FROM SIMULATION TO DEMONSTRATION – A SDR-BASED MULTI-MODE TESTBED

Lin HUANG; Kan ZHENG; Guillaume DECARREAU (Orange Lab, Beijing, China; {lin.huang, kan.zheng, guillaume.decarreau}@orange-ftgroup.com)
Hanwen CAO; Gang LI, Zhangchao MA, Zhi YAN, Huangcheng ZENG
(Beijing Univ. of Posts and Telecom, Beijing, China,{hanwen.cao, ligangcool, mzcroy, yanzhibupt, zenghuacheng}@gmail.com)

ABSTRACT

This paper presents a SDR-based multi-mode testbed being developed by Orange Lab Beijing and BUPT, which consists of the Universal Software Radio Peripheral (USRP) boards and the PC-cluster. GNU Radio, C++ / IT++ programming and MPI parallel computing framework are used to make the testbed easy to be developed. So the fast prototyping is possible. The purpose of this testbed is to validate the algorithm performance in realistic channels and in the presence of implementation impairments. Now the TD-SCDMA 64 kbps / 384 kbps links and a 2x2 MIMO-OFDM system are already built in this multi-mode PC-Cluster SDR platform. The system performance, SNR, BLER, MIMO channel fading and correlation matrix etc can be shown in graphic monitor windows in real-time.

Keywords- SDR; testbed; PC-cluster; GNU Radio

1. INTRODUCTION

Most of the wireless system research uses the simulation as an important tool to validate the system performance. However, it cannot replace the test in real wireless environment and sometimes it is even problematic. Therefore, the motivation of our work is to build a flexible testbed for evaluating the novel algorithms under wireless transmission environment and accelerate the transition from simulation to demonstration.

GNU Radio [1] is an open source framework which allows the construction of radios where the actual waveforms transmitted and received are defined by software. The Universal Software Radio Peripheral (USRP) from Ettus Research [2] is a device connected to computer with USB connection, to do RF / baseband conversion. It works with GNU Radio to create the software defined radios. Now this combination attracted the growing users from various research groups. In our testbed, the USRP plus GNU Radio are adopted as the RF front-end. Most of the baseband processing is done in the PC-cluster.

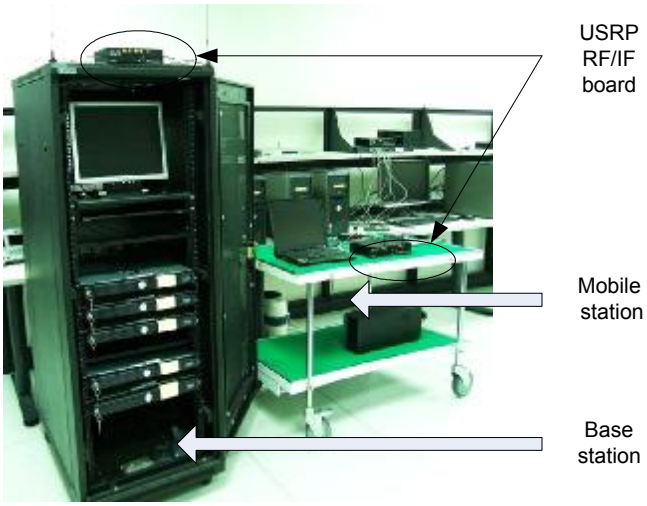

Figure 1 Testbed overview

To support real-time processing for the complex physical layer, PC-cluster is used for the parallel computing. GNU Radio sends/receives data to/from the PC-cluster with TCP connection. Almost all of the baseband signal processing modules are done in the PC-cluster parallel computing environment, including the frequency synchronization, channel estimation, modulation / demodulation, channel coding / decoding and so on. The parallel computing environment is built in the MPI (Message Passing Interface) framework. Several computers work together to meet the real-time requirement of intensive data processing. Some parallel scheduling strategies are used in the cluster and can be selected by the configuration interface. The simulation source codes based on IT++ library are very easy to be ported into the MPI framework. It makes the moving from the simulation to the demonstration fast and smooth.

Multiple wireless systems including TD-SCDMA and 3GPP LTE-like PHY layer have been implemented in this testbed with the necessary simplification due to the limitation of USRP. The video streaming application can be demonstrated on the different PHY layers. The experiments under the indoor environment and the short-distance outdoor

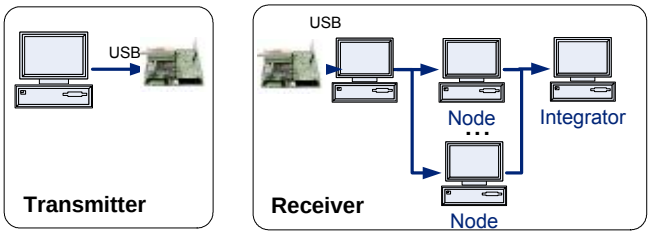

Figure 2 System architecture

environment are performed with good results. In addition, the graphic user interfaces (GUI) of SNR, BLER, the constellation and the channel correlation matrix are well-designed to monitor the system performance.

In the following section, the system architecture will be firstly introduced. Then the implementation details are discussed in the third part. An example of MIMO-OFDM system is shown in the forth part. Finally, the conclusion will be given.

2. SYSTEM ARCHITECHTURE

The system architecture is shown in Fig. 2. At the transmitter side, the source data stream is produced by the transmitter computer, then this baseband stream is sent to the USRP board and converted to RF signal to be transmitted. The USRP front-end is configured to support 2 x 2 antennas. So two receive antennas capture the RF signal from air then the USRP board converts them into baseband and sends them to the PC-cluster. In the PC-cluster, a node called 'Master' distributes processing tasks to several "Slaves" who are assigned to the processing jobs. The processed results generated by Slaves are then aggregated by the 'Integrator' node to recover the data packets.

All the baseband processing is implemented in software on the general computer by C++ programming. Although a FPGA-DSP testbed can provide higher throughput and shorter latency, using high level programming language on common Linux system makes people with little or no 'hardware' background implement the platform very easy. The software tools used in the development include C++, Python, IT++ library and MPI. Fig. 3 gives the block diagram of different software modules, their usage and the interface with each other. Here one process or thread, i.e. one software module, can be assigned to any node (computer) according to the parallel strategy. In the part 3, the details of different block will be introduced.

When a specific algorithm is created, to be implemented in the testbed, the steps include

- Matlab simulation
- IT++ simulation (C++)
- Embed the C++ modules into MPI framework

So we could see that, this is a very easy algorithm transfer from simulation to demonstration.

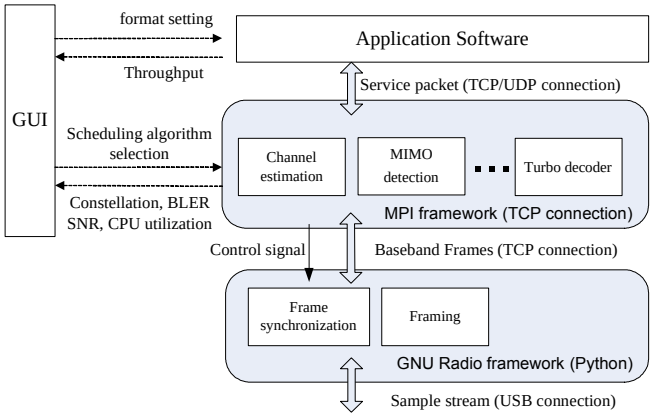

Figure 3 Block diagram of software modules

3. SYSTEM IMPLEMENTATION

In this section, the details of different parts in the testbed are introduced.

3.1 RF/IF Front-End

The USRP is adopted as the RF front-end for our SDR platform. It is designed to allow general purpose computers to produce radio signal. In essence, it is a RF and IF converter which sends or receives I/Q baseband signal to/from the host by the USB 2.0 cable. The USRP product family includes the motherboard, which contains an FPGA and several AD/DA converters, and changeable plug-in daughterboards that cover different frequency ranges. With the various daughterboards, the USRP has an overall range from DC to 2.9 GHz, which covers everything from AM radio to Wi-Fi and beyond.

The chief limitation of USRP is its bandwidth. USB connection is chosen as the interface between the board and the host. Now the maximum throughput on the USB cable is 32MB/s. Considering the 2 I/Q channels and 2 bytes for one sample, the available bandwidth is

32MB/s / 2channels / 2B/Sample = 8 MSample/s

In addition, the oversampling is needed, e.g. x2 oversampling. The 2-antenna MIMO transmission also doubles the bandwidth requirement. So finally if x2 oversampling and 2x2 MIMO are used, the supported bandwidth will decrease to

8MS/s / 2 / 2 = 2 MS/s

Therefore, the bandwidth that the USRP can provide is very limited. This is a big constraint for high data rate system design.

Together with USRP, GNU Radio is an open-source software for SDR development. GNU Radio has libraries for common software radio needs, including various filters, modulators, FFTs, timing recovery etc. It is a very flexible framework, and it allows programming on C++ and Python.

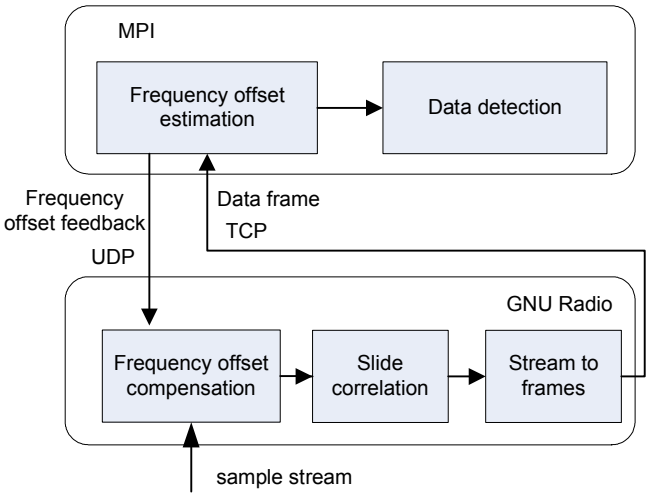

Figure 4 Frequency synchronization and information exchange between MPI and GNU Radio

In our architecture, the USRP is connected to a computer which is called 'frontend PC', where GNU Radio software runs to make USRP configurations and undertake some fundamental processing jobs, such as framing and synchronizations. Besides, the network interface is necessary for the frontend PC to enable it to exchange data with PC cluster.

3.2 The PC-Cluster

In our testbed, general purpose processors are used instead of dedicated DSP processors for its flexibility and low-cost. So here every computer in the cluster is looked as a data processing unit that processes the data stream in a parallel computing way. These processing units are called 'nodes' in this system and connected together with the Gigabit Ethernet. Every node has its IP address and the subnet IP address is used as a broadcast address to send the control information to all the nodes. There is a 'control point' assigned to help the communication among the nodes.

TCP connection is used between the front-end PC and the PC cluster, and also used in MPI among the computers in cluster. The TCP communication function is integrated into a C++ class which is convenient to be called.

3.3 The Physical Layer

Different physical layers can be implemented in this testbed. We have realized TD-SCDMA 64 kbps & 384 kbps links and simplified 2x2 MIMO 1.25 MHz BW LTE link. They have quite different physical layer design. But some common modules can be reused, such as modulation / demodulation, convolutional coding and turbo coding.

The testbed uses GNU Radio on the 'front-end' computer. Here, GNU Radio handles all hardware interfacing, multithreading, thus it frees us to focus on the signal processing. Fig.4 illustrates how the GNU Radio collaborates with the MPI side and exchange information. A slide correlation block is implemented in the front-end computer to find the pilot correlation peak and divide the continuous sample stream into frames. There is a frequency offset compensation block in front of the slide correlation, which receives the results of frequency offset estimation from PC-cluster periodically and adjust the phase of incoming signal stream. So this is a close-loop adjustment procedure which will quickly converge to accurately synchronized status.

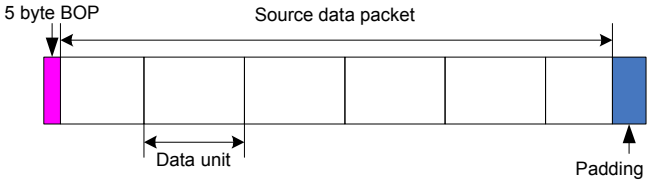

Figure 5 Packetization from MAC layer to PHY layer

Except the frame synchronization, most of baseband signal processing is implemented in the MPI framework. Different system consists of different modules and has different complexity. We measure the time each system spends on handling one frame, then multiply it by 3.0 GHz, the processing frequency of CPU. This gives a rough measure of their complexities, although it will be different when the CPU structure or the manufacture is different. So in fact, here the complexity is not absolute but relative. We list this relative complexity requirement in the following.

Table I Signal processing complexity

Completely implemented TD-SCDMA 64 kbps link	2.2 GCycles/s
Completely implemented TD-SCDMA 384 kbps link	11.3 GCycles/s
Simplified LTE-like MIMO-OFDM 770 kbps link	8.8 GCycles/s

According to this table, the number of computers that the system needs can be roughly estimated. Suppose all the computers are equipped by 3.0 GHz CPU, then a 64 kbps link requires one CPU; a 384 kbps link needs 4 CPUs; while the MIMO-OFDM link can be real-time processed by 3 CPUs. The real-time experiments prove this estimation is correct.

.

3.3 The Simplified MAC Layer

Above the physical layer, data packets are re-segmented and some additional information is added to these packets. This procedure is called packetization. For example, at our simplified MAC layer, data packets from upper layer (IP packets) are split into smaller units then the header is added. The packetizing procedure can be simplified because the

Table II MIMO-OFDM system parameters

System Parameters	MIMO-OFDM
Sampling rate	2 MS/s, 1 MS/s
Carrier frequency	2.45 GHz
FFT size	128
Number of occupied sub-carriers / valid sub-carriers	80
Symbols per subframe	7
CP length	14 for the first 6 symbols, 20 for the last symbol
Subframe length	1000 chips
Modulation	QPSK, 16QAM
Channel Coding	Convolutional code
Antenna Configuration	2x2
MIMO Scheme	BLAST

scenario in our experiment is simple, only one way and single user.

At the transmitter side, the PHY layer receives the UDP packet from upper layer, then the packet will be divided into smaller data units which is the payload of one radio frame or TTI etc. These data units are then channel coded, modulated and put into radio frames which are transmitted into air. At the receiver side, an inverse process is done to recover the source data. Thus, after the signal is decoded into data units, they should be combined into UDP packets and delivered to the data sink.

To facilitate the data unit combination, the BOP (Begin of Packet) tag for a source packet is added at the beginning of UDP packet as shown in Fig.5. The format of BOP tag is a 3-byte string followed by 2-byte packet size (unsigned short). The maximum packet size is 64K, which is the same as the maximum size of an IP packet.

When the size of source packet is not multiple of the size of data unit, the padding bits will be inserted at the tail. Although some of the payload is wasted, this part is much smaller, just dozens or hundreds of bytes, than the size of source packet which is usually large up to several Kbytes. Thus the overhead is acceptable.

4. MIMO-OFDM EXAMPLE

In this section, a MIMO-OFDM system implemented on this testbed is introduced as an example. This system design refers to the 3GPP LTE (Long Term Evolution) draft. The frame structure, bandwidth, number of subcarriers are similar with LTE's. Table II gives some key system parameters.

The subframe totally consists of seven OFDM symbols. The guard interval (GI) is added in front of every OFDM symbol to mitigate the effects of inter symbol interference (ISI) caused by channel time spread. Among one subframe, the first symbol is the reference signal for channel estimation. The SCH (Synchronization Channel) symbol is inserted at

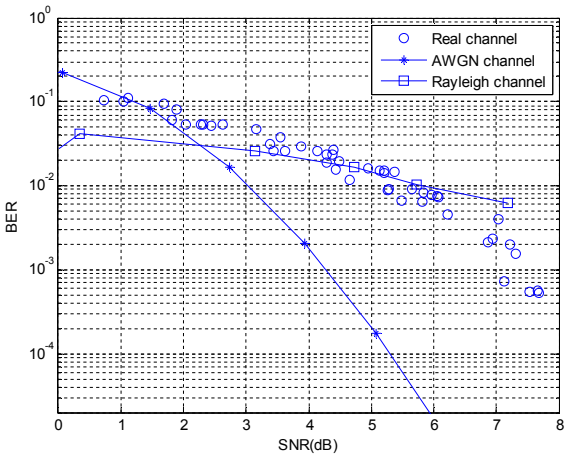

Figure 6 BER performance comparison between simulation and real channel experiment

the end of subframe for the initial frequency and timing synchronization. In the current testbed, the SCH symbol is inserted in every subframe, but we will assign only one SCH to every 4 subframes for less overhead. The Zadoff-Chu sequence is chosen for the SCH symbol, which has the immunity to large frequency offset.

Each antenna transmits its own reference signal, a PN sequence. The reference signals from different antennas are subcarrier-interleaved. That means they are frequency division. For each antenna, the PN sequence is differential coded to facilitate the integral frequency offset estimation.

The current version of USRP cannot support synchronously transmitting on two motherboards. So the basic 2x2 MIMO is configured now in our platform. This is a classical 2x2 BLAST spatial multiplexing. The information data is divided into two streams, then transmitted from two separated antennas. At the receiver side, ZF (Zero Forcing) and ML (Maximum Likelihood) detections are realized and tested.

The channel coding in this system is convolutinal code. The generator polynomials is selected as g(0)=0133, g(1)=0171 for the constraint length of 7 and the code rate of 1/2. In order to make the system more robust in severe wireless channel, constraint length of 9 and turbo coding may be implemented as our alternative in future. Furthermore, due to the high complexity of the viterbi decoding, we chose a SIMD (Single Instruction Multiple Data) optimized decoder from Phil Karn to accelerate the processing speed [3]. The Karn's viterbi decoder can reach up to 22 Mbps information bit rate for 1/2 rate and the constraint length of 7 on the 3.0 GHz CPU. When the constraint length is 9, the maximum throughput is 4 Mbps. This decoder greatly reduces the program execution time. However, the BER performance of 1/2 hard decision convolution code is not good enough. A high speed turbo codec is really appreciated. It's under further investigation.

Fig.6 shows the BER performance comparison between simulation and the real channel experiment. The simulation results are gotten under AWGN channel and Rayleigh channel. The measurement results of real channel are produced by the SNR estimation block and the error bit counting block. From this figure we chould see that the BER curve of real channel reflects its Rayleigh fading property.

To make the system performance visible, we developed many GUIs to monitor the real time status. Fig. 7-9 show the received constellation, the MIMO channel estimation and the SNR estimation respectively. These GUIs provide great help in the various experiments. Moreover, many parameter configurations and mode selections also can be made through the GUIs.

5. CONCLUSION

This paper introduces the SDR-based multi-mode testbed built on USRP and PC-cluster. GNU Radio, IT++ and MPI are used to facilitate the transition from simulation to demonstration. The algorithm evaluation in real wireless environment becomes easy since the time cost is greatly reduced. This is very useful especially for the current more and more complex wireless system.

The TD-SCDMA 64 kbps / 384 Kbps links and a 2*2 MIMO-OFDM subsystem are built in the same multi-mode testbed. The MIMO-OFDM system is introduced more detailed in this paper. The experiment results in real channel are also presented. The system performance, SNR, BLER, MIMO channel fading and correlation matrix etc. are shown in graphic monitor windows.

More experiments are currently being done such as channel correlation measurement and outdoor environment trial. We also plan to make some investigation and development on cooperative transmission or cross layer optimization in future.

6. REFERENCES

[1] www.gnu.org/software/gnuradio
[2] www.ettus.com
[3] http://www.ka9q.net/code/fec/simd-viterbi-2.0.3.tar.gz
[4] K. Mandke, S. Choi, G. Kim, R. Grant, R. C. Daniels, W. Kim, R. W. Heath, Jr., and S. Nettles, "Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed," in the Proc. of IEEE Vehicular Tech. Conf. , Dublin, Ireland, April 23 - 25, 2007.
[5] http://nms.csail.mit.edu/SpectrumWare/
[6] Sujian Zhao; Cong Shen; Xin Su; Yan Yao; "Architecture of a softwareradio system based on cluster of workstations". IEEE TENCON 2003. vol.4, pp.1439 - 1444, 15-17 Oct. 2003
[7] Lin Huang; Kan Zheng; Xiaoyu Wang; Decarreau, G."Timing Performance Analysis in an Open Software Radio System", Communications and Networking in China, ChinaCom '06, Beijing, China, 25-27 Oct. 2006 Page(s):1 – 5
[8] Huang, Lin; Cao, Hanwen; Zheng, Kan; Decarreau, Guillaume;"Practical Frequency Synchronization Scheme in an Open Software Radio TD-SCDMA System", Wireless and Optical Communications Networks, WOCN '07, Singapore, 2-4 July 2007

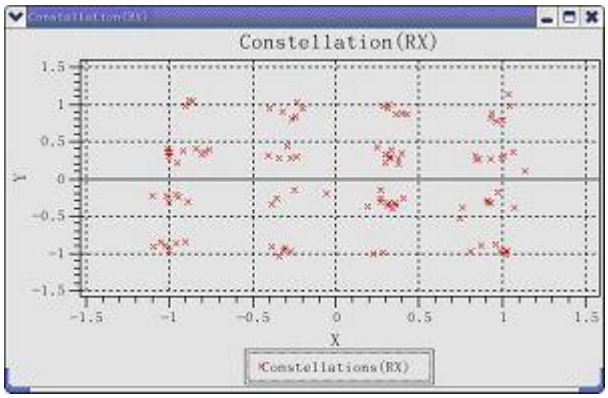

Figure 7 Received 16QAM constellation

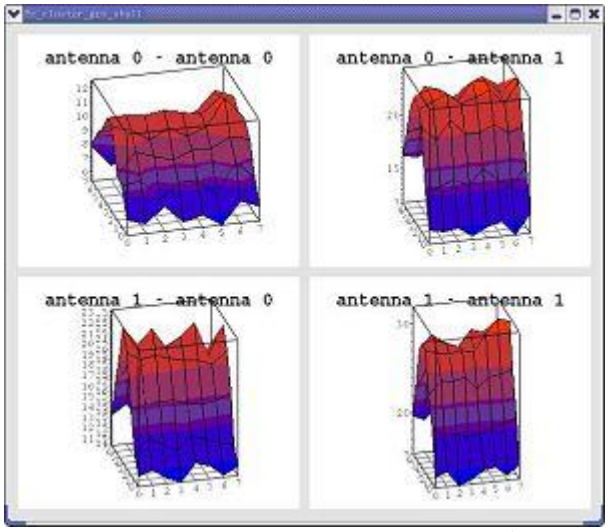

Figure 8 MIMO channel estimation, displayed in both frequency domain and time domain

Figure 9 SNR estimation, the average SNR in experiment is around 14 dB