Reprint

APSK Coded Modulation Techniques: From Simulink Models to DSP Implementation

P. Savvopoulos, M. Varsamou, N. Papandreou, Th. Antonakopoulos and V.Makios

The European DSP Education and Research Symposium – EDERS 2004

BIRMINGHAM, UK , NOVEMBER 2004

APSK Coded Modulation Techniques: From Simulink Models to DSP Implementation

P. Savvopoulos, M. Varsamou, N. Papandreou, T. Antonakopoulos and V. Makios

Research Academic Computer Technology Institute – CTI, 61 Riga Feraiou Str., 26100 Patras, Greece
Department of Electrical Engineering and Computers Technology, University of Patras, 26500 Rio - Patras, Greece
e-mail: {psavvop, varsamou, npapandr, theodore, makios}@loe.ee.upatras.gr

ABSTRACT

This paper presents the application of a versatile and flexible environment on prototyping data transmission devices that are based on digital signal processors (DSPs). This development environment integrates the Matlab/Simulink tools with the TI DSPs in a unified set-up that allows interaction between the model running on Matlab/Simulink and the software modules running on the DSP platform, through a data exchange mechanism via the PCMCIA interface. The application example described makes use of the APSK modulation technique that has been decided for the second generation of satellite broadband communications systems.

INTRODUCTION

The increasing demand for mobile, broadband communications has made the satellite systems an efficient and cost-effective solution for providing communication services, especially in rural areas. The capability of providing high speed and high bandwidth communications to large geographical areas, where the infrastructure of other communications systems is negligible, emphasizes the flexibility and the potential of such systems.

Great effort is devoted in the broadband satellite communications area for developing new technologies and perspectives, in order to enforce and extend the market success over a wide range of different groups of customers [1]. New efficient, yet complicated, algorithms along with new enhanced standards arise, with the demand of short time to market implementation, so as to gain advantage over other competitive technologies [2]. Given the described status on demonstrating new products in the field of satellite broadband communications, the necessity for more processing power makes DSP and/or FPGA platforms the best solution not only for their verification and validation but also for the critical phase of development and implementation.

According to this concept, this work presents a powerful and versatile methodology for analyzing, developing and prototyping of data communication and signal processing systems on a flexible platform that consists of a single or multiple DSP processors and reprogrammable logic, communicating with the Matlab/Simulink tools as well. As an illustrative example, the paper presents the implementation of APSK coded modulation techniques on a TMS320C6711 DSP processor, using the described methodology.

The next two Sections present the architecture of the development environment including the hardware platform and its interconnection with the simulation tools, while the last two Sections give a detailed description of the implementation of APSK coded modulation techniques using the aforementioned environment.

THE DEVELOPMENT ENVIRONMENT

Prototyping of communications and signal processing systems is a complicated task that involves several discrete design steps. Initially, an analytical system model has to be developed, the various algorithms have to be designed and the system behavior needs to be verified. As a next step, the respective prototype which combines a number of hardware and software functional modules has to be implemented and tested in terms of its consistency to the functionality of the analytical model. Due to the complexity of their algorithms, modern communications systems need long development and testing time for the completion of a prototype. Regarding the modeling and testing of a communications system, Matlab/Simulink tools offer a high performance simulation environment that supports the development and analysis of complex multi-domain models [3].

In this paper, we discuss an environment that provides an effective design and test approach, by exploiting the high-performance simulation and modeling capabilities of the Matlab/Simulink tools and the flexible modular hardware architecture of a prototype platform. The design methodology involves the use of the Matlab/Simulink tools for building and verifying the analytical model and then mapping selected system blocks into DSP processes on the prototype platform. These Matlab/Simulink blocks are being replaced by special library functions that are

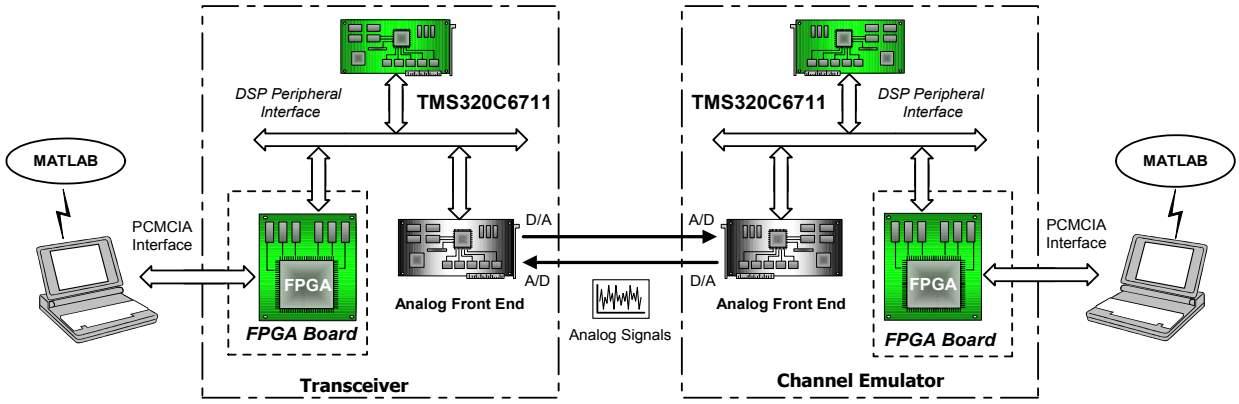

Figure 1. The complete development environment

responsible for the communication and synchronization with their DSP counterparts [4]. Throughout the development process we can utilize various Matlab-based testing tools in order to verify the proper behavior of all integration steps. This procedure continues until the high-level model functions have been integrated into a complete prototype.

In order to implement a low-cost and flexible prototype environment, we developed a hardware platform that is based on the high-performance floating point TMS320C6711 processor with 900 MFLOPS processing power [5], an analog front-end unit that includes two ADC and two DAC channels and an FPGA module with 8 kwords of external Dual Port RAM. Data exchange between the hardware platform and the computer that hosts the Matlab/Simulink tools is performed through the PCMCIA interface, which is implemented in the FPGA. By these means, we achieve the extension of the available memory space that can be commonly accessed by the Matlab workspace via an appropriate I/O device driver. The data exchange and synchronization between the model and its blocks that are mapped into the prototype platform, is accomplished through Matlab/Simulink custom functions that associate workspace variables with memory locations and structures at the interface memory space. The DPRAM contains all necessary user and control data required for the efficient implementation of the mixed-type model. In the current version of the prototype platform, all subsystems are implemented as DSP functions that are synchronized by the Matlab/Simulink environment which coordinates the whole procedure.

The most commonly used development set-up, which corresponds to a pair of communicating devices (implemented in the same hardware platform) and a real-time channel emulator (implemented in a separate platform) is shown in Figure 1, while it can be extended to support a separate platform for each communicating device. This set-up provides the capability of designing flexible component level architectures, thus enabling the implementation and verification of several end-to-end communication applications.

DSP INTEGRATION IN THE MATLAB/SIMULINK ENVIRONMENT

In a mixed-level design that includes a Matlab/Simulink model and various processing modules that are implemented as DSP functions in the hardware platform, synchronization plays a key role in effective and proper system prototyping. Synchronization has two different aspects: synchronization of the various functions performed at the same DSP processor-environment, and synchronization that is related to the data exchange between the Matlab workspace and the DSP platform.

Multi-threading in the DSP environment

The progressive substitution of several Simulink blocks by their respective DSP implementations, results in the development of DSP modules that perform a number of different functions simultaneously, often in response to external events, such as the availability of data or the presence of a control signal. These functions are called *threads* and multi-threaded programs run on a single processor by allowing higher-priority threads to preempt lower-priority ones and by allowing various types of interaction between threads, including blocking, communication, and synchronization. The used DSP processor supports multi-threading applications through DSP/BIOS that is supplied along with the Code Composer Development Suite. DSP/BIOS is a scalable real-time kernel, which is designed for applications that require real-time scheduling and synchronization, host-to-target communication, and real-time instrumentation. It provides preemptive multi-threading and system-level services such as memory management, communication mechanisms and interrupt handling [6]. Using these kernel features, distinct submodules of a Matlab/Simulink

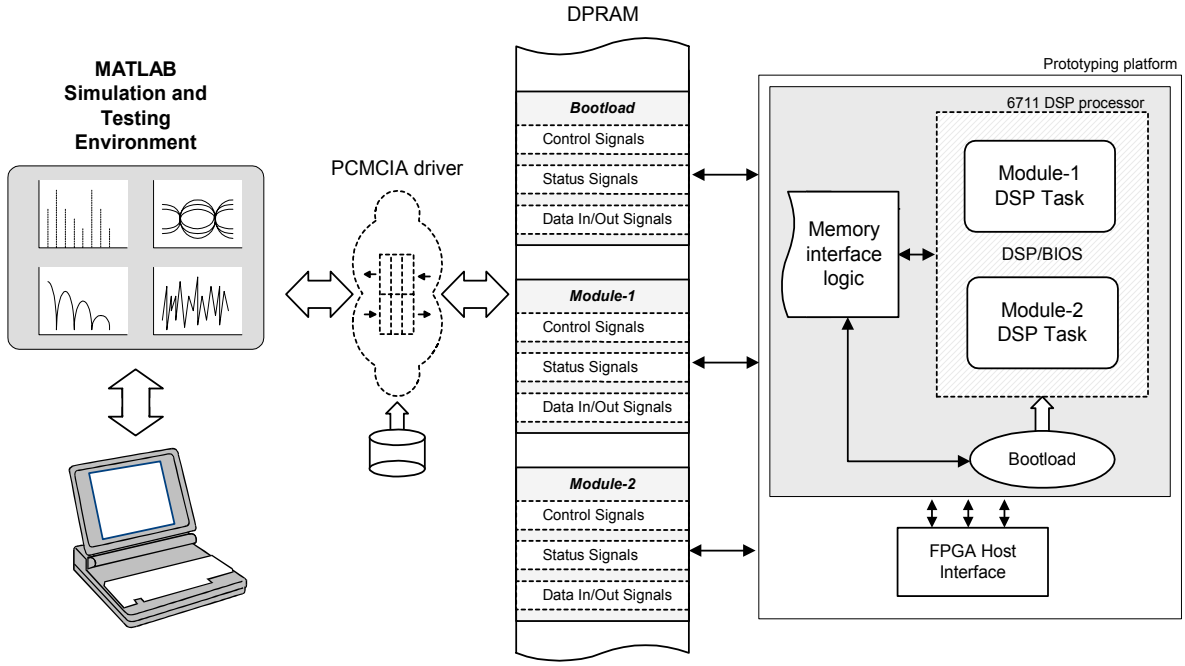

Figure 2. The Matlab-to-DSP interface

model can be mapped to independent or cooperative DSP functions that are executed concurrently.

Simulink and hardware platform synchronization

Synchronization should not only be achieved among distinct DSP functions, but also between the Matlab/Simulink functions and each DSP module, in order to ensure that the dataflow through the different stages of the system model is consistent with the system's specifications.

In our environment, data exchange is performed through the PCMCIA interface. In the high-level model, the blocks that are translated into DSP software modules in the prototyping platform are replaced by special functions that utilize a PCMCIA driver, in order to access the PCMCIA device. In the prototyping platform a custom FPGA module performs the interface logic. This custom module provides the necessary circuits for the PCMCIA initialization, signal transactions and extension of the available I/O memory space. It also controls the dual-port memory (DPRAM), which is accessed by both the simulation workspace and the custom DSP modules and provides the storage area of data and control information that is exchanged between the simulation environment and the hardware platform. In fact, the DPRAM provides the physical interface between the DSP processes, translated from model blocks, and their simulation environment.

In Figure 2, we demonstrate the general concept of the information exchange between the simulation environment and the DSP-implemented blocks of the system prototype. Each system module is associated with a specific region in the DPRAM, which contains the interface signals between that module and its simulation environment. In general, we distinguish the following interface signals regions:

- ***Control signals region***, which contains control information and data for the configuration of the system modules and the initialization of the communication between the modules and the simulation environment.
- ***Status signals region***, which contains the status information of the system modules. This information is mainly used for synchronization of the dataflow between the simulation model and the prototyping platform. Furthermore, it can be utilized for the debugging of the subsystem's functionality.
- ***Data in/out signals region***, which contains the data provided to or generated from the corresponding system module in order to participate in the model's dataflow.

When a certain block of a Matlab/Simulink model, like an APSK modulator as in the following application example, is replaced by the respective DSP function, synchronization is achieved by exchanging control information using the DPRAM. Matlab produces the data to be transferred to the DSP module for processing, and then triggers its execution until the response is received.

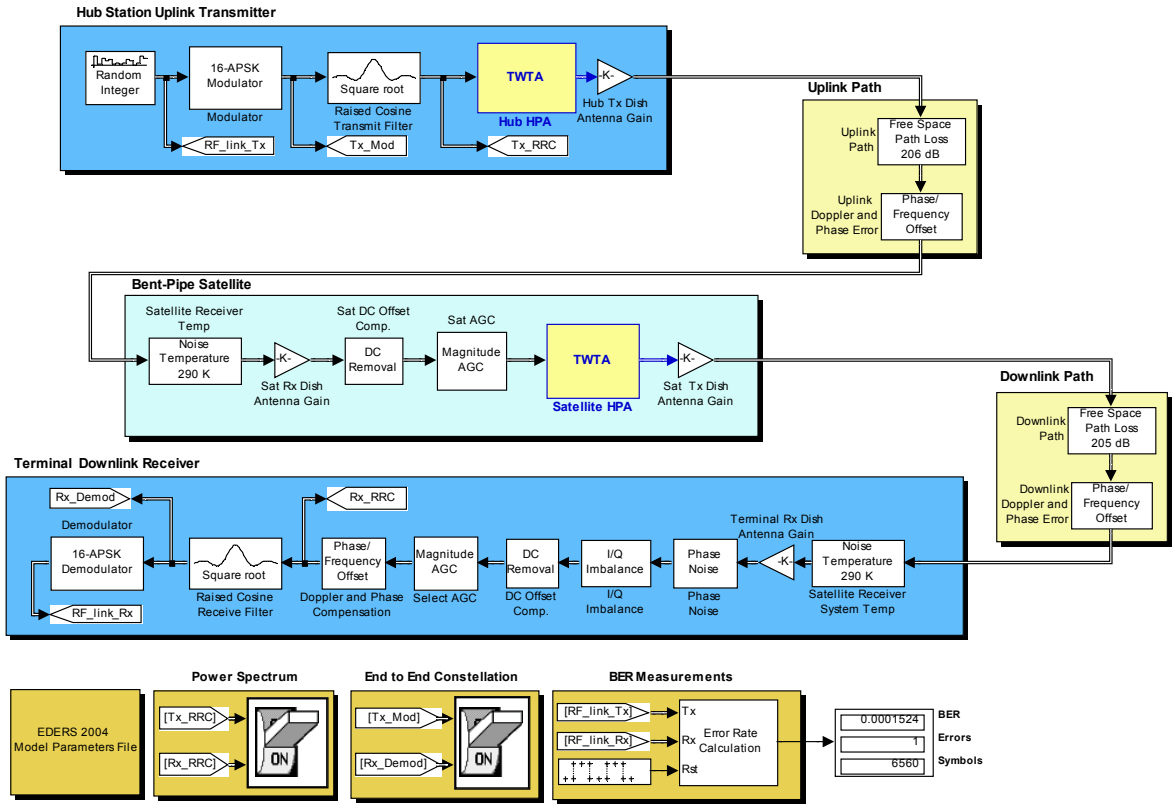

Figure 3. The application example system model

In addition to the model-specific DSP implementations, a separate bootload DSP module has been developed that enables the configuration of the DSP processor with new executables through the MATLAB environment and the PCMCIA interface. A special MATLAB function reads the Code Composer generated executable files and transfers the binary data to the DPRAM along with information about their organization. Then the DSP bootload program writes the data to the specified memory locations and transfers the control to the new application program. The synchronization between MATLAB and the bootload is achieved through a protocol that uses control information mapped on special locations in the DPRAM. In the future, we are planning to extend the FPGA-based interface capabilities, by connecting the JTAG signals of the DSP to a hardware module that will be accessible via the PCMCIA, thus providing more debugging functionality to our development environment.

THE SYSTEM MODEL

For demonstrating the applicability of the presented design methodology and the use of the hardware prototype platform in the development and implementation of a digital communications system, we use a satellite communication system model using a 16-APSK modulation technique, as a representative example (Figure 3).

The first generation of European broadband satellite systems, which was based on the DVB-S standard in the early 90's, has already been deployed in Europe and other parts of the world. Many devices supporting the functions and services of DVB-S standard are working reliably and properly with success for many years, which resulted to the wide spreading of the use of DVB-S satellite technology for broadband communications. In the beginning of 2004, the second generation of broadband satellite systems, called DVB-S2 [2], came along in order to improve and expand the initial DVB-S standard. The aim of the new standardization was to increase the efficiency and flexibility of such systems based on the peculiarities of the satellite link [7]. Link particularities, like satellite and hub station non-linear distortion, introduced by high power amplifiers (HPAs), the channel fading distortions and carrier phase noise at each receiver, are taken into account in the design process.

Given the intention for evolution of satellite communications, the new perspective in satellite system design introduces high order modulation such as the 16-

ary and 32-ary APSK (Amplitude Phase Shift Keying). This modulation technique, taking into consideration the satellite channel characteristics, has been proven [8] that minimizes link losses when proper digital signal processing algorithms are being employed on both ends of a point-to-point link. The 16-APSK modulation technique that is implemented on the described development environment, consists of two rings of PSK modulation. The inner ring includes 4 constellation points with 90^o degrees difference between them, while the outer ring includes 12 points with 30^o difference. Many studies [9] proved that this modulation scheme is resilient to the non-linearity of the high power amplifiers that are used on the ground station and the transponder lying on the space segment. It is also reported that APSK modulations outperform classical modulation types (16-QAM, 16-PSK) currently used in satellite links.

A typical GEO (Geostationary Earth Orbit) satellite forward channel system is modeled in the Matlab/Simulink environment. The model is based on a transmitter (Hub station Uplink Transmitter), using the aforementioned modulation type, that sends traffic to a far-end receiver (Terminal Downlink Receiver), making use of the respective demodulation. The connection in the forward channel is managed through a space segment of a bent-pipe satellite that interfaces the two terrestrial stations. This procedure takes place on a transponder that receives the original signal, amplifies it and drives it through downlink path to the terminal, using another carrier frequency.

The respective Hub station model sends data to the terminal derived from a binary random source generator, that produces data frames of integer numbers (symbols) consisting of 4 bits each. The 16-APSK modulator translates the received data into Gray-encoded constellation points, which comprise the I and Q transmission channels. The outputs of the modulator passes through square-root raised cosine band-limiting filters, used to shape the output signal in order to minimize the ISI (Inter-Symbol Interference) throughout the satellite link. An HPA module amplifies the filtered output. Non linear behavior is observed due to the fact that the HPA of the ground station operates near its saturation point, introducing non linear distortion. The last module of the transmitter model is the parabolic dish antenna.

Uplink and downlink paths introduce two kinds of impairments that result to signal attenuation at the receiver of the transponder and the terminal, respectively. These impairments are modeled in our system as two different modules per path, each one contributing with a different way on the overall signal distortion. One source of impairment in the satellite link is the free space loss which depends on the carrier frequency (different for the two links) and the distance between the hub/terminal and the satellite. Additional source of impairment is the phase and frequency offsets introduced on the traveling signal, due to the relevant movement of the space segment (Doppler Effect).

Within the satellite, the transponder is responsible for the reception of the transmitted signal through the satellite receiver dish antenna. As the IF signal is generated, a DC removal and a closed-loop AGC module (Automatic Gain Control) adjust the incoming signal for proper feeding of the HPA-TWTA of the transponder. At this stage additional non linear distortion is inserted on the transmitted signal. Afterwards the dish antenna transmits the signal over the downlink path of the forward channel. The signal then reaches the terminal's far-end receiver where various functions are executed, in order to demodulate the incoming signal and extract the initially transmitted data. In the current version of our model, no error correction mechanisms are used.

IMPLEMENTATION ISSUES

Following the exhaustive verification of the described satellite communication system model in the Matlab/Simulink environment, various submodules can be substituted by their DSP software counterparts. Specifically, the 16-APSK modulator at the transmitter and the respective demodulator at the receiver, along with their Square Root Raised Cosine (S-RRC) filters are replaced with relevant implementations on the same DSP environment. Figure 4 presents the key features of the implementation process Two DSP tasks are formed, one for the transmitter side and the other for the receiver that run concurrently and independently on the same DSP.

The internal logic of the modulator is based on the representation of 16 different symbols with a number of 16 constellation points into a look up table. In other words, for every single integer number (symbol) coming from the random integer source generator, a complex number is extracted. Each of the components of the complex output, as soon as it is upsampled by a specific factor, it passes through an S-RRC shaping pulse filter, reducing ISI through the satellite forward channel, and a new complex output is produced.

The inverse operation is performed at the receiver side before the execution of the symbol recovering function. Initially, the S-RRC is applied and immediately after the downsampler restores the sampling rate of the original transmitted signal with the same factor used on the transmitter side. The function of the demodulator is more complicated, since the minimum distance of each received symbol from all points of the constellation diagram has to be determined. That distance is related to the symbol most probably sent.

In order to achieve the goal of two independent processing functions running concurrently on the same DSP processor, we exploited the capabilities of the DSP/BIOS real-time kernel feature, supported by the TMS320C6xxx DSP processors. The DSP/BIOS schedules the timing execution of the two software modules and treats them as two different threads with the

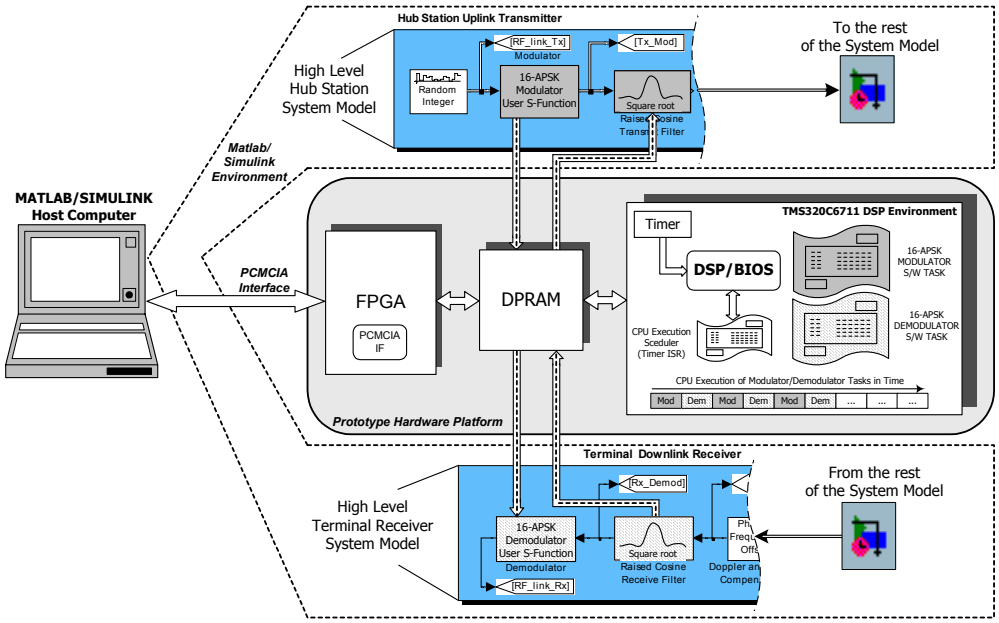

Figure 4. Implementation of the application example

same priority level. Both tasks use different control, status and data exchange regions in the DPRAM for achieving interconnection with the Matlab/Simulink simulation environment.

CONCLUSIONS

In this work, a dynamic development environment for prototyping communications systems, consisting of Matlab/Simulink tools being interconnected with a flexible hardware platform, was presented. The analyzed design methodology gives the opportunity to develop communications systems in a progressive manner, allowing the transformation of a high-level simulation model after extensive tests and verification to a fully functional, reliable system prototype. This perspective on system realization can be extremely useful for the design of complicated and resource demanding communications systems. The described environment can also be used for significant educational purposes on system modeling, validation and prototyping.

REFERENCES

[1]. Alberto Morello and Ulrich Reimers, "DVB-S2, the second generation standard for satellite broadcasting and unicasting", *International Journal of Satellite Communications and Networking*, Special Issue: The DVB-S2 Standard for Broadband Satellite Systems, Vol. 22, Issue 3, 2004, pp. 249-268.

[2]. Draft ETSI EN 302 307 (V1.1.1): "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive services, News Gathering and other broadband satellite applications", *European Standard (Telecommunications series)*, June 2004.

[3]. M. Varsamou, P. Savvopoulos, N. Papandreou and T. Antonakopoulos, "From Matlab/Simulink Models to Prototype Implementation: A Communication Systems Development Environment", *The Nordic MATLAB Conference 2003*, Copenhagen, Denmark, October 2003.

[4]. The Mathworks Inc., Writing S-Functions, Revised for Simulink 5.0 (Release 13), July 2002.

[5]. Orsys GmbH., User's Guide Micro-Line C6711CPU/C6712CPU, High Performance Digital Signal Processor Family, Rev 4.02, Mar. 2003.

[6]. Texas Instruments, Inc., TMS320 DSP/BIOS User's Guide, Nov 2002.

[7]. Ernest Chen, Joshua L. Koslov, Vittoria Mignone and Joseph Santoru, "DVB-S2 backward-compatible modes: a bride between the present and the future", *International Journal of Satellite Communications and Networking*, Special Issue: The DVB-S2 Standard for Broadband Satellite Systems, Vol. 22, Issue 3, 2004, pp. 341-365.

[8]. E. Casini, E. De Gaudenzi and A. Ginesi, "DVB-S2 modem algorithms design and performance over typical satellite channels", *International Journal of Satellite Communications and Networking*, Special Issue: The DVB-S2 Standard for Broadband Satellite Systems, Vol. 22, Issue 3, 2004, pp. 281-318.

[9]. R. De Gaudenzi, A. Guillen i Fabregas, A. Martinez Vicente, B. Ponticelli, "APSK Coded Modulation Schemes for Nonlinear Satellite Channels with High Power and Spectral Efficiency", *in the Proc. of the AIAA Satellite Communication Systems Conference 2002*, Montreal, Canada, May 2002.