OPEN WIRELESS SOFTWARE RADIO ON COMMON PC

Kan ZHENG  
France Telecom  
R&D Center (Beijing), China

Jianfeng WANG  
School of Telecommunication engineering  
Beijing University of Posts&Telecomms

Lin HUANG, Guillaume DECARREAU  
France Telecom  
R&D Center (Beijing), China

Abstract

Software radio is the promising technology that allows the different wireless standards easily be converged. Using general purpose processors and open-source operating systems instead of dedicated hardware and software to build the wireless communication system is very flexible and low-cost. In this article, the open-source platform based on common PCs is described, which allows rapid development and verification of software radio systems. We also discuss the efficient distributed strategies essential for this platform. Finally, the demonstration system of TD-SCDMA is developed and the conclusion given.

Keywords: software radio; PC; parallel computing

I. INTRODUCTION

Today the wireless communication systems are statically specified by their built-in link and physical layer functions. The innovation and technology tracking in wireless systems is slowed by the expensive hardware modification required to deploy different standards and services. The system infrastructure based on software radio technology can enable various services and even different standards to be deployed via software downloads to the unified infrastructure. In the future, we envision the systems that can dynamically modify their functionality to interact with different standards and/or adapt to the changing scenarios.

Up until now, the dedicated digital signal processing (DSP) processors have been the well-known technology associated with the development of software radio systems. Typical applications involve the digitization of baseband or intermediate frequency (IF) signals, thus allowing software to perform the works except frequency (RF) processing tasks. However, the implementation complexity and cost increases when the digitization signal is as close as possible to the antenna. Meanwhile, the assembly program languages used for DSP development are often difficult to learn and it takes long time for engineers to be able to write the effective codes even sometimes C/C++ language can be used together.

Common PCs continue to be improved rapidly in processor speed and memory size while maintaining outstanding cost/performance. Another novel approach for software radio does rely on general purpose processor instead of the special purpose DSP processors. It can be seen that the cluster of common PCs is a compelling platform for implementing the wireless systems. In such platform, the commonplace language such as C/C++ and the familiar development tools can be used, which means the software radio design is open to a new community of software developers. Meanwhile, the parallel processing is being seen as the only cost-effective method for the fast solution of computationally large and data-intensive problems[1]. The emergence of inexpensive parallel computers such as commodity desktop multiprocessors and clusters of workstations has made such parallel methods generally applicable. Parallel computing has made a tremendous impact on a variety of areas including real-time digital signal processing. Recently the PC-cluster software radio wireless systems are being developed and will be paid more attention in the future broadband wireless systems due to its low-cost and high flexibility [2][3][4]. Therefore, the software radio platform with common PCs is presented to meet the real-time requirements of digital signal processing in this paper.

The article is organized as follows. First, we briefly review the system architecture of PC-cluster software radio platform. Then we discuss several schedule strategies for parallel computing. Furthermore, the demonstration of TD-SCDMA systems will be provided. Finally, the conclusion is given.

II. ARCHITECTURAL OVERVIEW

In this section, we describe the basic architectural characteristics of the open wireless PC-cluster platform. The main purpose is to implement the entire system from the physical to the networking layers by software in the common PCs. Furthermore, this platform is developed on an open-source operating system such as RedHat Linux [7].

A. Testbed Components

The hardware of the testbed for this PC-cluster platform consists of 3 elements that are under software-control, which include:

- PCI-bus based data acquisition cards with an analog/digital (A/D) and/or digital/analog (D/A) interface
- up/downconversion RF cards
- common PCs

The AD/DA converter is connected to the PCI bus for its high throughput. We have chosen the generic PCI bus because it has become the standard for data acquisition and comes in many shapes and sizes (e.g., ICS, PMC, Spectrum) for different host environments. For high-speed data acquisition, PCI64(66 MHz/64-bit, 528 Mbytes/s peak) may be used instead of low-speed PCI32(33MHz/32-bit, 133 Mbytes/s peak). The RF architectures may also be applicable according to the system requirement such as 1.28MHz bandwidth at 2G carrier for TD-SCDMA. Details of the RF subsystem are beyond the scope of this paper. The software part of this platform is developed based on the RedHat Linux operating system in order to finish
the processing of physical and upper layers. The basic hardware/software architecture for PC-cluster platform are shown in Fig.1.

B. Data acquisition
We have to face the several challenging problems when using common PCs to perform signal processing tasks. One of them is an I/O device capable of digitizing a generic wideband signal and delivering the samples to the PC’s main memory (and performing the reverse operation as well)[5].

The system must provide the applications with what appears to be a jitterless sample stream. In standard signal processing systems based on dedicated digital hardware or DSPs, the incoming samples arrive at a constant rate and are processed with a fixed delay between when a sample enters the system and when the output based on that sample leaves the system. The processing happens in clock step with the I/O, so the DSP is guaranteed that it will have a constant stream of regularly spaced samples on which to do processing. In a common PC, however, there are no such simple guarantees. Virtual memory, multiple levels of caching, and competition for the I/O and memory buses add jitter to the expected amount of time required for a sample to travel from an I/O device to the processor. In addition, using a multi-tasking operating system ensures that the signal processing application will not always be the active process, which adds jitter to the rate at which samples are processed. The jitter introduced by all of these sources must be smoothed out.

Considering the characteristics of real-time data transfer, the fluctuation of the data stream travelling to the software-to-hardware interface is not extremely intensive. For example, in a TD-SCDMA system, the data are generated frame by frame at a fixed rate (e.g. $1.28M \times N$ samples/s, $N$ is oversampling rate) at the transmitter. The amount of time for each data frame to be processed by software modules and transferred via the Ethernet doesn’t fluctuate vastly. Thus the data flow travelling to the high-speed I/O port is a stream of approximately regularly spaced data frames and each frame contains constant amount of samples. Therefore, the AD/DA converter is able to provide a almost constant stream of samples by adopting a RAM whose capacity is large enough (to contain several frames of data). In our platform, the AD/DA converter cards have two work modes: one is standard generation mode and another is FIFO mode[6]:

- Standard generation mode:
  This mode replays the data in the on-board memory by continuous repeating or by single shot only once. This mode doesn’t do data writing from PC memory to the on-board memory when it is generating waveforms. Only when the generation is finished, data transmission can be started. So the hardware control is not complex. It allows generating signal at very high sample rates.

- FIFO mode:
  It allows writing data continuously between the PCs and the AD/DA converters. That means the data writing and the signal generating can be performed simultaneously. In this mode, the on-board memory is used as continuous buffer. And it needs the software buffer of PC to work together. Fig.2 illustrates how the FIFO mode writes data from PC to the DA converter. In the DA converter, the hardware memory is separated into two buffers with the same length. In the PC, the driver can hold maximum up to 256 software buffers with the length as same as that in the DA converters. Whenever a buffer in the DA converter is empty, it generates an interrupt and the PC transfers the data from the next software buffer to the empty hardware buffer. The driver does this job automatically in the background. At the same time, the application software can generate data or load data from hard disk and fill them into software buffer. These two sets of buffers must run at the same throughput. This protects the whole system against buffer underruns or overruns. More software buffers or larger buffer size may be useful when having buffer overruns. Similar with DA converter’s FIFO procedure, in the AD converter the onboard memory is used as a continuous buffer for FIFO reading with the similar procedure.

It is shown that FIFO and buffer streaming can be applied to solve the synchronization between AD/DA converter and PCs. This eliminates the need for designers to worry about synchronization issues.

C. Radio interface protocol
The elements of the radio interface protocol are shown in Fig. 3, where the radio interface consists of three protocol layers[8]:

- Physical layer (L1):
error-resilience enabled MPEG-4 coded video transmission is considered in this platform. To realize the real-time implementations of the MPEG-4 encoder/decoder and transport, there are two methods to accomplish it.

- To use current open-source software under GNU General Public License. VideoLAN [9] is one of the most popular open source MPEG-4 players. VideoLAN media player can be used as a server for streaming MPEG-1, MPEG-2 and MPEG-4 / DivX files or used as a client to receive, decode and display the MPEG streams.

- To develop our own MPEG-4 stream transmission mechanism is also applicable with choosing the simplified transport stack appropriately. We consider UDP/IP to transport the video stream rather than with RTP/UDP/IP. Later we will also use the MPEG-4 encoder and decoder with open sources under GPL.

According to Fig.3, if we want to transfer the MPEG-4 video stream through the protocol stack, many protocols on the different layers, such as PDCP, RLC, etc. should be realized. Since that is not our main target, the necessary simplification will be applied on the interface between the different layers.

- To simplify the appropriate transport stack. In actual case of the packet switch such as UMTS system, IP/UDP/RTP transport protocols are used. RTP runs on the top of UDP, which packetizes and provides in-order delivery of video frames. RTCP is used by the video client to inform video server concerning the received video quality. In this system, no interaction is assumed; hence RTCP will not be modelled. Similarly, RTP is also not modelled, but those functions that are needed for packetization. Packet sequence numbering and in-order delivery are added to the UDP model.

- To set the interface parameters. The length of the PDU is an importance parameter, especially on the UDP layer. So the MTU of UDP layer should be confirmed firstly. It is based not only on the Ethernet itself, but also on the actual ability of the physical layer. Then according to the different physical channel types, we can get the different interface parameters.

According to the different functionality, the whole platform consists of five parts as shown in Fig.4, which includes the video server/client software, the L1 function modules, the AD/DA converter API, the software architecture and the control/monitor modules.

### III. Software Architecture

The software architecture for parallel computing is one of the key parts of PC-cluster platform. The main objective of the software architecture design is to support real-time signal processing application on a general purpose PC-cluster platform and to create a programming environment that provides for a simple and straightforward implementation of signal processing functions. In this section, we will discuss how to build up...
Video content software

Throttle, error packet number and QoS etc.

MSC selection, TCI etc.

Constellation, IEEE, etc.

Sample rate, timing, etc.

Sample rate, etc.

To distribute data/algorithms, etc.

Load on each PC, state report, etc.

User Interface

Video Frame

Data to be processed

A/D D/A card API

TD-SCDMA Frame

PC-Cluster software architecture

(a) without feedback

<table>
<thead>
<tr>
<th>Slave</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Busy</td>
</tr>
<tr>
<td>2</td>
<td>Idle</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>Idle</td>
</tr>
</tbody>
</table>

(b) with feedback

<table>
<thead>
<tr>
<th>Slave</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Busy</td>
</tr>
<tr>
<td>2</td>
<td>Idle</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>Idle</td>
</tr>
</tbody>
</table>

Figure 4: Function modules in PC-cluster software radio platform

Figure 5: Schedule strategy for parallel computing

this kind of parallel computing environment for the implementation of software radio.

Each PC can be regarded as one node in the parallel computing system. Usually several parallel programming software such as Openmosix and MPI can be used in the parallel computing systems. However, the latency and jitter of data transmission between different nodes can’t meet the real-time requirement. And their task scheduling flexibilities are not satisfied for digital signal processing applications. Therefore, we have to design the new parallel computing schemes by UDP socket programming.

Since the computation complexity of the transmitter is much less than that of the receiver, the transmitter module can be implemented in one node. For the modules of receiver, the parallel computing across several nodes is necessary. There are two kinds of nodes in the PC cluster, one is the master node and another is the slave node. The master node has to not only receive the vast continuous high-speed data from low-level hardware, but also extract useful data and distribute them to other slave nodes in the network for further processing. The data to be processed in the PC-cluster platform is the samples of the signal or the data to be modulated. To be processed in a distributed computing environment, the data stream should be divided into blocks. As we know, most of data streams that wireless communication systems handle can be separated temporally. It means that two fragments of the data stream could be processed independently if they do not overlap each other. For wireless communication system, the data streams have frame structures and can be processed based on the unit of frame. The length of a frame determines the latency time of the processing, so it cannot be too long. If we separate the data stream in frames, the data blocks will not be too large for processors to handle. On the other hand, the correlation of two frames depends on the interval between them and the coding scheme of the system. In general, the data frames have little correlation between each other when the interval between them is long enough. So the temporal decoupling is feasible for our platform. After the separation, the parallel digital processing will be done in the slave nodes.

In order to let the system work low-latency and efficiently, the parallel computing schedule has to be carefully designed and make the load of slave node well-balanced. After receiving one packet, the master node must decide in which slave node the signal will be processed. The schedule strategy will quite affect the system performance. Currently there are several schedule strategies applied in our PC-cluster platform, which can be classified into two kinds according to whether the slave nodes give the feedback message to the master node.

A. Schedule strategy without feedback

Using this strategy, the master node will distribute the packets to the slave nodes according to a prearrange rule and it won’t consider the states of slave node in the system. The round robin without feedback is the simplest way to schedule the system. As shown in Fig. 5(a), the master node will distribute the received packets to all the slave nodes circularly. This scheduling can be applied in the condition that the computation abilities of all the slave nodes are the same, which is called as the homologous computing environment. But if the computation abilities of the slave nodes differ from each other much, this scheduling will lead to the unbalanced load among them and cause the out-of-order problem in the output data stream.
B. Schedule strategy with feedback

In some scenarios we need to know the state information of the different nodes in the PC-cluster platform, which can be used to improve the robustness and efficiency of the system, especially in the heterogeneous environment. Therefore, it is necessary to know the load status of the slave nodes in the master node at the cost of the feedback overhead. Also this may cause some amount of delay in scheduling because the master node always waits for the state update of the slave node. But in order to decrease the delay of scheduling, the master node can make scheduling according to the last state table of the slave nodes. In the state table of the master node, only two states including idle or busy are assumed currently only for simplicity and more states can be included if necessary. When the master node send a new packet to the selected slave node, this slave node will work and response one message to the master node. Then the master node mark this slave node’s state as busy. When finishing the packet process, the slave node will feedback another message to master node, then master node reset the slave node’s state as idle.

Figure 5(b) shows the example of the schedule strategy with feedback. Here $i$ is the index of the received packet in the master node. If all the states of slave nodes are listed as idle in the state table, the master node will check the table from the beginning node and choose the slave#1 to process the $i$th packet, marking the state of slave#1 as busy in the table. When the $(i+1)$th packet arrives, the master node will check the table again and choose slave#2 to work because slave#2 is one of the idle nodes in the table. But it takes time to update the state table. When finishing process packet, the slave node can’t update its state immediately due to the feedback delay. So such scenarios maybe happens that all the slaves are marked as busy in the table but some slave nodes are idle indeed. In this case, the master node could choose the slave node with the most idle possibility according to the previous records.

IV. Demo System for TD-SCDMA Implementation

An implementation of the PC-cluster software radio platform with five 2GHz Pentium PCs is developed on RedHat Linux 9.0 using C++. The unique architecture of the system allows rapid development of software radio systems. For the purpose of demonstration, a TD-SCDMA system is implemented in this testbed, whose digital signal processing procedure is shown in Fig.6. The processing CPU time required by the processor varies per application. As an example, the loads of different nodes are shown in Fig.7.

Figure 6: Diagram of baseband signal process in a TD-SCDMA system

Figure 7: Example of nodes load in the PC-cluster software radio platform

V. Conclusions

The software approach based on common PCs and open operating system provides tremendous flexibility for the mobile communication systems. The prototype testbed is being developed to demonstrate that many of the concepts can be realized using today’s general-purpose processors and the AD/DA converters. As the technology in these areas continues to be improved, it will be easier to implement more flexible processing algorithm of more wireless communication systems in the PC-cluster software radio platform.

REFERENCES


