A General Purpose Processor Based IEEE802.11a Compatible OFDM Receiver Design

Muhammad Khurram (Lecturer, Research Scholar, Dept of Computer & Information Systems Engg, NED University, Karachi, Pakistan; mkhurrum@neduet.edu.pk);
Prof. Dr. Shahid Hafeez Mirza (Dean(ECE), Faculty of Electrical & Computer Engg, NED University, Karachi, Pakistan; deanece@neduet.edu.pk)

Abstract - Using the processing power of multi-gigahertz general purpose processors (GPP) to perform radio functions can be a better and economical option to design a software defined radio (SDR) system. An efficient SDR system with multiple protocol support can be designed by identifying different blocks in the channel processing stream of different wireless protocols that can be mapped on GPP and field programmable gate arrays (FPGAs) processing platforms depending on throughput requirements of the corresponding protocol. This paper presents the ongoing research work in designing a novel architecture to prototype and develop efficient SDR systems using GPP as main digital signal processing (DSP) platform along with FPGAs to perform real-time signal processing tasks that can not be handled by GPPs. In this research project, a software defined radio is designed for the physical layer of WLAN standard IEEE 802.11a receiver. Different sub-systems of the channel processing stream of IEEE 802.11a OFDM receiver are mapped on GPP of a PC and a PCI board containing fast ADCs to receive the received analog signal from the RF front-end. The software radio architecture discussed in this paper is a scaled down version of the software radio that has to be developed as the research project for M.Engg. by research at NED University by the principal author.

I. INTRODUCTION

There has been a great advancement of wireless devices towards SDR systems that perform digitization of RF signals as close to the antenna as possible. Signal processing of baseband and IF signals digitally in software offers reprgrammability, flexibility and greatly reduces cost. Several commercial and open-source software projects are under progress to develop SDR systems that run on both dedicated DSPs and GPPs [1][2]. Dedicated DSPs have been used till now for implementing SDRs, which provide real-time performance with less power consumption but do not provide the flexibility required for future networks [3]. The software development languages for DSPs are very cumbersome to use and are in their infancy as they do not provide a suitable abstraction to underlying hardware [4]. This greatly increases development, testing and deployment time. In contrast, GPP technology has made tremendous advances in the recent past along with the rapid development of software languages. The flexibility and development tools of GPPs greatly outweigh the DSP technology.

In this paper, a novel architecture is described to process the radio signal using general purpose processing platform and operating system (OS). The designer can implement the software radio components in familiar high level languages like C/C++. The designer can use common approaches of object oriented programming and debugging to design real-time SDR systems. Use of general purpose processor loaded with general purpose OS and signal processing application software greatly reduces design time and increases flexibility.

The system, described in this paper, focuses on the implementation of IEEE 802.11a WLAN OFDM receiver. Different subsystems of OFDM receiver are mapped onto GPP and FPGA depending on the processing throughput requirements to ensure real-time signal processing. The high density FPGA is mounted on the SDR front-end PCI board containing fast broadband ADCs/DACs and provides reconfigurable logic of sampling and FIFO buffering.

II. SYSTEM ARHITECTURE

The system architecture presented in this paper is shown in figure-1. Which is to first downconvert a wideband of the spectrum to an IF frequency and then digitize it using wideband A/D converter. It is important that the downconverted band is not just singal band but a wide band containing many norrow band signals. The RF signal can also be fed to the A/D converter if it is amplified and band pass filtered before feeding it into A/D converter. In this case, bandpass sampling technique is used so that the wideband signal is automatically downconverted to IF. Further Processing is done in digital domain. If appropriate sampling frequency is selected, the whole band can directly be converted down to baseband. Demultiplexing and baseband signal processing is done completely in software. The FPGA shown in figure-1 is used for programmable functions of sampling frequency selection, FIFO buffering and optional signal up/down conversion. All these parameters are software selectable and modifiable. Other communications functions are performed by GPP programmed in software.

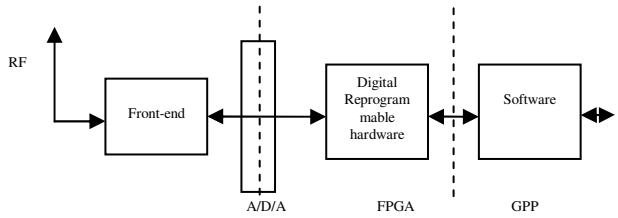

Figure-1

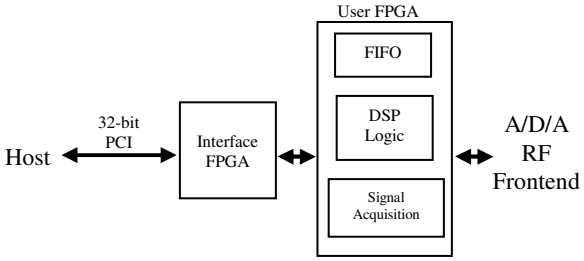

Figure-2

A. Signal Acquisition Front-end

The primary bottleneck to system performance is the I/O throughput to the application. Existing workstation architectures can not provide data at a rate sufficient to enable software radio applications. To digitize wideband signals of 20 MHz or 40 MHz wide, very high sampling rate wideband ADCs/DACs are required and also for this purpose, high throughput interface is necessary to transfer digital samples to/from GPP in real-time. For data conversion, Analog Devices ADC is used for A/D conversion which supports 14-bit quantization at 64 MSPS. For D/A conversion, onboard DAC provides 14-bit resolution at up to 160 MSPS. These ADCs and DACs are interfaced with high density Vertex-II FPGA. The onboard Vertex-II user FPGA is programmed in VerilogHDL/VHDL to provide signal acquisition, buffering and processing function.

For transmit and receive signal buffering, FIFO memories are programmed in user FPGA and are accessed by the GPP through PCI interface.

B. I/O System

The signal acquisition front-end mounted on a PCI board that is interfaced with the GPP through general purpose PCI interface. The interface FPGA, built-in on PCI board, is used to provide fast 32-bit PCI interface between the PCI bus of the host PC and the user FPGA. This interface, named as FIFO-PCI, supports both the programmed I/O and multiple DMA channels between FIFO memories programmed in user FPGA and the host PC. The block diagram of this interface is shown in figure-2.

C. Reconfigurable Hardware Accelerators

The key to increase processing capacity and still maintain flexibility is to introduce *accelerators* in the processor. An accelerator is extra hardware added to a programmable processor which performs a certain pre-configured task while the processor is free to perform other operations. However, every extra accelerated function will increase the hardware cost, so selecting the right accelerators to cover most processing needs over multiple standards is essential [5].

III. DEALING WITH VARIALBILITY

Using GPP platform with general purpose OS, brings in tremendous flexibility and ease of design but it also posses challenges to the implementation of real-time signal processing applications. The most important challenge is the variability of the execution time. In case of conventional DSPs, the execution time can be predicted but in GPP based systems with variable execution environment the real-time environment can only be bounded by the worst case performance.

To absorb the jitter caused by variable execution times, FIFO memories are programmed in user FPGA as described in section 2.2. Multiple FIFOs are written/read using DMA in both transmit and receive chains to output/input samples from data acquisition front-end in real-time.

A. Software Architecture

In GPP systems, abstraction layers of OS cloud the ability to be fully deterministic about code execution time. Also, multiple layers of caches, virtual memory implementation, multi-tasking and competition for I/O and memory buses add jitter to the expected time required for a sample to travel from signal acquisition front-end and GPP. A hard real-time system imposes a hard deadline for each task, and provides a mechanism that the deadline is met. The software architecture presented in this paper is a soft real-time system. A soft real-time application has timing constraints but there is no guarantee that they will be met [6].

In the software architecture described in this paper and shown in figure-3, DATA-PULL model is employed. In the data-pull model, the execution is driven by the data sink which requests the data, as it is needed by the upstream modules or layers. The signal processing software is a multithreaded application programmed in C++ and divided into different sub modules depending upon their data requirements. Each submodule or layer is executed independently in an independent thread. The data to be processed in these layers are kept in main memory queues present between two subsequent layers and fed to the layer upon its request.

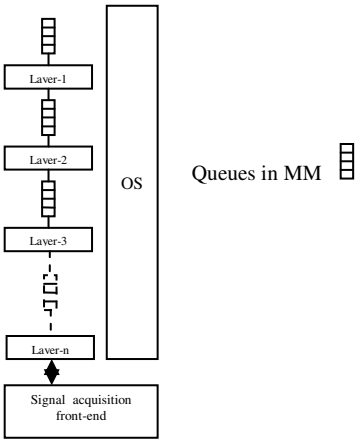

Figure-3

IV. IEEE802.11A RECEIVER

IEEE802.11a is a wireless local area network (WLAN) access technology and is similar to HiperLAN/2 WLAN standard. It operates in the 5 GHz frequency band and makes use of orthogonal frequency division multiplexing (OFDM) to transmit the radio signal. The bit rate of IEEE802.11a at physical layer depends on the modulation type and is 6, 9, 12, 18, 24, 36, 48 or 54 MBits/sec.

The basic idea of OFDM is to transmit high data rate information by dividing the data into several parallel bit streams, and let each one of these one of these bit streams to modulate a separate carrier [7]. An IEEE802.11a OFDM channel contains 64 subcarriers and has a channel spacing of 20 MHz. 48 out of 64 subcarriers carry actual data, 4 are pilot subcarriers and rest of 12 carriers are NULL. Important parameters of IEEE802.11a OFDM standard is given in table-1 and table-2 [8].

In practice, the OFDM signal for the standard IEEE 802.11a is generated as follows:

At the transmitter, binary input data is first scrambled using 127-bit length scrambling sequence, generated by the polynomial S(x) shown below:

$$S(x) = x^7 + x^4 + 1$$

After scrambling, data is encoded by a rate ½ convolutional encoder. The rate can be increased to 2/3 and ¾ using appropriate puncturing algorithms. The generator polynomial for convolutional encoding is shown below:

$$g_0 = 133_8 \text{ and } g_1 = 171_8$$

To minimize burst errors, interleaving is applied and binary values are then mapped onto the constellation depending on the type of modulation technique used. In 802.11a, BPSK, QPSK, 16-QAM and 64-QAM are provided as constellation mapping techniques to support different data rates. Four pilot values are added with each 48 data values, resulting in a total of 52 complex values per OFDM symbol. The symbol is modulated onto 52 subcarriers and 12 null subcarriers by

Data rate (Mbits/s)	Modulation Type	Coding rate (R)	Coded bits per subcarrier (N_{BPSC})	Coded bits per symbol (Nc_{BPS})	Data bits per symbol (N_{DBPS})
6	BPSK	½	1	48	24
9	BPSK	¾	1	48	36
12	QPSK	½	2	96	48
18	QPSK	¾	2	96	72
24	16-QAM	½	4	192	96
36	16-QAM	¾	4	192	144
48	64-QAM	2/3	6	288	192
54	64-QAM	¾	6	288	216

Table-1

Parameter	Value
N_{SD} : Number of data subcarriers	48
N_{SP} : Number of pilot subcarriers	4
N_S : Number of subcarriers, total	52 ($N_{SD} + N_{SP}$)
Δ_F : Subcarrier frequency spacing	0.3125 MH_Z (=20 MH_Z /64)
T_{FFT} : IFFT/FFT period	3.2 µs ($1/\Delta_F$)
$T_{PREAMBLE}$: PCLP preamble duration	16 µs ($T_{SHORT} + T_{LONG}$)
T_{SIGNAL} : Duration of the SIGNAL BPSK-OFDM symbol	4.0 µs ($T_{GI} + T_{FFT}$)
T_{GI} : GI duration	0.8 µs (T_{FFT} /4)
T_{GI2} : Training symbol GI duration	1.6 µs (T_{FFT} /2)
T_{SYM} : Symbol interval	4 µs ($T_{GI} + T_{FFT}$)
T_{SHORT} : Short training sequence duration	8 µs (10 x T_{FFT} /4)
T_{LONG} : Long training sequence duration	8 µs ($T_{GI2} + 2 \times T_{FFT}$)

Table-2

applying 64-point IFFT. The output is converted to serial and a 16-bit cyclic extension is added to make the system robust to multipath propagation. Windowing is applied after to get a narrower output spectrum. Using an IQ modulator, the signal is converted to analog, which is upconverted to the 5 GHz band, amplified, and transmitted through the antenna.

The receiver performs the reverse operations of the transmitter, with additional training tasks. In the first step, the receiver has to estimate frequency offset and symbol timing, using special training symbols in the preamble. After removing the cyclic extension, the signal can be applied to a

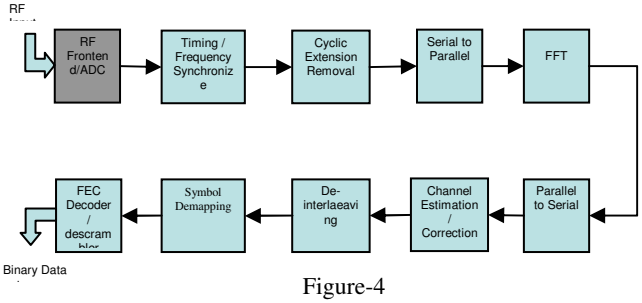

Figure-4

Fast Fourier transform block to recover the 52 information subcarriers. The training symbols and the pilot subcarriers are used to correct for the channel response as well as remaining phase drift. The constellation is then demapped into binary values, and finally a Viterbi decoder and descrambler decodes the information bits.

V. RECEIVER IMPLEMENTATION

The modules, mapped onto the general purpose processor and reconfigurable user FPGA, are sketched in figure-5. The complex input samples are downsampled, interpolated and then propagated to preamble detection, framing and frequency synchronization. This operation takes place in user FPGA as shown in figure-5. After frame detection and synchronization, complex samples are placed into receive FIFO buffers as described in section-2.2. When first FIFO buffer gets completely filled an interrupt is sent to the GPP to read the buffer using DMA. Further processing is done by the GPP programmed in C++. The C++ application is a multithreaded application as described in section-3.1.

A. Hardware Implementation

The PCI board structure is described in section-2.1. The user FPGA is programmed in verilogHDL using Xilinx System Generator software tool. Using this tool, the system has been tested using hardware co-simulation feature with Matlab. The user FPGA is programmed to provide A/D interface, FIFO-PCI and framing and synchronization modules. The FIFO-PCI I/O system full-duplex performance is shown in figure-6. This interface can support up to 512 MBPS in one direction; either transmit or receive. The performance of the interface is tested on Windows2000 platform. This can be increased if a less heavier OS, in terms of processing requirements, is used.

B. Software Implementation

An implementation of software architecture was developed on windows 2000 using C++ multithreading programming technique. While implementing different layers of software

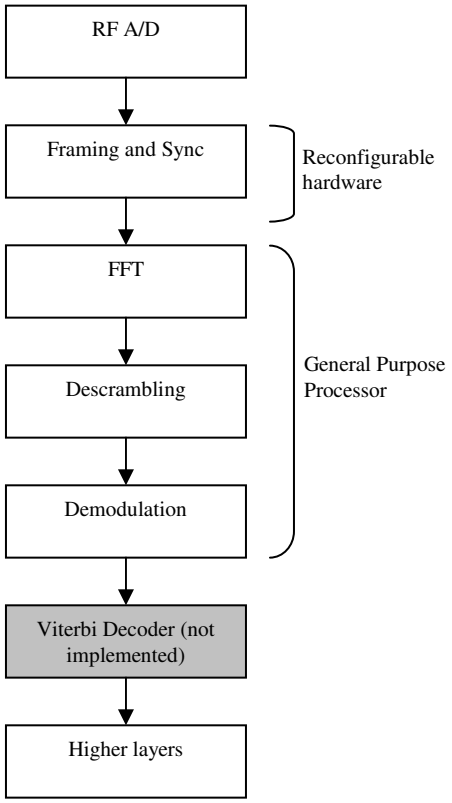

Figure-5

radio using GPP, as shown in figure-3, the designer can decide to implement a CPU bound synchronous layer or I/O bound asynchronous layer [3]. All the layers mapped on GPP are synchronous, only the DMA interface is I/O bound and a separate thread is instantiated to transfer data through PCI bus to main memory.

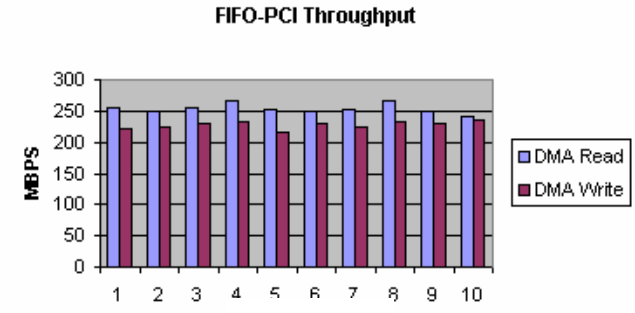

Figure-6

VI. CONCLUSION

A software defined radio system architecture using GPP as main signal processing platform is designed and tested by implementing IEEE 802.11a WLAN receiver. This paper has demonstrated the flexibility and capability of GPP based system to develop software radio solutions for next generation wireless protocols. Although, GPP based

workstations running on general purpose operating systems are high power consuming machines but they can provide the ultimate flexibility and processing capability for software defined solutions for those systems where power consumption is not a sensitive parameter to look for. Like, basestations, automobiles, stationary computer networks and many more. The ability of GPP based systems to upgrade via software only will open up many opportunities for next generation of wireless systems.

10. REFERENCES

[1] J. Chapin, V. Bose, "The Vanu Software Radio System", 2002 Software Defined Radio Technical Conference, San Diego, November 2002

[2] Peter G. Cook, Wayne Bonser, "Architectural Overview of SPEAKeasy System", IEEE Journal on Selected Areas in Communications, Vol 17, No. 4, April 1999

[3] P Mackenzie, L Doyle, KE Nolan, D O'Mahony, "an Architecture for the Development of Software Radios on General Purpose Processors", Proceedings of the Irish Signals and Systems Conference, ISSC June 2002

[4] Eyre, Jennifer, "The Digital Signal Processor Derby", IEEE Spectrum, Vol 38, No. 6, June 2001

[5] Anders Nilsson, Eric Tell, Dake Liu, "An Accelerator Architecture for Programmable Multi-Standard Baseband Processors", Proceedings of Wireless Networks and Emerging Technologies, WNET2004, July 2004

[6] Vanu Bose, Michael Ismert, Matt Welborn, and John Guttag, "Virtual Radios", IEEE Journal on Selected Areas in Communications, Vol. 17, No. 4, April 1999

[7] Yiyan Wu, William Y. Zou, "Orthogonal Frequency Division Multiplexing: A Multi-Carrier Modulation Scheme", IEEE Transactions on Consumer Electronics, Vol. 41, No. 3, August 1995

[8] IEEE Std 802.11a/D7.0-1999, "Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the GHz Band."

[9] Roel Schiphorst, 1Fokke Hoeksema, 2Vincent Arkesteijn, 1Kees Slump, 2Eric Klumperink and 2Bram Nauta, "A GPP Based Software Defined Radio Front-end for WLAN Standards", Preceedings of IEEE Benelux Signal Processing Symposium, 2004