Prototyping and Evaluation of Software Defined Radio using GNU Radio-USRP

Minseok KIM

† Graduate School of Science and Engineering, Tokyo Institute of Technology
S6-4, 2-12-1, O-okayama, Meguro-ku, Tokyo, 158-0094 Japan
E-mail: mskim@ide.titech.ac.jp

Abstract Recently, it is found that there are quite a large number of researches utilizing GNU Radio platform for the evaluation purpose. It is because GNU Radio is basically built on open source software which means that anyone can download the software and use it together with low cost hardware called USRP and off-the-shelf computers. It offers large amount of library for physical layer functions, which makes us easy to configure our own evaluation system with minimum effort on hardware. In this article, the concept, architecture of software/hardware and features and limitations will be reviewed. Moreover some of the latest researches on software defined radio and cognitive radios utilizing GNU Radio will be introduced.

Key words GNU Radio, USRP, software defined radio, open source, prototype, GPL

1. Introduction

So far, the concept of software defined radio (SDR) has been generating a lot interest from the pure hobbyists, academics and communication industry. As a result a number of software and hardware platforms have mushroomed on the scene in recent years, e.g., OSSIE (Open Source SCA Implementation - Embedded) project which is an SCA (Software Communications Architecture) implementation[1]. Moreover, FlexRadio with PowerSDR[2], HPSDR (High Performance SDR)[3], Simple Radio Peripheral[4] and so forth are some of the open source SDR projects which we can easily access.

Among all these, the GNU Radio project[5] has emerged and currently draws worldwide attention. The GNU Radio project began in 2001 and initially implemented by Eric Blossom. Later, additional work was completed as part of a United States National Science Foundation (NSF) grant. The project was aiming at making software radio technology more accessible with lower cost hardware called USRP
(Universal Software Radio Peripheral) and off-the-shelf computers. The GNU Radio software has a two level layered structure that C++ used to describe performance critical functions while Python is used to glue the C++ signal processing blocks into graphs. It uses a wrapper, SWIG (Simplified Wrapper Interface Generator) to interface the C++ code to Python. The USRP acts as the interface between the software and RF world, and USRP is virtually seen as a signal sink or source in software. Since all function of communication system from PHY, MAC to application can be described in software, it is indeed so-called software defined radio. So far there has been quite a lot of GNU Radio project application implementations. These include GSM Scanner, Open GNSS, IEEE802.11 WiFi protocol stack, RFID, Active/Passive Radar, OFDM, FM/AM/SSB Radio, and so forth.

Recently, we can find that there are quite a large number of researches utilizing GNU Radio platform for the evaluation purpose, (e.g. papers in [15], [16]). It is because GNU Radio is basically built on open source software, implying that anyone can download the software from the Internet and use it. The USRP is necessary but purchase it with reasonable price, which may be quite attractive for hobbyists, student, staffs in university and academic institution. For industry, it must be advantageous that it also made rapid prototyping possible without minimum cost. In this article, the basic information and features of the GNU Radio and USRP will be reviewed and the limitation will be also discussed. Then some of the researches on software defined radio and cognitive radios utilizing GNU Radio will be introduced.

2. What is GNU Radio and USRP

2.1 GNU Radio

According to GNU Radio Wiki [5], “GNU Radio is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors. It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems.” As well described above, GNU Radio is an open-source signal processing package mainly used for building various communication systems as well as software defined radios (SDR) and cognitive radios (CR).

As the brief description of the web site, GNU Radio top-level applications can be primarily written using the Python programming language which ‘glues’ processing blocks into a ‘graph’ as shown in Fig. 1, where the supplied, performance-critical signal processing path is implemented in C++. Currently there exist quite a growing numbers of libraries for basic functions, such as modulation, demodulation, filtering and so forth. SWIG is used to wrap these C++ libraries into codes for use with the Python scripting language. As far as Python is concerned, C++ blocks are regarded as just interfaces or black boxes, and Python doesn’t care what happens inside them. Fig. 2 shows some of the available modules within the current GNU Radio software library. Since this is an ongoing project, more blocks from various contributors keep being added.

The lowest level of the software is real-time processing block in FPGA (field programmable gate array) which is described by Verilog HDL, and compiled and synthesized by Altera Quartus II. From the layered architecture for application development, the user is able to implement real-time,
high-throughput radio systems in a simple-to-use, rapid-
application-development environment. It is noted that GNU
Radio is licensed under the GNU General Public License
(GPL) version 3 and all of the code is copyright of the Free
Software Foundation. Ettus research LLC is providing with
USRP which can be used to develop and implement vari-
ous software radio applications using GNU Radio software
package [7].

2.2 The USRP (Universal Software Radio Peri-
pheral)

The USRP allows to create a software radio using any com-
puter with a USB 2.0 for USRP or Gigabit Ethernet port
for USRP2 which has been released in Sept. 2008 improv-
ing some limitations of the USRP and enhancing the func-
tions. The specification is provided in Table 3. The entire
design of the USRPs is open source. As described above, the
USRP and USRP2 (simply USRP hereafter) complements
the GNU Radio software in the building of software radio
prototype as shown in Fig. 3. It consists of the mother-
board which is common part with primary function, and RF
daughterboards. Various plug-on RF daughterboards allow
the USRP to be used on various radio frequency bands. Cur-
cently daughterboards are available from DC to 5.9 GHz as
presented in Table 2 [7].

2.2.1 Receive Path

The basic architecture of GNU Radio and USRP is as
shown in Fig. 3. First, the daughterboard downconverts the
frequency of the received signal from RF (radio frequency) to
IF (intermediate frequency) around DC before sending it to
the analog-to-digital converter (ADC). In the USRP, there
are 4 high-speed 12-bit ADCs with a sampling rate of 64
Mps (Mega samples per second). Next, after the signal is
sampled, it is transferred to the FPGA. The main task of
the FPGA is residual frequency down conversion and rate
conversion.

Table 1: Comparison of USRP and USRP2

<table>
<thead>
<tr>
<th></th>
<th>USRP</th>
<th>USRP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>USB 2.0</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>FPGA</td>
<td>Altera</td>
<td>Xilinx</td>
</tr>
<tr>
<td></td>
<td>Cyclone EP1C12</td>
<td>Spartan 3 2000</td>
</tr>
<tr>
<td>RF Bandwidth</td>
<td>8 MHz @ 16bits</td>
<td>25 MHz @ 16bits</td>
</tr>
<tr>
<td>Cost</td>
<td>$700</td>
<td>$1,400</td>
</tr>
<tr>
<td>ADC</td>
<td>12-bit, 64 MS/s</td>
<td>14-bit, 100 MS/s</td>
</tr>
<tr>
<td>DAC</td>
<td>14-bit, 128 MS/s</td>
<td>16-bit, 400 MS/s</td>
</tr>
<tr>
<td>D’board capacity</td>
<td>2 TX, 2 RX</td>
<td>1 TX, 1 RX</td>
</tr>
<tr>
<td>SRAM</td>
<td>None</td>
<td>1 Megabyte</td>
</tr>
<tr>
<td>Power</td>
<td>6V, 3A</td>
<td>6V, 3A</td>
</tr>
</tbody>
</table>

Signal flow within the USRP is shown in Fig. 4, which
shows the main components of the USRP motherboard. The
standard FPGA configuration consists of 4 DDCs, but 2 of
them are not connected to half band digital filters due to
the lack of resource. This allows 1, 2 or 4 separate receive
channels. In the 4 DDC implementation, in the receive path
we have 4 ADCs, and 4 DDCs. Each DDC has two inputs of
I and Q. Each of the 4 ADCs can be routed to either of I or

...
Q input of any of 4 DDCs. This allows us to have multiple channels selected out of the same ADC sample stream [8].

In the FPGA, downconversion of the signal from IF to baseband and the decimation of the signal samples so that the data rate can be adapted by the performance of the transmission interface (USB 2.0 or Gigabit Ethernet) and the computers’ computing capability. The function of DDC in FPGA is as described in Fig. 5. The complex IF input signal is multiplied by the NCO (numerical controlled oscillator). The resulting signal is also complex and centered at DC. Then the signal is decimated with a factor N. The DDC employs 4 stage cascaded integrator-comb (CIC) filters, which are high performance filters that only use addition and subtraction without any multipliers. 31-tap half band filters are cascaded with the CIC filters to perform the complete DDC stage [8]. The resulting signal is complex baseband signal centered at zero frequency and the data rates are low enough to be transferred to the computer.

2.2.2 Transmit Path

In the opposite path of transmitter, logic is more or less the same except that everything is logically reversed. Through the USB 2.0 or Gigabit Ethernet a complex baseband I/Q signal is sent to USRP. Then it is interpolated and digitally up-converted to IF by the DUCs (digital up converters) on the mixed signal processor before the DAC where the signal is converted into analog and eventually the RF board takes over.

The DUC on the transmitter side are actually contained in the AD9862 CODEC chips (Analog Devices), not in the FPGA as shown in Fig. 4. The only transmitter signal processing blocks in the FPGA are the CIC interpolators. The interpolator outputs can be routed to any of the 4 CODEC inputs. The sampling rate of DAC conversion is 128 Msps.

2.2.3 RF Daughterboards

The mother board has four slots, which allows to plug in up to 2 receive basic daughter boards and 2 transmit basic daughter boards or 2 transceiver boards. Each daughter board slot has access to 2 of the 4 ADC input / DAC output.

This allows each daughter board to have 2 independent RF sections (real sampling), or single RF section (complex I/Q sampling), but actually each board supports a single RF section, for a total of 2 for the whole system. It should be noted that no anti-alias or reconstruction filtering is provided on the USRP motherboard for maximum flexibility in frequency planning for the daughterboards. The daughterboards available now are presented as Table 2 [7], where all transceivers come with 70 B AGC unless specified otherwise.

Actually the transceiver boards have quadrature direct conversion architecture. Therefore the center frequency tuning consists of two step process. Due to the Phase-Locked Loop (PLL) step size in the RF daughter board, first the RF front-end roughly tunes to the desired center frequency as close as possible. Next, the result of this operation and our target center frequency is used to determine the resid-
ual frequency used for down-conversion by the digital down converter (DDC). In FPGA, the phase generator in NOC is clocked at 64 MHz and NCO output has 32 bits of resolution. Therefore the frequency resolution is 0.014 (= 64MHz/2^{32}) in Hz.

3. Features and Limitations of USRP

As described so far, GNU Radio provide a tool for rapid and easy realization of software defined radio with USRP. To make GNU Radio-USRP based system most flexible there should be some hardware limitations. This section discusses the features and limitation of USRP for further understanding.

- Frequency and Bandwidth
  As Table 2, daughterboards mounted on the USRP provide flexible RF front-ends. A wide variety of available daughterboards allows to use different frequencies for a broad range of applications from DC to 5.9 GHz. Regarding the bandwidth which can be managed for real-time processing, we can sustain 32 MB/sec across the USB. All samples sent over the USB interface are in 16-bit signed integers in I/Q format which means 4 bytes per complex sample. This results in 8 Msps for I, Q respectively across the USB. Since complex processing was used, this provides a maximum effective total spectral bandwidth of about 8 MHz by Nyquist criteria. As shown in Table 1, USRP2 was developed with wider bandwidth of 25 MHz by employing Gigabit Ethernet interface.

- Roll-off Characteristic in DDC
  The FPGA on USRP performs decimation on the signal after it has passed through the ADC. This decimation is done in the DDC by a 4-stage CIC filter and a halfband filter (Fig. 5). It produces a roll-off characteristic, resulting in the curved spectrum at around edge. The frequency response of the cascaded CIC and half band filter is shown in Fig. 6 where the decimation factor of 16 results in the data rates of 4 MHz. From this frequency response, we can see that roll-off at half data rate (2 MHz in Fig. 6) is about -9.6 dB to the level of DC and the cut-off frequency is about the quarter data rate. Therefore half of the decimated data rates are actual bandwidth of pass band RF signal.

- Multiple Channel Extension and Synchronization
  To realize multiple antenna system in multiple USRPs, the USRPs should be clock-synchronized. Clock synchronization is possible by introducing the daisy chain of the clock in master USRP which needs some modification of the circuit. Coherent operation between multiple USRPs is also available for receive side with some modification of FPGA and host coding. For transmit, DUC should be implemented on FPGA. This is because there’s no way to directly control the phase accumulator in mixed signal processor, AD9862.

In case of USRP2, it offers MIMO cable for this purpose. Please refer to [10] for more detail.

- Analog Filters
  As mentioned above, no anti-alias or reconstruction filtering (baseband filter) is provided on the USRP motherboard for maximum flexibility in frequency planning for the daughterboards. Depending on daughter boards, there is no bandpass filter mounted on the circuit.

- DC offset and I/Q Imbalance Compensation
  Every daughterboard has its own calibration data for DC offset or I/Q imbalance compensation stored in onboard EEPROM. Daughter boards basically include single LSI for RF front end and local oscillator circuit. Thus the RF performance totally depends on that of RF LSI.

4. Implementation Examples in Academic Papers

In this section, some implementation examples using GNU Radio are presented.


This subsection introduces an implementation example of cooperative spectrum sensing system for emergency radio [13] using GNU Radio-USRP. Individual sensor nodes sense
the spectrum of interest, and then send to the head node either the 1-bit decision data or the received power data. The head node will then determine which frequency bands are currently being used in that particular cluster area. The section presents the spectrum sensing in individual node implemented on USRP is present. For more detail about the system some literatures can be referred [11]–[13].

GNU Radio also offers many example programs written in Python such as an FM transmitter, receiver and a signal generator. For spectrum sensing purpose, ‘usrp_spectrumSense.py’ is provided in the package, which is similar to the function of spectrum analyzer. ‘usrp_spectrumSense.py’ connects various DSP blocks to perform the Fast Fourier Transform (FFT) on time domain data, and outputs the FFT data. In this program, frequency sweep is done automatically on the Python, which can scan the wide range of frequency band quickly. The flowchart is shown in Fig. 7.

As shown in Fig. 8, DSP blocks in GNU Radio are divided into a few modules. In Python, the required blocks are connected by first importing their respective modules. We have developed a new block in GNU Radio and have modified ‘usrp_spectrumSense.py’ to output the FFT data directly into Matlab for flexible post processing. This new block is actually a modified version of the ‘bin_statistics_f’ which can be found in the GR module. A detailed explanation of the blocks can be found in Table 3.

By using all the blocks described in Table 3, the processing flowchart will be as shown in Fig. 9. It takes the average of periodograms, and Matlab will read the output directly.

For flexibility of processing, Matlab was used to implement the second half of the spectrum sensing program. GNU Radio outputs the spectrum in small segment of frequency band of interest based scanning the center frequency. Once Matlab receives these individual sections from GNU Radio, it will combine them to display the PSD of the entire frequency sweep. In the hard decision combining scheme, Matlab will compare each of the power spectrum to a threshold, decide whether a primary signal exists or not, and send this decision data to the head node using TCP/IP implemented in Python script. In the soft decision combining scheme, Matlab will just record the power spectrum and send them to the head node.

In USRP, due to the roll-off characteristics of CIC filter, it results in the curved noise floor. Here the decimation rate of the CIC filter and half band filter are 8 and 2 respectively, resulting in a $64/(8 \times 2) = 4$ MHz bandwidth. As mentioned in previous section, the attenuation at edge frequency is quite large. In order to avoid this slow filter roll-off, 25% of both ends of the spectrum was discarded, and the

---

**Table 3** Detailed description of each DSP block

<table>
<thead>
<tr>
<th>Block Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Interface to USRP Rx path. Copy samples from the USRP buffer. Output the complex sample data into a stream of items.</td>
</tr>
<tr>
<td>stream_to_vector</td>
<td>Convert a stream of items into vectors of specified length. In our program, this vector length equals FFT size.</td>
</tr>
<tr>
<td>fft_vcc</td>
<td>Compute FFT. Input complex vectors. Output complex vectors. Able to specify type of windowing function to be used. Default window function is the four term Blackman Harris window with a $-92$dB side-lobe.</td>
</tr>
<tr>
<td>complex_to_mag_squared</td>
<td>Compute the magnitude square of the input. Input complex vector. Output float vector. This block can be used with fft_vcc to compute the periodogram.</td>
</tr>
<tr>
<td>modified_bin_statistics_f</td>
<td>Automatically tunes the USRP to sweep a wide bandwidth in steps based on the start and end frequency specified by the user. Computes the average of inputs over a time period determined by the dwell_delay variable (sensing time). After calculating the average, it composes a message and inserts it into the message queue. This block is a modified version of bin_statistics_f from the GR module.</td>
</tr>
</tbody>
</table>
frequency response of the CIC and half band filter was compensated from the spectrum. The hardware also introduces a spike (DC offset) in the output at the center frequency. We avoided this spike altogether by dividing the remaining spectrum into three equal parts, and ignoring the middle part. Figure 10 shows the spectrum combining scheme to sweep a wide bandwidth [17]. The center frequency is incremented by 1 or 3 MHz, and the separate parts are combined in Matlab.

In this study, the authors proposed a cooperative spectrum sensing system using 4 USRPs. They also demonstrated the performance compared to the fundamental theory and found GNU Radio & USRP setup performs closely to what the theory predicts. The system will be demonstrated in [18].

4.2 Other Examples Demonstrated in IEEE DySPAN 2010

In this subsection, some selected recent researches utilizing GNU Radio and USRP which have been presented in IEEE DySPAN 2010 are briefly introduced. For more detail, see the references.

4.2.1 OFDM-based Dynamic Spectrum Access [19]

In this demonstration, the authors implemented interference-free coexistence of two OFDM-based systems within a common frequency band with optimally configured transmission parameters for given system constraints in GNU Radio framework. From the demonstration, they illustrated that a highly reconfigurable framework allows for implementation and evaluation of various transmission strategies for different DSA scenarios, different classes of given requirements and various sets of controllable parameters.

4.2.2 OFDM Pulse-Shaped Waveforms for Dynamic Spectrum Access Networks[20]

In this demonstration the authors presented a dynamic spectrum access network which employs a reconfigurable orthogonal frequency-division multiplexing (OFDM) based waveform. In order to avoid the creation of harmful interference, the out-of-band (OOB) emissions of the waveform are dynamically tailored to the properties of spectrum neighbors through the use of OFDM pulse shaping. The demonstration network was built upon Iris 2.0 software radio platform (not GNU Radio). However the USRP was adopted as the minimal hardware RF front end. They illustrated the capabilities of this platform as well as the utility of OFDM pulse shaping in the context of dynamic spectrum access networks in the demonstration.

4.2.3 Demonstration of Sequence Detection Algorithms for Dynamic Spectrum Access Networks [21]

This work implemented a sequence detection spectrum sensing system with GNU Radio and USRP. The proposed algorithms considered utilizing the transition information in the primary user’s channel access for better sensing performance. The demonstration consisted of primary and secondary data links that are both streaming video in the same frequency band, forcing the secondary link to opportunistically access the spectrum. Various sensing algorithms are selectable and configurable at the secondary transmitter, including multiple sequence detection algorithms and energy detection. The superior performance of the sequence detection algorithms is evident at the receivers through a number of metrics, including video quality, plots of the historical data rate, and estimates of the detection and false alarm probabilities. The physical setup for the demonstration consists of two laptop computers running GNU Radio, each of which is connected to two Ettus USRPs. For performance reasons, they emphasized that the sensing schemes and channel access decisions were all performed in the FPGA on board the USRP.

5. Summary

In this article, the basic information and features of the GNU Radio and USRP was reviewed and some limitation was also discussed. Then some of the researches on utilizing GNU Radio were introduced. It can be seen that GNU Radio-USRP can be a low cost common evaluation tool in academic field thanks to the feature of low cost and open source concept. Now there is some discussion group in the world for all things related to GNU Radio including the usage and problems [22]. It is also another advantage to obtain various information easily from such open discussions.

Reference

howto-write-a-block.html

[22] GNU Radio discussion group, discuss-gnuradio@gnu.org