Testbed-Assisted Learning for Digital Communications Courses

José A. García-Naya, Paula M. Castro,

Miguel González-López, Adriana Dapena

Abstract

We introduce testbed-assisted learning as an effective means for teaching digital communications. Laboratory teaching activities of digital communications courses benefit very much from utilizing a hardware testbed, since it greatly facilitates the understanding of very important effects introduced by real-world transceivers. We overcome the main drawback of communications hardware, i.e. the cumbersome low-level programming interfaces provided by hardware manufacturers, by introducing a distributed multilayer software architecture. This architecture provides different abstraction levels to access hardware testbeds, releasing students from the low-level interaction with the hardware. Also, the distributed nature of this architecture results in a high flexibility of operation. This way, students can focus on learning communications topics without devoting any time to low-level programming, that is usually out of the scope of digital communications courses. Thanks to testbed-assisted learning, they are able to perform illustrative experiments to understand digital communications concepts

The authors are with the Department of Electronics and Systems, University of A Coruña, Spain. Mailing address: Facultad de Informática. Campus de Elviña, s/n. 15071 A Coruña. Spain. Tel.: +34 981167000. Fax: +34 981167160. e-mail: {jagarcia,pcastro,mgonzalezlopez,adriana}@udc.es

(e.g. source coding, modulation, space-time coding, etc.) and to test algorithms without developing a new program from scratch, speeding up both the implementation and the debugging tasks. However, those students interested in hardware implementations can use the software architecture to access and interact with lower programming level until they are as close as possible to the hardware.

Keywords: Assistant tools, testbeds, multilayer architecture, digital communications.

1 Introduction

Research shows that students learn and retain much more when they experiment directly in the laboratory, as opposed to when they only listen or see concepts in traditional classroom lectures [1, 2]. In digital communications courses, it is common to propose laboratory activities consisting on computer simulations designed to illustrate the theoretical concepts. Usually, typical experiments in under- or graduate courses assume ideal and fully controlled conditions. In this sense, computer simulations are useful as an starting point in understanding the key concepts of modern digital communications. However, very important effects introduced by hardware elements are often ignored, like for example those caused by the antennas, by the D/A and A/D converters or by the radio-frequency (RF) equipment. Some of these effects can only be well understood if the students experience the problem by themselves using hardware components.

In our experience, the utilization of a hardware testbed is the best way to experiment and learn in lectures of digital communications. In the case of wireless digital communications, there is a variety of multi-antenna testbeds that provides many educational opportunities [3], allowing the students to learn, step by step, all signal processing stages involved in the generation of the signals to be transmitted through the antennas. This transmit signal processing chain is very similar to that usually found in computer simulations. However, when the students have to implement the corresponding signal processing blocks of a real-world receiver, they find several important differences. Some of them are caused by implementation impairments, i.e. frequency and phase noise or non-linear distortions caused by RF power amplifiers.

Other differences are inherent to the fact that transmitter and receiver are situated at different physical locations. Moreover, signal processing concepts like time and frequency synchronization are never taught to the students, as a result, they will not be concerned about them.

To the knowledge of the authors, a multitude of universities as well as public and private research centres have been investing a lot of efforts in setting up testbeds with research purposes. Nevertheless, very few works [3] consider the possibility of taking advantage of the educational possibilities offered by testbeds. This motivates us to introduce the term *testbed-assisted learning*, consisting in involving testbeds in the learning process.

However, to implement and test algorithms in a testbed requires more multi-discipline skills than the theoretical computer-based approach. To develop software allowing to transmit, acquire and properly process signals involves cumbersome low-level programming to access the hardware, making difficult to test new methods, that is precisely the motivation for students to start interacting with hardware testbeds [4]. Due to this reason, it is convenient to add a mechanism to the testbed that allows its access at different levels of abstraction. This means that a student starting to implement his first algorithms should access the testbed at a higher level than another one who is prepared to deal with lower-level details of the testbed. Consequently, such a mechanism permits the students to focus exclusively on the development and test techniques, releasing them from the task of low-level programming.

Our aim is, thus, to provide a mechanism that allows undergraduate and master students to access the testbed at an abstraction level similar to that required by computer simulations. Once the student acquires skills enough to deal with lower level details, this mechanism should also permit using the testbed but still abstracting *even lower* level details. The understanding of a multilayer scheme, where each layer deals with specific problems at a different abstraction level, clearly leads to enforce the knowledge of the students.

Notice that this approach is not restricted to the digital communications field: in other areas, hardware testbeds exhibit the same problem of cumbersome low-level programming interfaces. Due to its modularity, our distributed multilayer software architecture would be easily adaptable to other types of hardware testbeds. We, however, focus on digital communications since our hardware testbed is targeted to it and most of our research knowledge comes from this area. Consequently, we will restrict our discussion to this field.

The rest of this paper is organized as follows. Section 2 presents the distributed multilayer software architecture of the testbed and describes the testbed hardware components. Section 3 details the architecture layers. Section 4 illustrates how to use the testbed-assisted tool in digital communications courses. Finally, Section 5 is devoted to the conclusions.

2 Testbed Overview

Figure 1 shows the general organization of our hardware testbed and Figure 2 shows a picture of the current equipment. The testbed is hosted by two ordinary PCs, one for the transmitter and another one for the receiver (referred to as Tx PC and Rx PC, respectively). Figure 3 shows the block diagram of the entire system. Three main parts can be distinguished (from bottom to top): the testbed hardware that allows us to transmit discrete-time signals over multiple antennas at 2.4 GHz; the multilayer software architecture that makes the hardware accessible to end users (i.e. the students); and finally, any user/student application, implemented on top of the multilayer software architecture.

The hardware of the testbed is entirely based on Sundance Multiprocessor Ltd (see the bottom of Figure 3). The transmitter consists of the SMT310Q PCI carrier board and a SMT365, a basic processing module equipped with an FPGA, a DSP, memory buffers and two buses capable of sustaining a transfer rate of 400 MB/s. The basic processing module is directly connected to the data acquisition module (DAQ module), the SMT370. It contains a dual D/A converter with dedicated memory operating at the same speed of the D/A converter. The DAQ module also has two A/D converters. Finally, the DAQ module is connected to the RF front-end module, the SMT349, which performs up and down conversion operations from 70 MHz to 2.45 GHz with 16 MHz of maximum bandwidth. The receiver employs the same configuration as the transmitter but incorporating a memory buffer module (SMT351) allowing to store in real-time the data acquired by the A/D converters to be later transferred to the Rx PC.

Our distributed multilayer software architecture is divided into three layers: the *user layer*, the *signal processing layer* and the *middleware layer*. As it can be seen in Figure 1, the lowest level layer (i.e. the middleware) is required to be installed in the computers hosting the testbed hardware, but the other two layers can be installed in any other available PCs. However, the configuration we present here simplifies the set-up because all software needed is installed in the Tx PC and in the Rx PC. Only the user layer is installed in a different PC (the student PC), so as to be available to the user application.

3 Architecture Layers

We describe in this section the three layers we developed to provide high-level access to the testbed (see Figure 3): the user layer, the signal processing layer and the middleware layer.

3.1 User Layer

The user layer interacts with the user application (usually a program that simulates a communications system) by using a simple function implemented in MATLAB® (any other software providing socket connections would also be valid). Its main task consists in sending the symbols to be transmitted as well as the necessary parameters to the signal processing layer, at the transmitter side. In the same way, the user layer receives the acquired symbols by the receiver, being noticed if any error occurs.

The main target of the user layer is making the rest of the layers accessible to high level applications, taking its type of development environment into account. For this reason, the user layer is jointly executed with such applications (see Figure 3).

3.2 Signal Processing Layer

The signal processing layer is network-connected with both the user and the middleware layers (see Figure 3). It decouples them, acting as an intermediate between them. This layer supports remote access and consists of two different processes that carry out the signal processing operations needed to link the user and the middleware layers. The first process (TxProc) receives the symbol vectors from the user layer and performs the up sampling, pulse-shape filtering, I/Q modulation and frame assembling operations in order to generate the IF signals that will be sent to the middleware. Similarly, the second process (RxProc) waits for the acquired signals from the middleware and performs the time and frequency synchronization operations followed by the I/Q demodulation, filtering and down sampling. It sends the resulting vectors to the user layer next. Note that depending on the type of the signals sent to the signal processing layer, this layer could perform all or only some of the supported operations. For example, if the input signals are already I/Q modulated, then the signal processing layer acts as a bypass.

3.3 Middleware Layer

The middleware concept constitutes a great leap forward in hardware testbed technology, making the testbed hardware accessible through ordinary network connections. It means that the middleware is commanded using socket connections and, afterwards, the middleware provides the response using such socket connections. This layer fills the gap between the testbed hardware and the signal processing layer, allowing discrete-time signals to be transferred through the PCI bus and making possible the synchronization between the Tx PC and the Rx PC using a network connection.

The middleware is divided into two different sub-layers (see Figure 3). The *top* sub-layer is responsible of establishing the network connections between the transmitter and the receiver, and with the higher-level layer (the signal processing layer). The *bottom* sub-layer corresponds to the testbed hardware configuration and control software.

The middleware is constituted by four different processes. The first two processes (TxHost and RxHost) implement the top sub-layer and run, respectively, on the Tx PC and the Rx PC. They are implemented in standard C++ language and use sockets to establish the necessary network connections: one between the TxHost and RxHost processes (used to synchronize the transmitter and the receiver, so as to the receiver knows when the signal acquisition process has to start); another one, established between the TxHost process and the Tx signal processing layer; and, finally, another one between the RxHost process and the Rx signal processing layer. The remaining two processes are the transmitter and the receiver processes that run on their respective DSPs included in the testbed hardware. They implement the bottom sub-layer. The transmitter DSP process (`TxDSP`) performs data transfers through the PCI bus jointly with the `TxHost` process and configures and controls the hardware components at the Tx PC. In the same way, the `RxHost` process and the DSP receiver process (`RxDSP`) are responsible of transferring the data through the PCI bus and, from the DSP side, controlling and configuring the testbed hardware components at the Rx PC.

The multilayer software architecture provides a high abstraction level, which allows us to implement user applications without knowing the testbed hardware details. In order to show the advantages derived from the use of a multilayer software architecture, Figure 4 plots a frame transmission step by step:

- Once the symbols to be transmitted have been generated at the user application, a function is called passing to it the corresponding symbol vectors (one vector per transmit antenna).
- These symbols are sent to the signal processing layer (`TxProc`) through the user layer, where they are converted (if necessary) to pass band signals that are subsequently sent to the middleware.
- When both the Tx PC and the Rx PC are ready to complete a transmission, the signals are passed to the testbed hardware to be transmitted by the antennas.
- At the receiver side, the middleware stores the signals into the hardware buffers and then it forwards them to the receiver signal processing layer (`RxProc`).
- Finally, the acquired signals are forwarded to the user application through the user layer, completing the entire process.

Using our multilayer software architecture, the transmission process is transparent to the students and, therefore, the students will have the same vision like in conventional computer simulations but with the advantage that the transmission is doing using real hardware.

In this work we will restrict our approach to describe the benefits provided by the use of the multilayer software architecture in an academic environment, but it is worth emphasizing that the testbed plus the architecture also enables many teaching possibilities for more advanced, like PhD students, by accessing the testbed at specific layers.

4 Testbed-Assisted Learning

Thanks to the abstraction level provided by our distributed multilayer software architecture, it is very easy to devise programs simulating communications systems using testbed-assisted learning oriented to both graduate and master courses, such the ones we are currently teaching:

- Undergraduate course “Introduction to Digital Communications”. This course focuses on the analysis and design of digital communications systems. We cover coding, digital modulation and demodulation techniques, performance of detection of modulated signals in noise and limited bandwidth channels, and application of these techniques to practical systems.

- Master course “Advanced Topics in Wireless Communications”. The students should be familiar with the basics of wireless digital communications, such as modulation formats, symbol rate and occupied bandwidth, the concept of SNR (Signal-to-Noise Ratio), matched filtering and optimum detection, etc. The course focuses on systems with several transmit and receive antennas and on the utilization of space-time codes. An important part of the course is devoted to explain channel estimation algorithms, including pilot-assisted and unsupervised techniques.

In this case, the learning goal of testbed-assisted learning is help in explaining concepts about wireless digital communications, including the following topics:

- *Topic 1 – Modulation*. Our multilayer software architecture allows us to test transmissions of either complex-valued discrete symbol sequences or pulse-shaped signals, both base band or pass band. Several parameters affect performance such as the number of bits to be transmitted, or how they are generated (equiprobable source or Gaussian); the modulation type (PAM, PSK or QAM), and the number of levels of the modulation; the number of samples for each symbol or the pulse-shape form to be used (rectangular, square-root raised cosine, ...) [5].

- *Topic 2 – Intermediate Frequency*. In communications and electronic engineering, it is very frequent to use an intermediate frequency (IF) as an intermediate step in transmission or reception between base band and RF [5]. Our multilayer software architecture allows us to experiment with different values of the intermediate frequency (f_0) and the sampling frequency (f_s).

- *Topic 3 – Space-time codes*. A space-time code (STC) is a method employed to improve the reliability of data transmission links in wireless communications systems that make use of multiple transmit antennas [6, 7]. STC can be divided into two main types:

  - Space-time trellis codes (STTCs) that distribute a trellis code over multiple antennas and multiple time-slots. They provide both coding gain and diversity gain.

  - Space-time block codes (STBCs) that act on a block of data at once. They provide only diversity gain, but are much less complex in implementation terms than STTCs.

  A special class of STBC is termed Orthogonal-STBC (OSTBC) because the coding matrix is orthogonal. This idea was introduced by Alamouti in [8] considering a system with two transmit antennas and only one receive antenna (Alamouti 2×1), and generalized by Tarokh *et al.* in [9] for any number of antennas. Our testbed plus the multilayer software architecture allows us to experiment with uncoded systems (1×1) and with OSTBCs considering different number of transmit and receive antennas.

- *Topic 4 – Channel Estimation Techniques*. The performance of communications systems scheme strongly depends on the channel estimation accuracy. Depending on the information needed to estimate the channel, the techniques can be classified into two groups:

- Supervised algorithms that utilize pilot symbols known by both the transmitter and the receiver. Among the supervised methods, the Least Squares (LS) criterion [10] constitutes a frequent starting point, given the simplicity of the resulting technique.

- Unsupervised algorithms where the channel is directly estimated from the received signals, assuming that both the transmitted signals and the channel are completely unknown at the receiver. Some of these algorithms have been proposed in the context of blind source separation (BSS), being the most known the so-called joint approximate diagonalization of eigenmatrices (JADE) [11] and FastICA [12]. More recently, several algorithms for OSTBCs which take advantage of the specific structure of these codes have been designed (e.g. [13] and [14]).

The utilization of a hardware testbed plus our multilayer software architecture allows to devise a very rich set of laboratory exercises. As an example, we present below one experiment used in our teaching.

4.1 Example Experiment: Wireless Transmission

This experiment is oriented to show how to transmit and acquire signals using the distributed multilayer software architecture. A simple MATLAB® code (see Figure 5) suffices to perform the construction of the I/Q signals to be transmitted through the hardware testbed. Equiprobable bits are generated and mapped to 4-QAM symbols using the Gray code. The symbols are then filtered using a square-root raised cosine pulse shape.

Figure 6 shows the MATLAB® code that performs the transmission through the wireless channel and the acquisition of signals by the hardware testbed. Transmission is performed at the 2.4 GHz carrier frequency. All is done by simply calling the function provided by the multilayer software architecture, `txAndRx()`, thus making access to the testbed really simple. Note that only one signal vector is provided, thus it will be transmitted by the 1st TX antenna. The function returns the data as it has been directly acquired after the ADCs (`signalMatrixRxRaw`), and after synchronization (`signalMatrixRx`). The parameter `retry` is equal to 1 if the acquisition failed. The variable `SNREstdB` contains the estimated SNR at the RX.

For illustrative purposes, the MATLAB® code only performs the processing of the signal received at the first antenna of the testbed. Figure 7 shows the constellation obtained at reception after matched filtering. The estimated SNR at reception is 28 dB and the BER is zero.

5 Conclusion

We have introduced the concept of testbed-assisted learning, a very powerful tool to help in the teaching process of digital communications. We presented a distributed multilayer software architecture that overcomes the main problem of hardware testbeds: the cumbersome low-level programming interfaces provided by hardware manufacturers. This architecture provides different abstraction levels to accommodate students with different skills, making the hardware testbed useful for both graduate and master courses. Learning through experimentation with real data transmissions in realistic environments greatly enforces the students to understand real-world problems in digital communications. Also, it clearly contributes to increase their motivation as well as their personal reward in learning digital communications.

The concept of testbed-assisted learning and the distributed multilayer software architecture that supports it are not limited to the field of digital communications. Many other areas can benefit from it, since the modularity of our software architecture makes it easily adaptable to other fields of interest.

Acknowledgements

This work has been funded by the Xunta de Galicia, the Ministerio de Ciencia e Innovación of Spain, and the FEDER funds of the European Union under the grants with numbers PGIDIT06TIC10501PR, TEC2007-68020-C04-01, and CSD2008-00010.

References

[1] A. Diong, R. Wicker, D. Pieana, and Q. R., “A laboratory designed to enhance students’ interest in learning of controls,” *International Journal of Engineering Education*, vol. 20, no. 4, pp. 628–637, 2004.

[2] W.-J. Shyr, “Integrating laboratory activity into a junior high school classroom,” *IEEE Transactions on Education*, vol. 53, no. 1, pp. 32–37, 2010.

[3] R. Rao, W. Zhu, S. Lang, C. Oberli, D. Browne, J. Bhatia, J.-F. Frigon, J. Wang, P. Gupta, H. Lee, D. Liu, S. Wong, M. Fitz, B. Daneshrad, and O. Takeshita, “Multi-antenna testbeds for research and education in wireless communications,” *IEEE Communications Magazine*, vol. 42, no. 12, pp. 72–81, Dec 2004.

[4] M. Rupp, S. Caban, and C. Mehlführer, “Challenges in building MIMO testbeds,” in *Proc. of the 13th European Signal Processing Conference (EUSIPCO 2007)*, Poznan, Poland, Sep. 2007.

[5] B. Sklar, *Digital Communications: Fundamentals and Applications*. Prentice Hall, 2001.

[6] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and H. V. Poor, *MIMO Wireless Communications*. Cambridge University Press, 2007.

[7] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[8] S. Alamouti, “A simple transmit diversity technique for wireless communications,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, Oct 1998.

[9] V. Tarokh, N. Seshadri, and A. R. Calderbank, “Space-time codes for high data rate wireless communication: Performance analysis and code construction,” *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 744–765, 1998.

[10] S. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice Hall Information and System Sciences Series, 2001.

[11] J. Cardoso and A. Souloumiac, “Blind beamforming for non-gaussian signals,” *IEE Proceedings F Radar and Signal Processing*, vol. 140, no. 6, pp. 362–370, Dec 1993.

[12] E. Bingham and A. Hyvärinen, “A fast fixed-point algorithm for independent component analysis of complex valued signals,” *International Journal of Neural Systems*, vol. 10, pp. 1–8, 2000.

[13] E. Beres and R. Adve, “Blind channel estimation for orthogonal STBC in MISO systems,” *IEEE Transactions on Vehicular Technology*, vol. 56, no. 4, pp. 2042–2050, July 2007.

[14] H. J. Pérez, J. García-Naya, A. Dapena, L. Castedo, and V. Zarzoso, “Blind channel identification in Alamouti coded systems: A comparative study of eigendecomposition methods in indoor transmissions at 2.4 GHz,” *European Transactions on Telecomunications. Special Issue: European Wireless*, vol. 19, no. 7, pp. 751–2008, 2008.

Figure 1: General organization of the testbed and the student PC.

Figure 2: Testbed picture showing the Tx PC, the Rx PC, and the student PC (a laptop).

Figure 3: Platform scheme.

Figure 4: Example of a transmission using the distributed multilayer software architecture, and the testbed hardware.

Figure 5: Example experiment: MATLAB® code for generating the I/Q signals.

Figure 6: Example experiment: MATLAB® code for transmission through the testbed and acquisition.

Figure 7: Experiment: Symbol constellation corresponding to the acquired signals.