社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SR2008-94(2009-03)

Implementation of a Channel Sounder using GNU Radio Opensource SDR Platform

Mutsawashe GAHADZA[†], Minseok KIM[†], and Jun-ichi TAKADA[†]

† Graduate School of Engineering, Tokyo Institute of Technology
2-12-1 O-okayama, Meguro-ku, Tokyo, 152-8550 Japan
E-mail: †{mutsa,mskim,takada}@ap.ide.titech.ac.jp

Abstract GNU Radio refers to some open source software which, together with low cost hardware called USRP, can be used to realize a software radio platform. Cognisant of the possible practical limitations inherent to GNU-radio technology, this research seeks to explore the suitability of this technology in realization of a software radio platform. A PN sequence based channel sounding system with two Linux based host PC and two USRPs configured in the Master - Slave mode, was therefore implemented as a target application.

Key words GNU Radio, Channel sounding, USRP, PN, FPGA, SDR

1. Introduction

The concept of software defined radio (SDR) has been generating a lot interest from various quarters, from the pure hobbyist through to academics and business minded people. The benefits to SDR include reconfigurability and possibility of rapid prototyping among others. As a result a number of software and hardware platforms have mushroomed on the scene in recent years. Virginia Tech came up with the OSSIE (Open Source SCA Implementation - Embedded) project which is an SCA(Software Communications Architecture) implementation. FlexRadio PowerSDR, HPSDR(High Performance SDR), Simple Radio Peripheral are some of the open source SDR projects. Among all these, the GNU Radio project has emerged as one the most exciting ones. The GNU Radio technology provides an opensource software platform which together with low cost hardware called USRP (Universal Software Radio Peripheral) can be used to develop and implement various software radio applications. The GNU Radio software has a two level layered structure, with C++ performing performance critical functions while Python is used to glue the C++ signal processing blocks into graphs. The software uses SWIG (Simplified Wrapper Interface Generator) to interface the C++ code to Python. The USRP acts as the interface between the software world and the RF world. There has been quite a lot of GNU radio project application implementations. These include GSM Scanner, Open GNSS, IEEE802.11 WiFi Stack, RFID, Active/Passive Radar, OFDM, FM/AM/SSB Radio among other projects. Of the tens of possible applications channel sounding seems to be one of the interesting applications from the economic point of view. The idea is to measure and evaluate channel sounding capabilities of the GNU Radio using a maximal PN (Pseudo Noise) sequence based channel sounder application within the GNU Radio software. GNU Radio is basically built on open source software, implying that anyone can download the software from the Internet and use it. The USRP has to be purchased but it costs a minimum price. All this implies is that one can quickly build software radios at a very low cost. Rapid prototyping of wireless communication systems is also made possible.

This paper first gives an overview of GNU Radio Software, the USRP and PN sequence based channel sounding. After that the measurements methodology undertaken is explained before the obtained results are presented discussed. Subsequently conclusions are drawn from these results before laying out the future works.

2. GNU Radio Development Tools

2.1 GNU Radio Software

GNU Radio may be defined as ‘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’ [1]. It is licensed under the free software foundation’s GPL (General Public License) conditions.

Figure 1 shows the sofware architecture of the GNU Radio software. At the highest level is a script language called

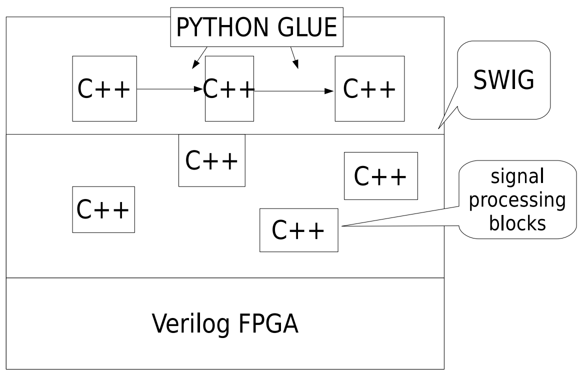


Figure 1　GNU Radio software architecture.

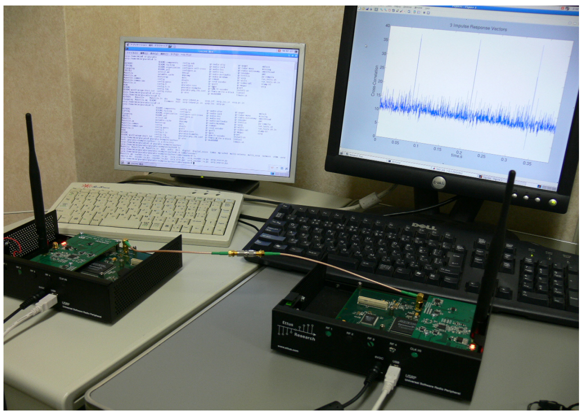


Figure 2　USRP (universal software radio peripheral).

Python. Writting applications follows two software coding stages. Signal processing blocks are written in C++ and currently there exist quite a growing library of these. The next stage involves writting a python application to 'glue' these into a 'graph'. There is an excellent tutorial on how to write signal processing blocks at [2]. As far as Python is concerned, C++ blocks are just interfaces or black boxes, and Python doesn't care what happens inside them. Figure 4 shows some of the availlable modules within the current GNU radio software library. Open source g++ compiler is used for the c++ code. Since this is an ongoing project many more blocks from various contributers keep being added. The lowest level of the software is Verilog's HDL code is compiled and synthesized using Verilog HDL for the FPGA (field programmable gate array).

2.2 The USRP (Universal Software Radio Peripheral)

The USRP complements the GNU Radio software in the building of software radios. It consists of two boards, the motherboard and RF daughterboards. When the USRP daughterboard receives an RF signal, it downcoverts the signal to IF which the ADC on the motherboard can digitize by oversampling.The received complex signal goes through the FPGA which essentially implements a DDC (digital down converter). Finally, the I and Q signals are then decimated in order to condition their data rate to the capabilities of the USB. The host then receives the signal through the USB. In the opposite direction the processing logic is more or less the same except that everything is logically reversed. From the USB the complex signal has to be interpolated and goes digitally upconverted by the DUCs (digital down converters) on the mixed signal processor before the DAC converts the signal into analog form and eventually the RF takes over. The specifications of the USRP and daughterboard used in this study are shown in Table 1. Signal flow within the USRP may be understood by analyzing Fig. 3, which shows the main components of the USRP motherboard.

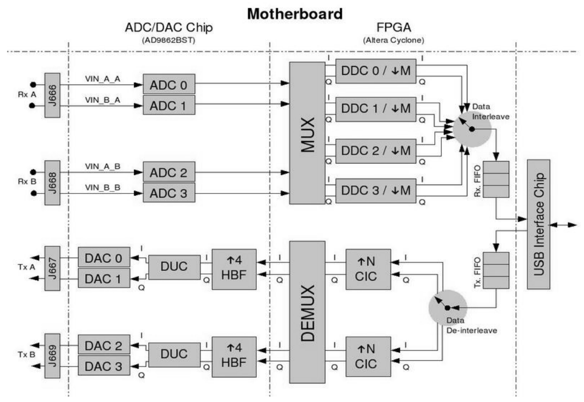


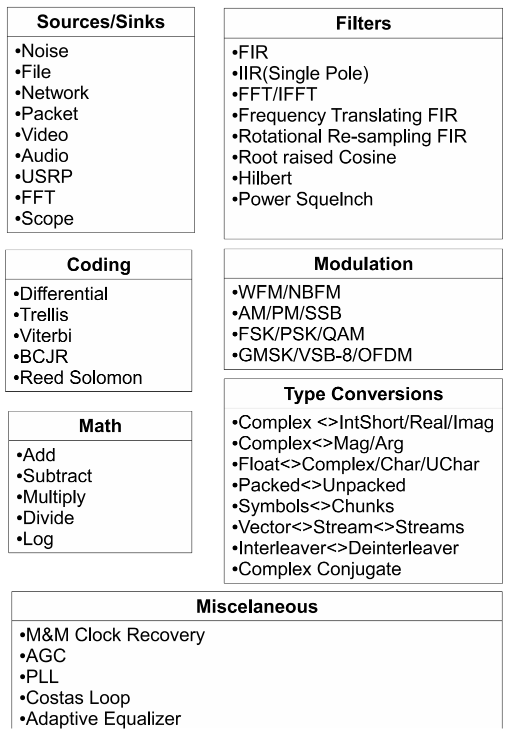
Figure 3　The USRP motherboard [4]

Table 1　USRP Component Specifications

Component	Specifications.
ADC/DAC Chip Analog Devices AD9862BST	**12 bit ADCs** 64MSa/s,Vin=2Vpp@50 Ohm (40mW/10dBm) **14bit DACs** 128MSa/s,Vout=2Vpp (10mW/10dBm) Programmable gain amplifier (up to 20dB) Interpolation Filters
Altera FPGA	2 DDCs each with I and Q Inputs Decimation filters Max User I/O -173 two PLL RAM - 239 616bits
XCVR2450 daughterboard	2.4-2.5 GHz - 100mW 4.9-5.9 GHz - 50mW Antennae - Sleeve Dipole
Cypress FX2 USB Controller	8051 microprocessor based

2.3 Channel Sounding Implementation

Generally channel sounders maybe classified into three as itemized below.



Sources/Sinks
- Noise
- File
- Network
- Packet
- Video
- Audio
- USRP
- FFT
- Scope

Filters
- FIR
- IIR(Single Pole)
- FFT/IFFT
- Frequency Translating FIR
- Rotational Re-sampling FIR
- Root raised Cosine
- Hilbert
- Power Squelnch

Coding
- Differential
- Trellis
- Viterbi
- BCJR
- Reed Solomon

Modulation
- WFM/NBFM
- AM/PM/SSB
- FSK/PSK/QAM
- GMSK/VSB-8/OFDM

Math
- Add
- Subtract
- Multiply
- Divide
- Log

Type Conversions
- Complex <>IntShort/Real/Imag
- Complex<>Mag/Arg
- Float<>Complex/Char/UChar
- Packed<>Unpacked
- Symbols<>Chunks
- Vector<>Stream<>Streams
- Interleaver<>Deinterleaver
- Complex Conjugate

Miscelaneous
- M&M Clock Recovery
- AGC
- PLL
- Costas Loop
- Adaptive Equalizer

Figure 4 The GNU Radio Modules.

- PN sequences based sounder (spread spectrum sliding correlator sounder).
- Direct RF (radio frequency) pulse system
- Frequency domain channel sounder

The PN sequence based sounder was the one used for this research and shall therefore be briefly explained. It relies on the properties of PN sequences, which are periodic sequences whose spectrum closely resemble random binary sequence spectrum. They are usually produced from feedback registers. The maximum period of the PN sequence L, is given by,

$$L = 2^m - 1, \tag{1}$$

where m is the length of the feedback shift register. The period of the sequence waveform T_b, is given by

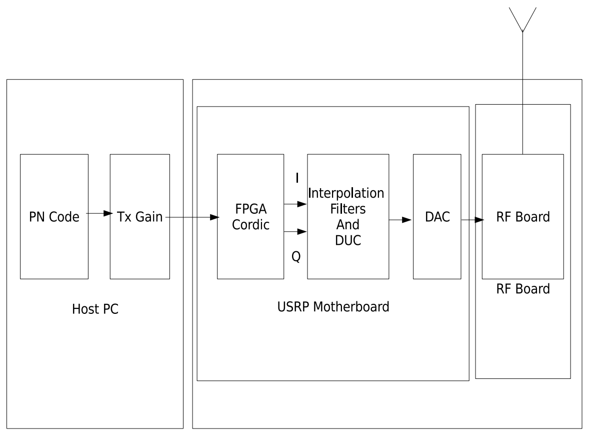
$$T_b = LT_c, \tag{2}$$

where T_c denotes the chip period long. The autocorrelation function is defined as

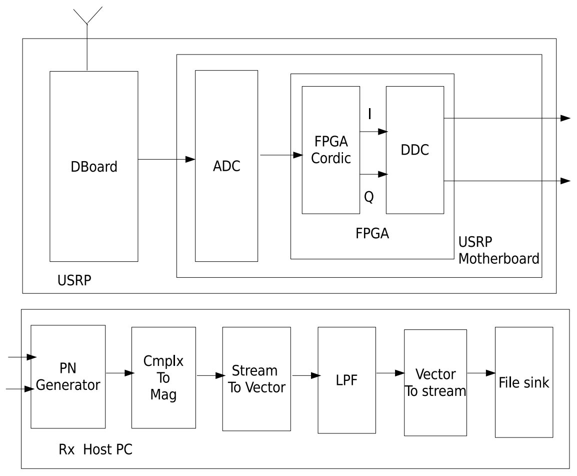
$$R_c(\tau) = \frac{1}{T_b}\int_{-T_b}^{T_b} s(t)s(t-\tau)dt. \tag{3}$$

The autocorrelation function of the PN sequence is given by

$$R_c(\tau) = \left\{ \begin{array}{ll} 1 - \frac{L+1}{LT_c} & |\tau| \leqq T_c \\ -\frac{1}{L} & \text{(otherwise)} \end{array} \right. . \tag{4}$$



(a) The transmitter



(b) The receiver

Figure 5 The block diagrams of the transmitter and receiver

Therefore considering the received signal *r(t)* is related to transmitted signal *s(t)* and channel response *h(t)* as follows:

$$r(t) = s(t) * h(t), \tag{5}$$

where * denotes convolution operation. If the PSD(Power Spectral Density) is considered flat for the frequency range of channel considered then

$$P_s = |S(f)|^2. \tag{6}$$

We can therefore estimate *h(t)* from the relation as

$$\hat{h}(t) = \frac{1}{P_s}r(t) * s(-t) = \frac{1}{P_s}s(t) * h(t) * s(-t). \tag{7}$$

Eq.(7) can be re-written by

$$\hat{H}(f) = \frac{1}{P_s}S(f)H(f)S^*(f) = \frac{1}{P_s}|S(f)|^2.H(f). \tag{8}$$

Therefore $\hat{H}(f) = H(f)$.

Two USRPs, each connected to a host PC were configured into master transmitter and slave receiver as in Fig. 5(a) and Fig.5(b) respectively. A detailed picture of the signal flow within transmitter and receiver signal processing blocks can be understood from these diagrams. In this system, synchronization of the carrier frequency was obtained by sharing the

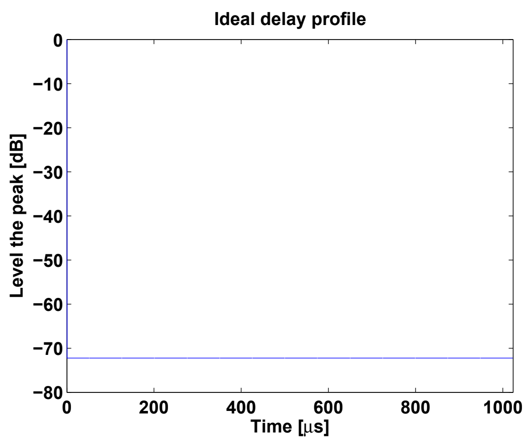


Figure 6 The delay profile loopback output

oscillator via a cable connection from the master transmitter to the slave receiver. The transmitter host generates a PN sequence which BPSK modulates a carrier for transmission. The slave receiver then cross-correlates the received signal with a locally generated copy of the PN sequence. The resulting cross-correlation signal is then interpreted as the channel impulse response (CIR).

3. Channel Sounding Demonstration

In order to verify the operation of the sounding system in particular, and the USRP in general, the output from various stages were obtained and analyzed against the expected results. The PN sequence data which goes into the transmitter USRP was lagged on to a file and the extracted result analyzed in Matlab. Similarly the PN output sequence from the receiver USRP before being cross correlated with a locally generated PN sequence was also extracted and analyzed in Matlab. This process confirmed the data input and output at various stages of the sounding system. After this process was completed the signal from the transmitter was looped back into the transmitter and correlated with its copy in oder to verify the system. The obtained results are as in Fig. 6

Table 2 shows the GNU Radio sounder specifications while table 3 shows the adopted measurement parameters the channels sonding experiments done. Figures 7(a) and 7(b) show the obtained delay profile and the impulse response respectively, under LOS conditions. Figures 8(a) and 8(b) show the obtained delay profile and the impulse response respectively, under NLOS conditions.

4. Disscussion

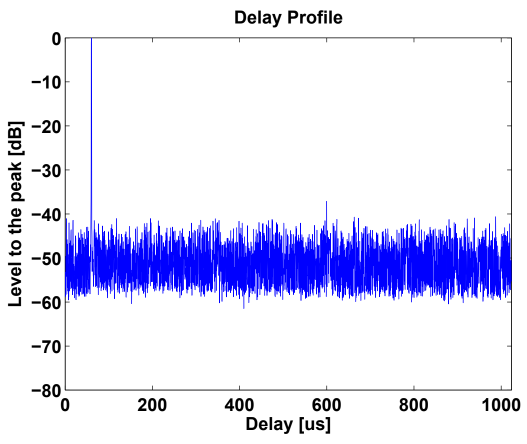
From the measurements at least three significant MPCs (Multi-Path components) were observed. However a number of limitations were noted. Frequency synchronization was achieved by using a cable to supply the slave USRP with the clocking signal from the master USRP. Needless to say, it

Table 2 Sounder specifications.

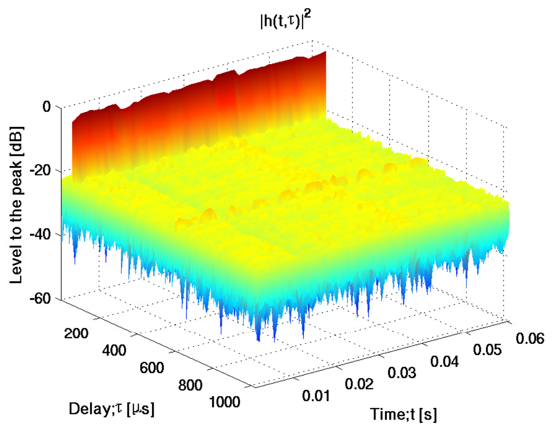
Frequency	2.4-2.5 GHz, 4.9-5.9 GHz
PN Sequence Chip rate	4 ∼ 32 M chips/s
PN Sequence Length (L)	3 ∼ 4095 chips
Receiver Gain	0 ∼ 92 dB
Resolution	31.25 ∼ 250 ns
Transmitted Power	-2.75 dBm
ADC sample rate	64 MSa/s

Table 3 Experiment parameters.

Frequency	5.2GHz
PN Sequence Chip rate	4 M chips/s
PN sequence length (L)	4095 (=$2^{12} - 1$) chips
Receiver gain	46 dB
Delay resolution	250 ns
Transmitted power	-2.75 dBm
ADC sample rate	64 MSa/s
USRP Decimation rate	16



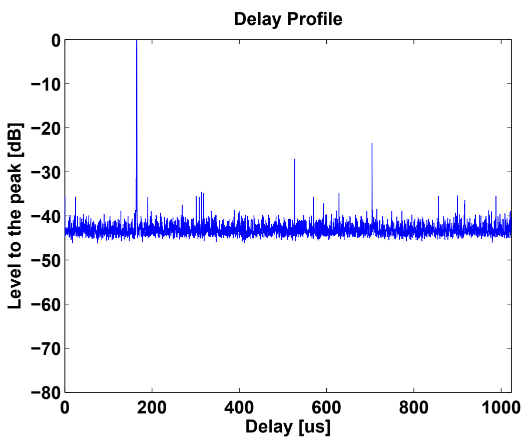
(a) Delay profile.



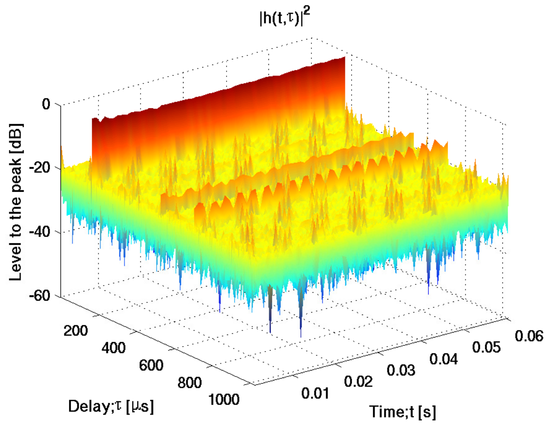
(b) Impulse response.

Figure 7 LOS delay profile and impulse response

is only applicable for use in indoor channel measurements. The transmission bandwidth can potentially be increased to 32MHz but at at present only up to 4MHz has been success-



(a) Delay profile.



(b) Impulse response.

Figure 8 NLOS delay profile and impulse response

fully used. The USB speed was one of the main limitations to the achievable resolution for this sounding system. The FPGA buffer for both transmit and receive directions was only 2k words, a further handicap for the real time demands of channel sounding.

5. Conclussion

From the results and discussion of this sounding system, it appears the GNU Radio is usable for channel sounding but with some limitations as far as the resolution is concerned. The main limitation appears to be the USB bandwith. A solution that can avoid this USB bottleneck by processing the correlation within the USRP's FPGA could implement a 32 Mb/s chip rate, which is the maximum that to which the ADC can digitize a signal to. It is hoped that soon the CIR measurements shall be done using an implementation which generates the PN code in the FPGA and avoid the usb bottleneck.

References

[1] S. Haykin, Communication Systems 4th Edition,2001,Spread Spectrum modulation, pp 479–509.

[2] E. Blossom, `http://www.gnu.org/software/gnuradio/doc/howto-write-a-block.html`

[3] `http://www.gnuradio.org/trac`

[4] `http://www.gnuradio.org/images/`

[5] `http://www.propagation.gatech.edu/Archive/PG_TR_050515_RJP/PG_TR_050515_RJP.pdf`

[6] A. Christopher,Design and Implementation of an Ultra-broadband Millimeter-Wavelength Vector sliding Correlator Sounder and In Building Multipath measurements at 2.5 and 60 GHz, 2002, `http://scholar.lib.vt.edu/theses/available/etd-05092002-101656/unrestricted/AndersonThesisETD.pdf`