architecture (SCA), which defines the common interfaces of waveform components [2].

SCA enables software reuse, hence, reduces the development time and associated costs. To facilitate software reuse, the JTRS Joint Program Executive Office (JPEO) maintains an Information Repository (IR), where waveforms are stored and shared by JTRS partners. Ideally, a new JTRS radio can be built efficiently by downloading existing waveform components from the JTRS IR and combining them with minimum changes on a specific host platform. Within the JTRS community, this process is called waveform porting. There are two levels of code portability that facilitate the waveform porting process [3]. First, the waveform components in the IR which are written for one particular host and/or environment can be moved to another with minimal changes to the original code. Second, regardless of diverse features and requirements of different host environments, there should be a common core framework defined to install, initiate and control waveforms and radio functionalities.

Here, we describe the porting of an existing implementation of the Future Multi-waveform Modular Tactical Radio (FM3TR) waveform in the IR to a different SDR platform – the SDR-4000 from Spectrum Signal Processing [4]. The FM3TR waveform was initially developed as an international cooperative effort between the United States, Germany, France and the United Kingdom to develop a reconfigurable communication system for ground and airborne applications. The Government Furnished Software (GFS) FM3TR source code is an implementation based on this international effort [5]. The FM3TR waveform implements frequency hopping over both VHF and UHF military bands (30 MHz - 400 MHz), using continuously variable slope delta (CVSD) modulation for voice digitizing. The FM3TR waveform defines two operational modes: voice and data [6]. The FM3TR waveform is an open standard and used as an instrument to promote international interoperability. It is also a reference waveform chosen by the SDR Forum [7] as a test and demonstration vehicle. Therefore, FM3TR is an ideal tool to promote the SCA and study waveform porting issues.

The SDR-4000 is a commercial off-the-shelf (COTS) multi-purpose software reconfigurable transceiver [4]. It comes with a comprehensive software stack including an SCA Core Framework with development tools, Spectrum's quicComm hardware abstraction layer and API library, and a real-time operating system (RTOS) with an integrated development environment.

In this paper, we will discuss in detail the design of the ported SCA-compliant FM3TR waveform. This will include a description of both the software portion of the waveform implemented in the SDR-4000 as well as the RF up/down-conversion portion of the waveform implemented in National Instruments hardware. The rest of this paper is organized as following. In Section 2, we introduce the targeted lab system setup. The design and partition of the FM3TR waveform into SCA-compliant components is discussed in Section 3. The porting experience is described in Section 4, followed by discussion and conclusions in Section 5.

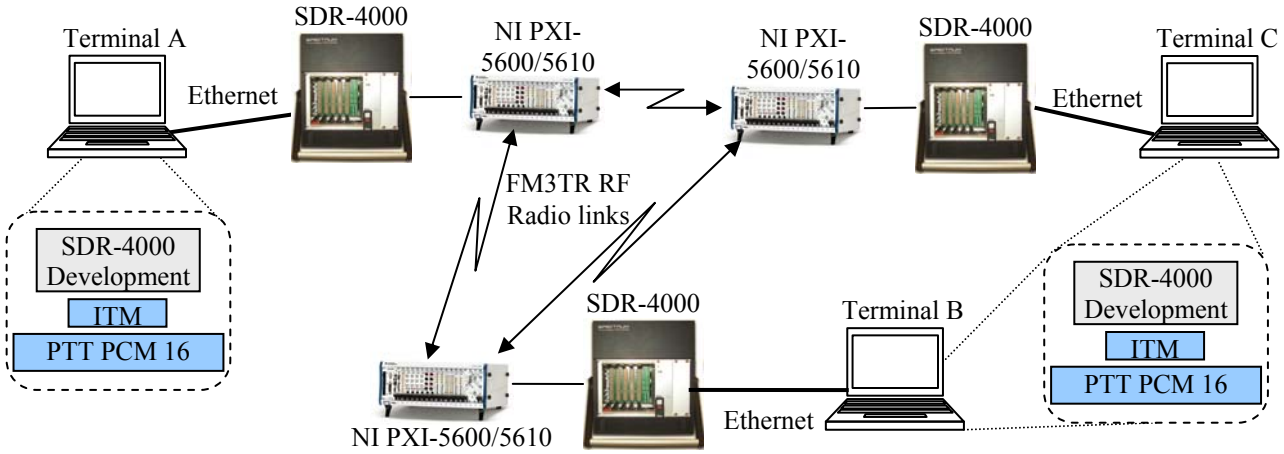


**Figure 1:FM3TR Lab System Setup**

## 2. FM3TR LAB SYSTEM SETUP

The targeted FM3TR lab system setup is illustrated in Figure 1. Three SDR-4000 units intercommunicate using the FM3TR waveform and support two applications: push-to-talk (PTT) and instant text messaging (ITM). The incoming/outgoing PTT (PCM samples) and ITM (ASCII messages) are carried in UPD/TCP/IP packets over Ethernet (100 Mbps) using available communication APIs supported by the GPP OS on the SDR-4000. Since it is a local communication link only and the applications are of a low-rate type, the TCP/IP/Ethernet performance is sufficient. In the lab setup, the SDR-4000 development laptops are hosting the PTT and ITM applications.

The FM3TR waveform has two operation modes: voice and data. The FM3TR voice mode is based on a 16 KHz PCM that is CVSD coded. The voice mode is utilized by the PTT application to carry voice between PTT "terminals", i.e. laptops in the lab configuration. Both the PTT application and FM3TR voice mode are well documented in the FM3TR base code, which was the existing implementation to be ported. The base code was ported with minor modifications. No provisions for media access control is provided for the voice mode, i.e. the PTT application does not first check if the media is busy before starting a PTT session but leaves it to the user to not initiate sending while receiving.

The FM3TR data mode uses the same basic radio transport channel (16 kbps) as the voice mode, but with additional Reed-Solomon (RS) coding to give better bit error protection, i.e. a robust mode is introduced for data traffic. This brings down the user data rate to 9.6 kbps. The FM3TR specification [7] describes an option for ARQ support which requires the system to be able to operate in frequency division duplex mode. In this optional mode, acknowledgments are sent simultaneously on a separate frequency, from the (data) receiving unit to the (data) sending unit. However, for this project only half-duplex mode is supported by the radio modem. A simple ARQ scheme for half-duplex data mode was included in the project to improve the data mode capability. This ARQ scheme is hosted by a data link control (DLC) component. No provision for media access control is assumed for the data mode (same as for the voice mode), i.e. messages are sent without a prior check to determine if the media is busy or not. If messages collide the system relies on the ARQ scheme to resend the packets.

A National Instruments up-converter (PXI-5610) and down-converter (PXI-5600) are used in the lab as the RF front-end for each unit. In the transmitter path, the PXI-5610 up converts the IF output of SDR-4000 to RF frequency. In the receiver path, the PXI-5600 down converts the FM3TR RF signal to IF frequency before being digitized by the SDR-4000 ADC.

## 3. SCA-BASED FM3TR COMPONENTS

This section describes the system components in terms of software functional blocks for the FM3TR voice mode and data mode implementation on the SDR-4000. These waveform components are a mix of ported FM3TR base code and new developed code for the data mode. The overall partition of the FM3TR waveform is depicted in Figure 2, with components existing in the base implemented code, newly added components, and OS supporting functions illustrated, respectively.

### 3.1. FM3TR SCA Components on the GPP

The following SCA resources (see Figure 2) reside on the SDR-4000 GPP:

- Net Device (Data/Voice): the Net Device functions handle the platform specific transport of voice and data packets between the SDR-4000 and the application PC.

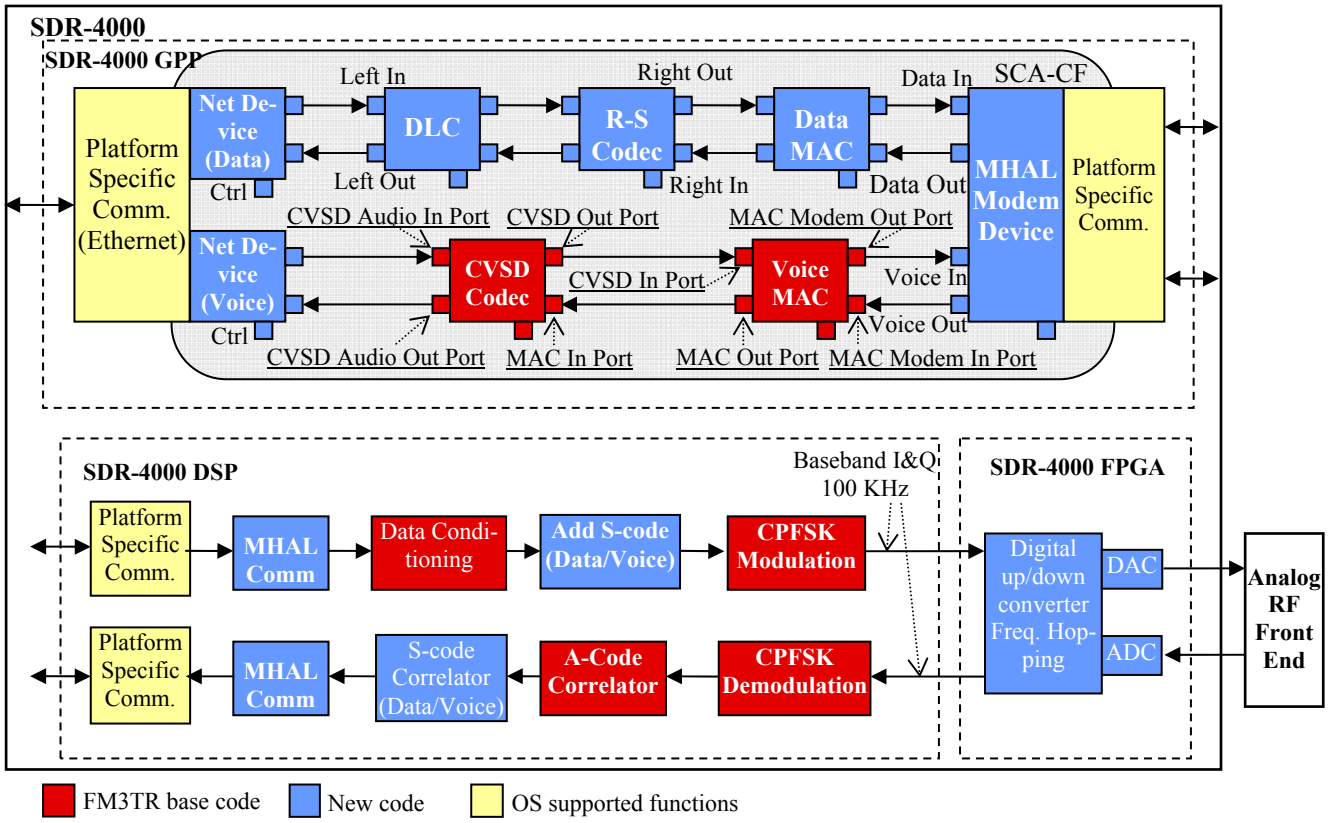


**Figure 2: FM3TW SCA-based Waveform Components for the SDR-4000 platform**

There is one Net Device component for each operational mode (voice and data) and the interfaces towards the GPP based FM3TR waveform components are SCA specific, i.e. CORBA based. The interface between the application PC and the SDR-4000 is TCP/IP/Ethernet based and will also host buffers to handle temporary rate differences between the units.

- Data Link Control (DLC): the DLC is an SCA resource that performs segmentation and reassembly of the data messages to and from the ITM application. The ARQ functionality also resides in the DLC resource.
- Reed-Solomon (R-S) Codec: the R-S Codec is an SCA resource that encodes outgoing data packets into an R-S block code and decodes received R-S encoded blocks. The system uses an RS (105, 72, 7) inner codec and optionally an outer per-hop RS (16, 14, 5) codec. The latter allows for an optional per hop ARQ scheme in full-duplex systems. This system will, however, be half duplex only.
- Data MAC: the Data MAC is an SCA resource that converts the format between data frames and data symbols grouped into 21 hops to match the R-S coding format. A total of 6 frames (24 data hops) are generated, including two end of message (EOM) frames that also carries data.
- Voice MAC: the Voice MAC is the original SCA resource ported from the base code. The CVSD encoded voice is sent/received on a per frame basis. At the end of a PTT session two EOM frames are sent out.
- MHAL Modem Device: an SCA compliant MHAL Modem Device resource encapsulates the voice and data packets (sent from the Voice MAC and Data MAC respectively) into the MHAL frame structure and sends the frames to a corresponding MHAL function in the DSP.
- CVSD Codec: the CVSD codec resource is ported directly from the FM3TR base code. The codec converts the 16 bit, 16 KHz PCM voice into 1-bit encoded voice at 16 kbps.

### 3.2. FM3TR SCA Components on the DSP

The DSP part of the SDR-4000 hosts the following FM3TR modem components:

- MHAL_Comm: the MHAL_Comm component takes care of the MHAL header encapsulation and de-capsulation of the packets sent to and received from the GPP.
- Data Conditioning: a preamble is added to the packet, consisting of a 2350-bit phasing pattern and a 256-bit synch pattern. The synch pattern contains a 32-bit pattern (A-code) that the receiver will correlate on to determine the start of a packet. Moreover, padding between the hops in each frame is added to account for the ramp-up and ramp-down delays

that occurs as the radio hops between frequencies (250 hops per second).

- S-code: in order to differentiate between voice, data and ACK messages, a combination of four 32-bit code words (S-codes) as part of the synch pattern, makes up code points to identify the packet types.
- CPFSK Modulation/Demodulation: a Continuous Phase Frequency Shift Keying modulation is used in FM3TR. The output of the modulator is a 100 KHz I&Q signal with 4 samples per symbol.
- A-code Correlator: after demodulation the receiver correlates for the A-code for start of message.
- S-code Correlator: once an A-code is found, correlation for the 4 subsequent S-codes follows to determine type of packet (voice, data, or ACK)
- Frequency Hopping: for each hop, a signal will be sent to the FPGA up/down converter to change the center (IF) frequency according to a predetermined hopping pattern.

### 3.3. FM3TR Components on FPGA

The FPGA of the SDR-4000 hosts the following functions:

- Digital Up-conversion: the 25 KHz baseband signal (sampled at 100 KSPS) is up-converted to a 20 MHz IF signal (sampled at 96 MSPS) that includes hopping within a 3.2 MHz frequency window at 250 hops per second. This signal is then sent to the system's DAC to be converted into an analog waveform.
- Digital Down-conversion: after ADC, the 20 MHz IF signal is down-converted and de-hopped to the I&Q signal at 100 KHz sample rate.
- Frequency Hopping: the up/down conversion changes the center frequency according to a pre-determined hopping pattern based upon signals sent from the DSP.

## 4. PORTING FM3TR TO SDR-4000

Each SDR-4000 IDS (Integrated Development System) unit includes the following hardware and software components.

The hardware includes: A modem unit consisting of two boards interconnected in a mezzanine configuration (200 MB/s full-duplex over ePMC). One board is the PRO-4600 SDR, which is a heterogeneous processing engine employing a combination of Xilinx Virtex-4 FPGA, TMS320C6416T DSP and MPC8541E GPP. The other board is the XMC-3321, which is a dual channel transceiver module that supports industry standard 10.7, 21.4 and 70 MHz IF frequencies through the use of dual 14-bit A/D converters sampling at up to 105 MSPS and dual 14-bit D/A converters sampling at up to 300 MSPS. A rear panel transition module (TM2-4900) provides connectivity to the PRO-4600 Gigabit Ethernet module, a high-speed serial, RS232 and JTAG interfaces. The chassis has five cPCI slots with front access for mounting modem board sets (PRO-4600 and XMC-3321) and rear access for mounting transition modules (TM2-4900) [4].

The software of the SDR-4000 platform includes: Green Hills INTEGRITY Operating System and MULTI IDE Integrated Development Environment; Green Hills Slingshot JTAG device and MPServe software; Spectrum's quicComm SW library; the PRO-4600 and XMC-3321 software API is provided via Spectrum's quicComm SDK, which abstracts the underlying hardware and provides users with basic link level access to communication between the processors in the SDR-4000; Windows XP development PC; Xilinx ISE Foundation FPGA Tools; Texas Instruments Code Composer Studio; SCARI++ Software Communications Architecture (SCA) Core Framework from CRC; ORB Express from Objective Interface Systems (OIS) [4].

Voice and data packets generated from the application PC pass through the FM3TR modem built on the SDR-4000 from the GPP to DSP, then FPGA before being sent out to RF. The GPP sends voice and data packets encapsulated in MHAL packets. The modem code on the DSP expands the data with additional headers and guard times. A four times up-sampling is also implemented per symbol. Frequency hop control data is embedded in the beginning of each hop sent to the FPGA. In the FPGA, further up-sampling is performed along with digital up conversion, which transforms the data into digital IF format for the DAC and RF up-conversion. The reverse process takes place in the receiver chain.

### 4.1. Porting FM3TR GPP Components

The CRC's SCA Architect enables and automates XML development with regard to the CORBA connections between components, as well component development utilizing multi-tier inheritance (properties, device types, devices, component types, components). In addition, SCA-compliant component skeleton code is generated to aid the porting of the component C++ code. This significantly reduces the coding effort.

The process of porting FM3TR to the SDR-4000 in an SCA sense involves two major steps -- node development and application development.

SCA devices belong to an organizational unit called a node, which contains an executable device that loads and executes the SCA application. Device components that interface external (actual) devices are also installed on the node, together with other essential SCA components such as the Domain Manager, Log Server, and an Event Channel (IDM Channel, which controls the state machine of every connected SCA device). The actual node itself is merely an XML construct, and with SCA Architect, provides for a very simple visual way of generating XML files. Nodes are used by the Device Manager in SCA to determine which

devices to load as well as which IDM Channel and Log Server connections to make and if any startup parameters are to be sent to each device. For the SDR-4000, node development consists of simply importing an already existing node implementation and appending the FM3TR devices developed for the FM3TR waveform, i.e., Net Devices, Voice devices and Modem devices. Once the node development was completed, the device binaries, as well as all generated XML files were installed into the SCA operating environment for the SDR-4000 unit, i.e. on the development laptop with an NFS path for the SDR-4000 to load the system.

SCA components are grouped into an organizational unit called an application. This is similar to how devices are grouped into a node, with the main difference that an SCA application describes and defines all the CORBA data connections and data flows that are made between components and devices (component-to-component as well as component-to-devices). Note that even if an SCA application is independent of hardware, it makes CORBA references to devices that it expects to exist on the target platform, and thus, can only be ported to platforms that provide the same device components (and same CORBA device names as well) in their respective nodes. In addition to the connections between application components, an extra component, known as the Assembly Controller, is responsible for starting, stopping, shutting down, and unloading any application component. Even though the Assembly Controller is a component itself, it is special in the sense that the application node designates it and will allow applications such as CRC's Radio Manager to install, configure, load/unload, and start/stop an installed application. Once the application component development is complete, an application must be exported to a zip file for installation by the Radio Manager (part of the SCARI++ software suite). After the application is installed, it can be started, stopped, or shut down. Logging of message monitoring, viewing connections between components and devices, etc. can also be done with the Radio Manager.

### 4.2. Porting FM3TR Modem (DSP) Components

The FM3TR modem base code was developed for a TI C5416 DSP and we needed to port it to a TI C6416 DSP on the SDR-4000 platform and add necessary inter-processor communication channels (to the GPP and the FPGA). The main tasks for the FM3TR modem porting to the SDR-4000 platform included:

- Porting the already implemented FM3TR voice (PTT) capability to the SDR-4000 Radio Hardware.
- Implementing the Data (ITM) capabilities for the FM3TR on the SDR-4000 in accordance with the FM3TR extended specification and design decisions.
- Use the MHAL specification for communication between the DSP and the GPP according to the MHAL specification.
- Develop code for frequency hopping control, which sends hopping signals to the FPGA on each frequency hop.

The porting of the FM3TR mode (DSP) code was straightforward and most of the code provided in the base modem implementation was reused. Modifications were made to adapt the code to the SDR-4000 platform and also a data mode was added.

A central controller was added for managing the request, processing and forwarding of packets to and from GPP and FPGA in order to decouple the base modem code from the SDR-4000 specific code. This is in line with the code portability requirement discussed in Section 1. A wrapper library was added for communication links to the FPGA and GPP (based on the quicComm library from Spectrum Signal Processing). During the process, the following was done to port the base modem implementation of FM3TR (voice only) and implement FM3TR data mode on the SDR-4000:

- Two new message types namely ITM and ITM_ACK were added.
- A new set of S-Codes to differentiate between VOICE, ITM and ITM_ACK messages were defined.
- The Receiver was modified to support detection of the different message types.
- The FM3TR modem control structure was modified to recognize ITM and ITM_ACK messages.
- The FM3TR modem was modified to send only Preamble and EOM for ITM_ACK packets, i.e. no actual data is sent.
- MHAL data encapsulation was added for the communication between the FM3TR application (GPP) and the FM3TR modem (DSP).

### 4.3. Implementing FM3TR Modem FPGA Components

Each FM3TR frame of voice or data consists of five hops, in which the first 4 hops carry information and the last hop contains a synchronization pattern. The transmission and receiving frequencies change for each hop, which is controlled by the FM3TR modem. The frequency hopping is realized in the digital domain via hopping the IF frequency before DAC. The DSP sends control messages to the FPGA, signaling the frequency of each hop. Note that the data frame detection is done by the FM3TR modem (DSP) functions, i.e. functions residing in the FPGA have no knowledge about whether the received data is valid or not.

During transmission, the control messages are embedded in the data flow as inbound control messages. Each hop starts with 5 zero symbols (20 samples) to give the transceiver guard time for the frequency change and, likewise, ends with 15 zero symbols (60 samples) for the next fre-

quency transition to follow. The first sample of a hop is modified to carry the frequency hopping message for this hop. This message is identified by the FPGA and switches the transmission frequency accordingly. The sample value is changed back to zero before transmission. Using these in-bound control messages for frequency hopping makes it easy to synchronize between the frequency hopping and the hop sequences of the data flow.

During reception, the DSP sends control messages to the FPGA to set the receiving frequency for the next hop as soon as it finishes receiving the 320 information samples (80 symbols) of the current packet. The control message is sent using the data channel as described for the transmission: when the FPGA receives this control message, it sets the receiving frequency accordingly.

Data transmission between the DSP and the FPGA uses the Spectrum quicComm channel interface. The interface to the DSP is based on C function library calls that send and receive data. To the FPGA, the interface is provided by the TEM (Transport Endpoint Manager), which defines signaling protocols to transmit and receive data. On both ends of the channel, there are buffers for the data. The quicComm channel works in the "burst transmission mode" in which the data stream is split and then packed into packets for transmission. The packet size must be a multiple of 64 bytes.

The digital IF up/down conversion functions are implemented in the FPGA on the XMC3321 board. There are two parts in the implementation. One is the signal processing part, which performs the up and down sampling and conversion between the baseband signal and the IF signal. The other part is a wrapper for the TEM interface, which provides a simple FIFO interface (much simpler than the TEM interface) for data communication to and from the channel.

The TEM wrapper was implemented in VHDL and has logic controls and transitions between different states. VHDL tools are easy to use for simulating these functions. The signal processing part is designed and implemented using the System Generator tool, which is a combination of the Matlab Simulink toolbox and the Xilinx FPGA synthesizer. The signal processing system was first designed in System Generator where simulations for system verification are performed. A netlist was then generated from the System Generator model and incorporated in the VHDL wrapper files for final bitstream generation. The results of the actual output can be compared with the System Generator model output to verify the accuracy of FPGA operations.

### 4.4. RF Front End

SDR-4000 only outputs or accepts IF signals. To perform IF-to-RF up/down conversion, National Instrument hardware is used consisting of an up-converter (NI PXI-5610), down-converter (NI PXI-5600), and an RF preamplifier (NI PXI-5690) with both fixed (+30dB) and variable (-10 to +20dB) gain in-box. A Virtual Instrument (VI) program using LabView was developed that controls and configures both the PXI-5610 and PXI-5600 for desired RF operation.

## 5. DISCUSSION AND CONCLUSIONS

The FM3TR porting project was a success and a real-time over-the-air demonstration of push-to-talk (PTT) and text messaging (ITM) applications was presented at the 3rd JTRS Science & Technology Forum [8]. Based on this experience, we believe that a successful waveform porting project always has two aspects – the radio system design and the SCA-compliant waveform partition. The radio communication functions, such as synchronization, protocols, etc., as well as operations of applications and how they fit on the target hardware platform will need to be investigated and verified. Some project or platform specific functions that do not exist in the base code for porting will need to be developed. At the same time, the partition of waveform into SCA-compliant components will need to be evaluated and revised. Furthermore, a reliable hardware platform and appropriate software tools can drastically reduce the development effort.

## 6. ACKNOWLEDGMENT

This work was supported by the Joint Program Executive Office of the Joint Tactical Radio System (JPEO JTRS) through SPAWAR Systems Center – Pacific under contract no. N66001-08-D-0155/T.O.0001.

## 7. REFERENCES

[1] http://jpeojtrs.mil
[2] "Software Communication Architecture Specifications," Version 2.2, Nov. 2001.
[3] D. Stephens, B. Salisbury, and K. Richardson, "JTRS Infrastructure Architecture and Standards," in Proceedings of the IEEE Military Communications Conference (MilCom 2006), Oct. 2006.
[4] Spectrum Signal Processing, "SDR-4000 Software Defined Radio Small Form Factor Transceiver Platform," http://www.spectrumsignal.com/products/3u/sdr_4000.
[5] J. Marks and R. Belly, "Software Radio Cooperative Research Project (SRCRP)," OMG Third Software-Based Communications (SBC) Workshop, March 2008.
[6] FM3TR Technology Group, "Test Waveform Specifications, Issue 1.0", February 4, 1998, Revised March 2006, document number FM3TR/TWS/v0.4/, June 4, 2006.
[7] http://www.sdrforum.org
[8] The 3rd JTRS Science & Technology Forum, 26-29 January 2009, http://jtrs.calit2.net.