Reprint

xDSL Systems Prototyping using a Flexible Emulation Environment

N. Papandreou, M. Varsamou and T. Antonakopoulos

The 14th IEEE International Workshop on Rapid System Prototyping - RSP'03

SAN DIEGO, CA, JUNE 9-11, 2003

xDSL Systems Prototyping using a Flexible Emulation Environment

Nikolaos Papandreou
Academic Research
Computers Technology Institute
61 Riga Feraiou Str., 26100 Patras, Greece
npapandr@cti.gr

Maria Varsamou, Theodore Antonakopoulos
University of Patras
Department of Electrical Engineering
26500 Rio, Patras, Greece
{varsamou, theodore}@loe.ee.upatras.gr

Abstract

In this paper we describe the methodology and architecture of a flexible modular environment for prototyping data transmission systems and its application on xDSL systems. The development environment is based on custom and commercially available software tools and a custom hardware emulation platform for mapping the basic data-pump modules of xDSL systems into hardware/software functional modules. The road-map from a high-level xDSL system model to the actual prototype is based on the progressive substitution of high-level submodules of the initial model with their respective hardware/software counterparts, and their integration into a complete functional system. A library of custom blocks is used for data exchange and synchronization between the high-level model and the emulation platform, and for real-time visualization of the critical parameters of the emulated system as well. The application of the proposed development environment in the implementation and testing of an emulator of a bundle of DSL lines and of a centralized bit-loading algorithm for multicarrier ADSL systems is also described.

1. Introduction

Prototyping of complex communication systems involves a number of concise and discrete design steps, starting from the development and verification of the analytical model up to the implementation of the corresponding system into a prototype that combines multiple hardware and software functional modules. The prototype system has to be consistent with the functional requirements of the analytical model and further determine an optimized and cost-effective solution. Modern digital subscriber line (DSL) technology systems [1], [2] involve complex design architectures, in digital and analog domains. Sophisticated signal processing algorithms are used for implementing the basic data-pump operations [3], [4] increasing the data-load and processing power requirements for the physical circuits that implement these operations. Such complexity introduces long development and testing times for the complete prototype system. On the other hand commercial, high-performance simulation tools facilitate the development and analysis of complex multi-domain models via off-the-shelf library modules that can interoperate with application-specific, user-defined functions. Moreover, verified simulation models that correspond to standardized industry protocols are usually offered enabling the fast evaluation of new algorithms.

In this paper we present the methodology and architecture of a flexible modular environment for prototyping various types of data transmission systems and we focus on xDSL systems and specifically on asymmetrical digital subscriber line (ADSL) systems. The proposed environment combines the high-performance simulation capabilities of a commercially available simulation tool with a flexible hardware platform and provides a rapid prototyping approach that transforms simulation blocks into component level circuits, thus building a mixed high-level analytical and low-level circuit/microcode model. The road-map from the xDSL simulation model to the actual prototype is based on the progressive substitution of the top-level submodules with their respective hardware counterparts and their integration into a complete functional system. A library of custom blocks is used for data exchange and synchronization between the high-level model and the emulation platform and for real-time visualization of the critical parameters of the emulated system. The proposed environment offers a rapid prototyping approach that reduces the total development time, since it provides a direct link between the prototype hardware system and the simulation tool, so that the high-level performance analysis can also be exploited for assisting the testing of the implemented prototype.

Section 2 describes the used methodology for rapid prototyping of xDSL communication systems and discusses the design steps from the high-level analytical model to the low-level prototype system. Section 3 describes the architecture

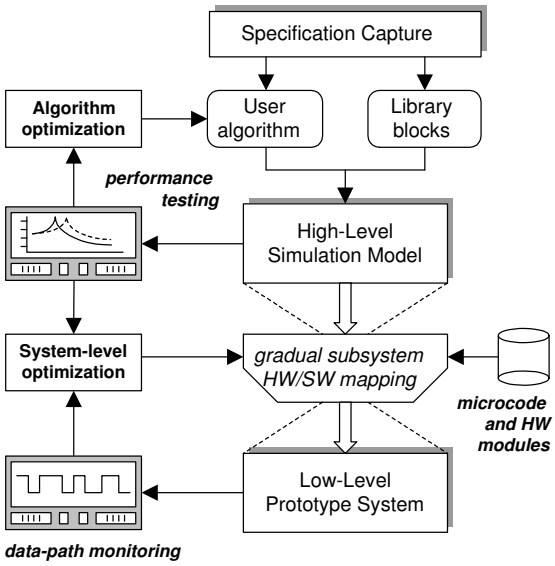

Figure 1. System prototyping methodology

of the design environment and highlights its flexibility and modularity in prototyping a complete data transmission system. Finally, Section 4 presents two examples that demonstrate the application of the proposed methodology on the development of ADSL systems.

2. The Prototyping Methodology

The top-down prototyping methodology is based on a number of discrete steps that determine how a high-level system model is transformed to the actual system prototype. Figure 1 describes the top-down methodology for prototyping a communication system.

As the first step the formulation of the system specifications is considered. This step supplies the necessary information regarding system functionality and implementation restrictions, and enables the detailed investigation of the system complexity from the early beginning of the design process. At the next step a software tool[1] is used for building a simulation model via off-the-shelf library blocks and user-defined modules. The user-defined modules implement application specific functions and custom algorithms at a high abstraction level. Simulation is used for validating the system's functionality according to the initial specifications and for estimating its performance under different data sequences. Moreover, simulation results provide the means for optimization of the model subsystems and study of various aspects of the system architecture. This step also

[1]In our system we use the MATLAB simulation environment for system modelling.

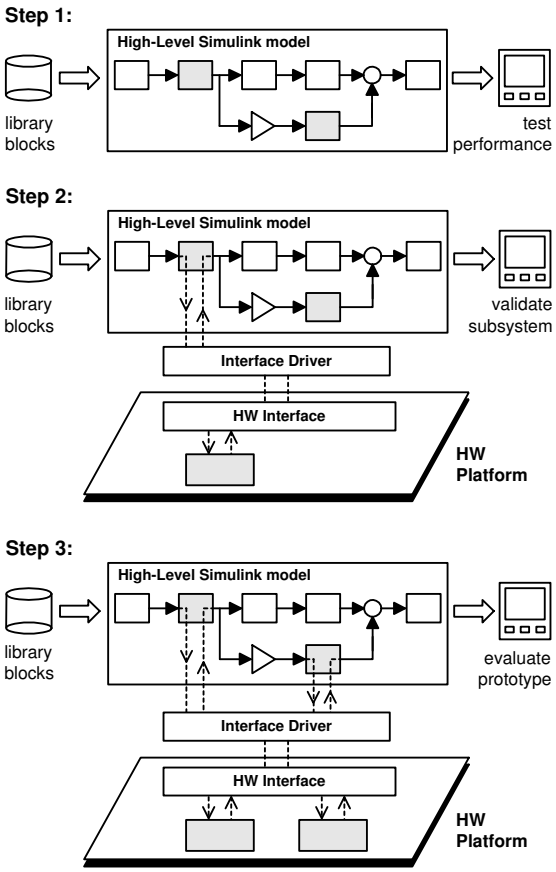

Figure 2. Progressive substitution of simulation blocks with emulator's submodules

includes the optimization of custom algorithms and their integration in the general system architecture, in order to improve the overall system performance. After the high-level simulation model has been developed and its functionality has been verified, the next step is to map selected system components into hardware and/or software modules.

Figure 2 describes the progressive substitution of simulation blocks with their hardware/software counterparts. Communication between the simulation tool and the emulation platform through data exchange and synchronization enables the integration of the top-level design and the low-level circuit components into a complete functional system. The new mixed-level model is examined through simulation tests that now reflect also implementation details. Low-level data-path debugging along with the top-level simulation results provide an overall verification process for the system prototype. The hierarchical process presented in Figure 1 describes a modular and flexible environment for prototyping a communication system. The high-level simulation model is directly interfaced with the low-level hardware

prototype, thus enabling the design and testing of sophisticated algorithms into hardware and/or software IP cores. Moreover the top-down system data-flow (see Figure 2) acts also as a complete test-bench for the prototyped circuits. Optimization is performed for each component individually as well as for the complete system. Use of reprogrammable devices (FPGAs) along with floating point DSPs in the prototype platform provides the flexibility to design a low-level architecture that is application dependent. Reconfiguration of the hardware architecture can be realized cost and time effectively during the optimization and verification steps of the top-down prototype process.

3. The Emulator's Architecture

In order to have a low-cost, expandable and flexible emulation environment, we developed a hardware platform that is based on reprogrammable and reconfigurable devices (FPGAs and DSPs). Figure 3 demonstrates the environment for developing and prototyping a communication system. The hardware platform is based on two FPGAs and two DSPs, which are interconnected using a mesh topology. This topology provides the capability to design a flexible component-level architecture that corresponds to the architectural requirements of each application. In the current prototype version, we used Virtex-II FPGAs, having 2 Million gates in total, while as a DSP we used the high-performance floating point TMS320C6711 processor with 900 MFLOPS processing power. This configuration enables the development of complex systems and the exploration of more efficient designs. The platform expansion capability based on the structure depicted in Figure 3 increases the overall design flexibility. For applications with high demand in system resources, the platform can be easily extended via the mesh topology to include additional devices (FPGAs and/or DSPs).

The high-level simulation blocks are physically assigned to programmable logic and/or DSP circuits and are mapped into HW and/or SW modules of the system architecture. Interconnection between the high-level simulation model and the low-level circuits is performed via a custom FPGA module, that allows concurrent data exchange between the simulation model and the various prototyped submodules. The hardware emulator communicates with the host computer using the PCMCIA interface. Data exchange is performed using a dual-port memory that contains all necessary user and control data for the mixed-model data-flow (see Figure 2).

The mixed-model, which is based on high-level simulation blocks and component-level prototype modules, represents a complete functional system that combines model-level performance simulation and implementation-level debugging. In particular, commercial real-time debugging tools such as the Code Composer of TI and the ChipScope of XILINX provide the essential means for investigating the details of the implementation and optimizing the hardware prototype. On the other hand, complex build-in functions provided by the simulation environment, including FFT-Scope, Averaging-Spectrum-Analyzer, Cross-Correlator and more, enable the run-time visualization of system parameters that would otherwise require a more complex and expensive setup.

4. Application Examples

In this section we present two examples that demonstrate the application of the proposed methodology on prototyping ADSL system components. The first example demonstrates how the emulation environment can be used for designing and testing an emulator of a binder of xDSL lines, while the second example shows how a bit-loading algorithm of an ADSL system is prototyped and tested.

4.1. xDSL binder-loop emulation

xDSL systems use the twisted-pair cables' infrastructure for connecting a modem at the user premises with a modem at the central office (CO). There is a bundle of lines connected at the CO, and each line is affected by noise generated by the other lines due to crosstalk interference. There are several models in the literature that describe how this noise is introduced, based on measurements of various installations. When new xDSL modems are under development and testing, there is the need to have a real-time xDSL binder emulator that can reliably emulate the performance of a real binder. That means that the xDSL binder emulator has to be parameterized in terms of number of lines, their lengths, the type of transmitted data (HDSL, ADSL, T1 etc.), their transfer function, crosstalk generated noise etc.

In this example, we describe the development of a prototype of such an xDSL binder-loop emulation system, using the previously described methodology. The emulator consists of a high level model that is used for defining the binder's characteristics and the hardware platform for real-time emulation of the line performance, especially the crosstalk interference. This emulation system can be used as a testing equipment for evaluating the performance of ADSL modems operating over subscriber loops that share the same binder. Figure 4 presents the binder-emulation setup.

We assume a binder of M modems and we discuss only the issues of the downstream transmission in an ADSL line, although the emulator supports full-duplex operation. An external analog front-end (AFE) board is used that performs

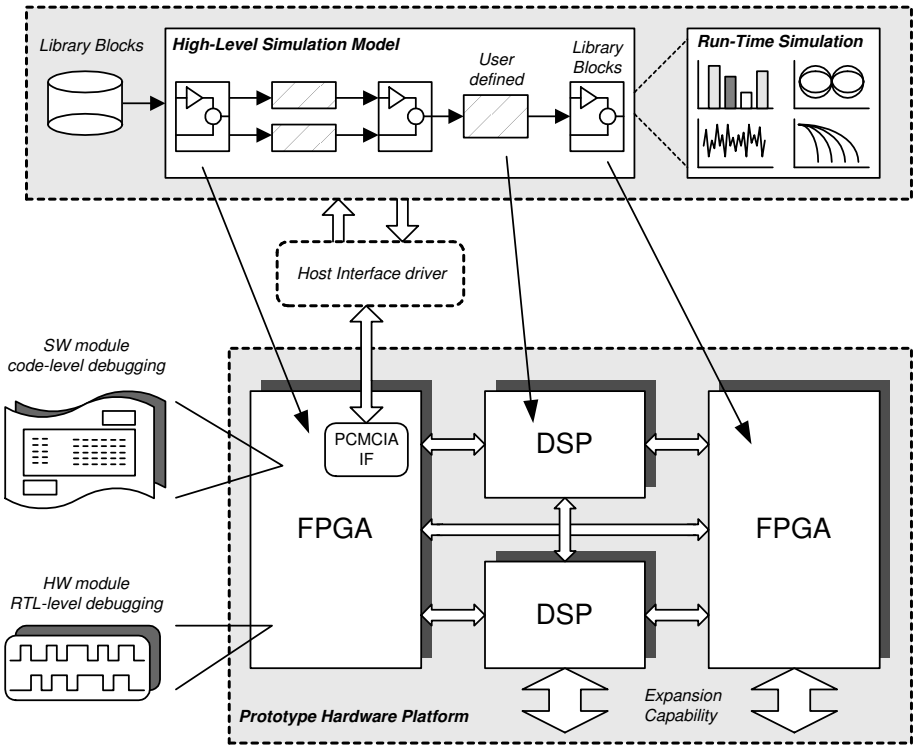

Figure 3. Modular environment for prototyping a communication system

all necessary A/D conversions as well as the required filtering functions in order to produce the discrete samples of the signals that are transmitted into the DSL line. These samples are supplied to the hardware emulator platform where an FFT module is used for transforming the signal into its frequency components. The distortion inserted due to the DSL interference environment is emulated using the analytical model that is briefly explained below.

Assuming a N-point FFT, we denote as $X_{k=1:M}$ the N-point FFT block of the transmitted signal for the kth modem, which is the modem of interest. The received signal's FFT-block is denoted as $Y_{k=1:M}$ and U_k is used for the AWGN random process. Denoting as $H_{k,k}$ the loop transfer function of the kth modem and as $H_{i,k(i\neq k)}$ the far-end crosstalk (FEXT) transfer function, that represents the interference generated from the signals of the ith modem, the following relation reflects the binder interference environment [5].

$$Y_k = H_{k,k}X_k + \sum_{i=1,i\neq k}^{M} H_{i,k}X_i + U_k \qquad (1)$$

Near-end crosstalk (NEXT) has been neglected, assuming that the ADSL modems use frequency division for multiplexing downstream and upstream traffic [1], although the analysis can be easily extended to include also NEXT interference. Figure 4 shows a graphical description of the DSL interference environment. Implementation of (1) is realized using real-time calculations (e.g. for applying the transfer function of the channel of interest to its input data) and pre-stored signal values (e.g. for the transfer function of crosstalk) by exploiting the modular mixed-level design capability of the prototyping environment presented in Section 3. The loop and crosstalk transfer functions are calculated based on analytical expressions [6]. These N-point components are pre-stored in the system and form a database for all standard test-loops [3]. The crosstalk X_i and noise U_k components are provided to the emulator from the high level traffic generator. These components are calculated via analytical simulation models, which in particular reflect the disturber's xDSL technology specifications. Figure 4 shows an example of the basic blocks comprising the simulation models. The disturber's transmitter consists of a Bernoulli random data generator followed by the encoding/modulation and filtering function blocks based on the specifications of the DSL technology standards. These blocks create the signals that generate the crosstalk to the line of interest. An N-point FFT block is used for generating the crosstalk components of (1). AWGN components are calculated in a similar manner based on the noise-

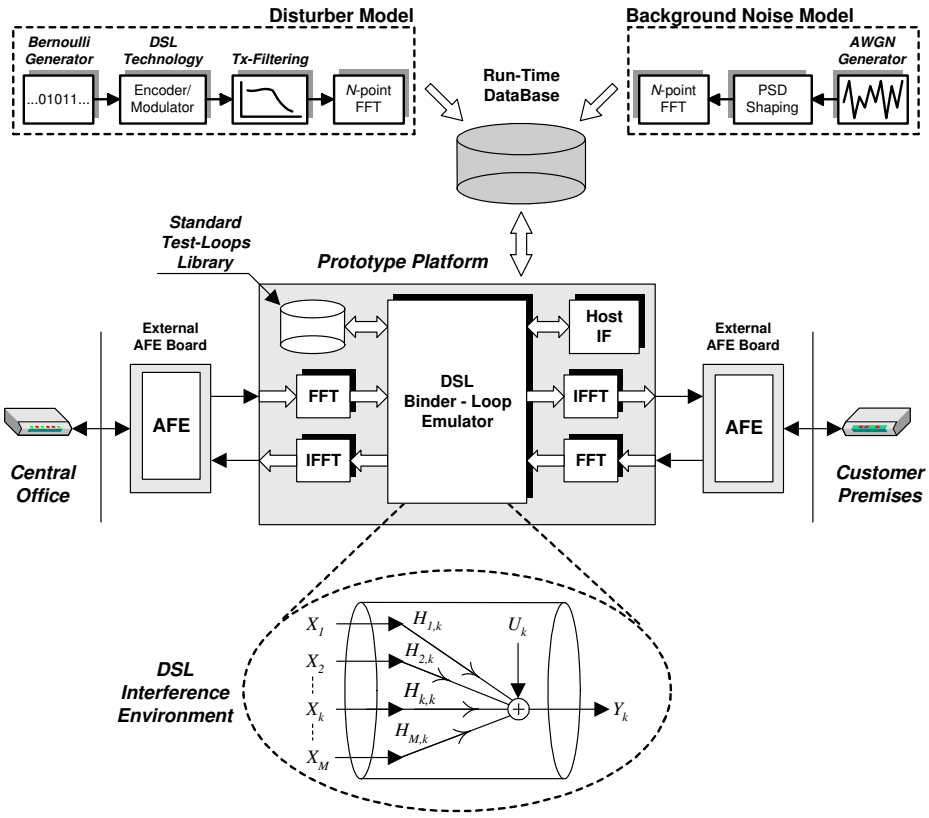

Figure 4. DSL binder-loop emulator

level of the test procedure. Run-time emulation of (1) is achieved by means of asynchronous data transfer from the host-computer to the binder's emulator. Using a FIFOs structure of size much larger than N, all operations involved in (1) are performed within the measurement time-base. For this purpose, the crosstalk and noise components are pre-calculated and pre-stored in files and FIFOs are used for controlling the transfer of data to the emulator. Figure 5 presents the implementation details of (1).

Based on the architecture of the hardware platform depicted in Figure 4, the calculations involved in (1) are assigned to the high-performance floating-point DSP devices, while the input/output FFT operations are implemented in the FPGAs using custom optimized HW cores. The output samples are finally fed to the second AFE board at the output of the emulator for D/A and filtering operations. The signal is next supplied to the modem at the receiver's side. The DSL binder system was originally developed in MATLAB using simulation models for the downstream ADSL transmitter and receiver circuits as well as for the binder environment. The hardware implementation of the DSL interference emulation functions was tested and validated through the mixed high-level model and low-level component simulation process described in Section 2.

The described binder emulator provides an efficient testing tool for emulating the crosstalk interference environment replacing long cables and noise injection circuits. Moreover it can be used as worst-case crosstalk emulation for performance evaluation of single-pair ADSL systems. In this case the power spectral density of the various disturbers is modelled based on well-defined expressions [6]. Programmability of the background noise level as well as of the type of the crosstalk disturber provides an appealing option of the system.

4.2. Bit-loading control

In multicarrier modulation (MCM) systems the channel spectrum is decomposed into a set of independent narrowband subchannels [7]. ADSL modems use a form of MCM, known as discrete multitone (DMT). In DMT, the transmitter power budget, and consequently the achievable data rate, is distributed among the discrete frequency bands based on the signal-to-noise ratio (SNR) of each subchannel. The scheme used to assign power and bits is known as bit-loading (BL) algorithm, and in general there exist two types of algorithms - those that try to maximize data rate and those that try to maximize performance at a given data

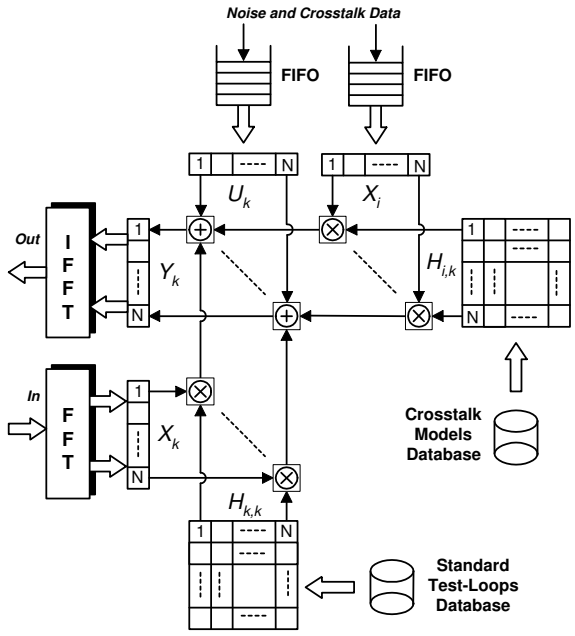

Figure 5. The crosstalk interference model

rate. Both algorithms use the estimations of the subchannel SNRs based on measurements during the modem initialization phase. In subchannels with higher SNR levels, more bits are allocated. Many optimal or sub-optimal algorithms are reported in the literature for computing the bit and power allocation profiles of a single multicarrier link [8], [9]. Recently the development of methods for coordination among the MCM modems sharing the same binder in order to improve the binder-system performance has become an appealing challenge [10], [11]. In this example, we describe the prototype of a module that implements the BL algorithm for ADSL systems, using the modular environment described in section 3. The BL module is implemented as an independent SW core that calculates the bit and power profiles based on the channel conditions. The initial profiles are generated during link establishment and specify the encoding and decoding process of transmission. On the other hand, a significant increase in the noise level will force the system to re-evaluate its profiles, in order to maintain its QoS. In both cases, the BL module uses the estimated SNR values for each subchannel in order to utilize efficiently the available spectrum. These values are stored in memory tables and along with the transmission parameters including total power-budget, power spectral density (PSD) mask, system-margin and target-rates define the BL algorithm input-data.

Figure 6 describes the BL prototyping system. The high-level model simulates a complete ADSL transmission link including the transceiver's data-pump functions, as defined in [3], and the DSL interference channel described in the previous section. The transmission is based on bit and gain tables whose values are read during the simulation run-time. System performance is examined by means of bit-error-rate (BER) and SNR degradation, using custom simulation blocks. The BL mechanism is implemented in the prototype platform as an independent module. Initialization or update of the BL profiles is controlled by the device controller, the soft IP MicroBlaze processor core. Communication between the hardware platform and the high-level model is performed via the PCMCIA interface. The hardware architecture is based on the CoreConnect bus. The complete system is tested through run-time change of channel conditions in the high-level simulation environment. The new subchannel SNR values are calculated and transferred to the HW platform. In case of performance degradation the BL module re-evaluates the bit and gain profiles which are fed back to the high-level model. The new profiles will affect the transmission at the next simulation step. Verification of the proper system response, after the channel change, is realized using the high-level simulation tools by examination of transmission performance before and after the application of the BL mechanism. The described BL prototype consists an independent soft-core module that can be embodied in a complete multicarrier system. The methodology used for the BL prototype can be extended for developing a centralized mechanism that controls the bit and power profiles of a system of MCM modems that share the same binder. In that case the individual loop and crosstalk transfer functions, described in the previous section, need to be known and stored in memory tables of the BL module. Moreover the high-level simulation blocks can be further decomposed into hardware or software modules in order to develop an ADSL prototype. The modular environment of Figure 3 enables the gradual implementation of the complete data-pump as well as the investigation of the architectural details, based on the embedded HW and SW cores.

5. Conclusions

In this paper we have presented a rapid prototyping approach that can be used for developing and testing data transmission systems. The used methodology allows the combination of high-level system models along with prototyped hardware and software components in a flexible integrated environment. The main advantage of the proposed approach is that the system designer can start from a high-level description of the system, develop its testing and verification procedures and reuse the same environment for testing the actual prototype. This can be achieved by progressively replacing the various modules of the high-level model with their respective HW/SW counterparts and then use the same test sequences for verifying their proper functionality.

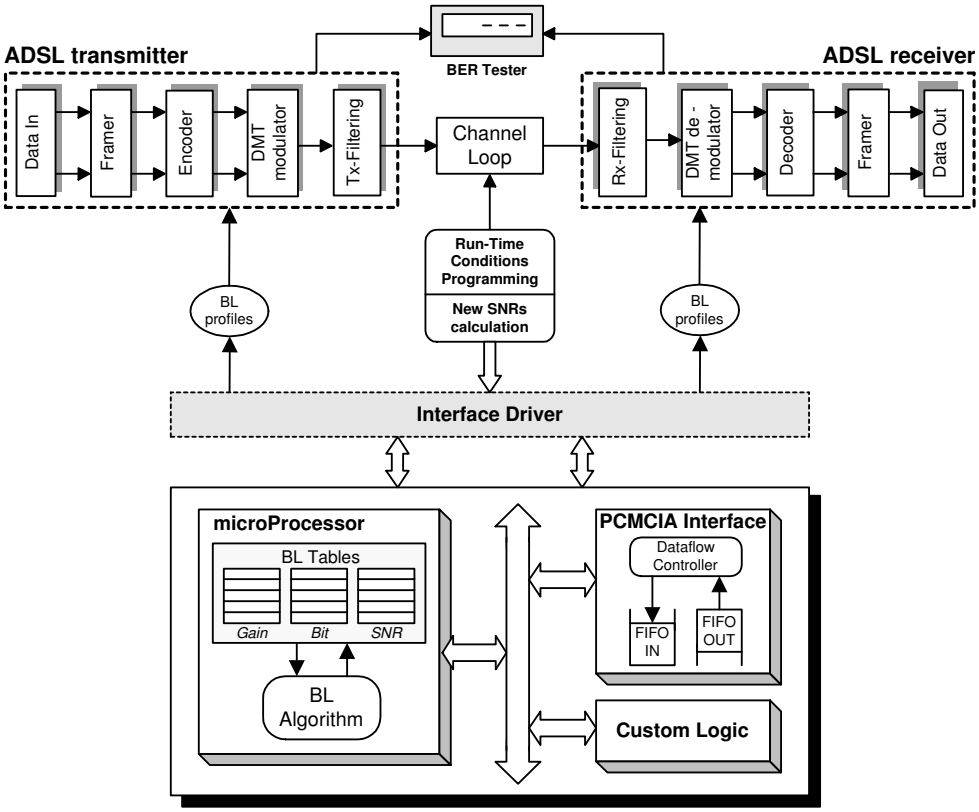

Figure 6. The bit-loading algorithm implementation module

6. Acknowledgments

This work was partially supported by the "Karatheodoris" R&D program of the University of Patras and Project 00BE33 entitled "Digital Subscriber Lines Technology" of the Greek Ministry of Industry. The authors would like to thank Mr P. Savvopoulos, Mr T. Arampatzis and Mrs D. Anastasiadou for their valuable contributions during the prototype development.

References

[1] T. Starr, J. M. Cioffi, and P. J. Silverman, *Understanding Digital Subscriber Line Technology*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[2] W. Y. Chen, *DSL: Simulation Techniques and Standards Development for Digital Subscriber Line Systems*. Indianapolis, IN: Macmillan Technical Publishing, 1998.

[3] "Asymmetrical Digital Subscriber Line (ADSL) Transceivers," ITU - G.992.1, July 1999.

[4] "Very-High-Bit-Rate Digital Subscriber Line (VDSL) Metallic Interface: Part 1: Functional requirements and common specification. Part 3: Technical specification of a multi-carrier modulation transceiver," ANSI - T1E1.4/2000-013R1, 2000-009R3.

[5] G. Tauböck and W. Henkel, "MIMO Systems in the Subscriber-Line Network," *The 5th International OFDM-Workshop 2000*, Hamburg, Germany, pp. 18.1-18.3.

[6] "Spectrum Management for Loop Transmission Systems," ANSI Std. T1.417-2001, Jan. 2001.

[7] J. M. Cioffi, "A Multicarrier Primer," T1E1.4/91-159, Nov. 1991.

[8] A. Leke and J. M. Cioffi, "A maximum rate loading algorithm for discrete multitone modulation systems," *IEEE GLOBECOM '97*, Nov. 1997, pp. 1514-1518.

[9] R. V. Sonalkar and R. R. Shively, "An Efficient Bit-Loading Algorithm for DMT Applications," *IEEE Commun. Lett.*, vol. 4, pp. 80–82, Mar. 2000.

[10] J. M. Cioffi *et al.*, "Proposed Scope and Mission for Dynamic Spectrum Management," T1E1.4/2001-188R4, Greensboro, NC, Nov. 2001.

[11] W. Yu, G. Ginis, and J. M. Cioffi, "Distributed Multiuser Power Control for Digital Subscriber Lines," *IEEE J. Select. Areas Commun.*, vol. 20, pp. 1105–1115, June 2002.