Reprint

Transmission Systems Prototyping based on Stateflow/Simulink Models

N. Papandreou, M. Varsamou, and Th. Antonakopoulos

The 15th IEEE International Workshop on Rapid System Prototyping - RSP 2004

GENEVA, SWITZERLAND, JUNE 2004

Transmission Systems Prototyping based on Stateflow/Simulink Models

Nikolaos Papandreou[1], Maria Varsamou[2] and Theodore Antonakopoulos[2]

[1]Research Academic
Computer Technology Institute - RACTI
61 Riga Feraiou Str., 26100 Patras, Greece
npapandr@cti.gr

[2]University of Patras
Department of Electrical Engineering
26500 Rio - Patras, Greece
{varsamou, theodore}@loe.ee.upatras.gr

Abstract

In this paper we describe an efficient methodology for rapid prototyping of data transmission systems based on Stateflow/Simulink models using a multi-level system development and testing approach. Transmission systems incorporate multi-domain functions and algorithms, i.e. physical layer circuits and communication protocol controllers. The Stateflow/Simulink environment enables the development of precise simulation models that include signal and protocol processing units. The proposed prototyping methodology is based on the progressive translation of high-level model blocks into hardware/software modules of the prototype architecture using custom and/or automated code generation tools. A custom data exchange and synchronization interface between the Stateflow/Simulink workspace and the circuit modules enables the integration of the simulation model and the prototyping platform into a complete functional system. The application of the proposed methodology in the development of an ADSL modem in a custom prototyping platform is also described.

1. Introduction

The development of data transmission systems requires an efficient methodology for mapping the system functions and algorithms into its architectural components. The design steps should ensure accurate implementation and short development times. As a first step, an analytical model needs to be developed that describes the functional system behavior. Then, a system prototype has to be implemented and tested in terms of its consistency to the specifications of the analytical model. Based on the design requirements and complexity, the system architecture usually combines multiple hardware and/or software components. In [1], we have presented a flexible environment for prototyping xDSL system components, based on a custom interface that enables the integration of model blocks and actual circuits in a prototyping platform, where the MATLAB simulation environment was used for high-level system modeling.

In general, data transmission systems incorporate multi-domain operations, i.e. physical layer circuits and communication protocol units. Although the algorithms embedded in different levels of the functional hierarchy may be independent in terms of implementation, the verification of each unit requires a complete system testing environment, in order to evaluate the implementation under all possible conditions. As the system complexity increases, the development of such a test-bench becomes a laborious task that leads to long development and testing times.

On the other hand, the Stateflow/Simulink tools from the MATLAB simulation environment facilitate the development of complex models that include data and signal processing functions, as well as protocol state-machine modules. We can therefore build a complete model of the data transmission system using custom and 'off-the-shelf' library blocks and moreover benefit from the embedded build-in simulation functions, in order to analyze and evaluate new algorithms in the complete system model.

In this paper, extending the work in [1], we describe a methodology for prototyping a complete data transmission system using verified Stateflow/Simulnk models of a fully functional system. The methodology is based on the progressive mapping of the high-level model blocks into architectural blocks of a prototype system. This mapping is achieved using custom and/or automated code generation tools. The substituted model blocks are replaced by special library functions that are responsible for the communication and synchronization with their circuit/microcode counterparts.

Section 2 describes the prototyping methodology and discusses the designs steps from the analytical model to the system prototype. In Section 3 we discuss the modeling environment using the Stateflow and Simulink tools and we present the integration of the simulation model and the hard-

Proceedings of the 15th IEEE International Workshop on Rapid System Prototyping (RSP'04)
1074-6005/04 $20.00 © 2004 **IEEE**

ware platform. Finally, in Section 4 the application of the proposed methodology in prototyping an asymmetric digital subscriber line (ADSL) modem on a custom prototyping platform is presented in details.

2. The Prototyping Methodology

The proposed prototyping methodology, as it is shown in Figure 1, is based on a top-down process which determines how an actual prototype is generated starting from a high-level simulation model. Initially, the specifications are defined and all information regarding the design requirements, the system's functionality and complexity are collected. This step also involves the system's functional decomposition into circuit components and protocol units, as well as the definition of the signal and data interfaces between each component and its environment, according to the architectural requirements and restrictions of the prototype platform.

Based on this information, an analytical model that describes the system's functionality is developed using the MATLAB environment. Using the Stateflow and Simulink tools and various blocksets, we are able to build a multi-domain simulation model that combines the complete dataflow determined by the specifications, along with signal processing and data transmission algorithms, and also communication protocol entities based on multiple state-machine modules. We can therefore build a fully operational system model and also evaluate standard as well as custom algorithms. The model's functionality is verified and optimized using simulations and an initial estimation of the implementation's complexity can be performed. The high-level simulation environment is used also for generating a set of testing scenarios in order to evaluate normal as well as exceptional cases.

After the model design is completed, the next step is to map the model components into hardware/software modules in the prototyping platform, based on the design architecture. The progressive substitution of the model components with circuit/microcode modules can be realized using automated code generation tools or custom implementation designs. The MATLAB environment provides special libraries for automated code generation of standard functions (i.e. FIR/IIR filters, FFT operations etc.) supporting selected families of Texas Instruments, Inc., Motorola, Inc. and Xilinx, Inc., devices. In the simulation model, the translated model blocks are replaced by special functions that provide communication between the simulation workspace and the circuit modules in the prototyping platform. In our development environment, this communication is realized via a PCMCIA-based custom interface that supports data exchange and synchronization. As a result, the mixed-level model that consists of simulation model blocks

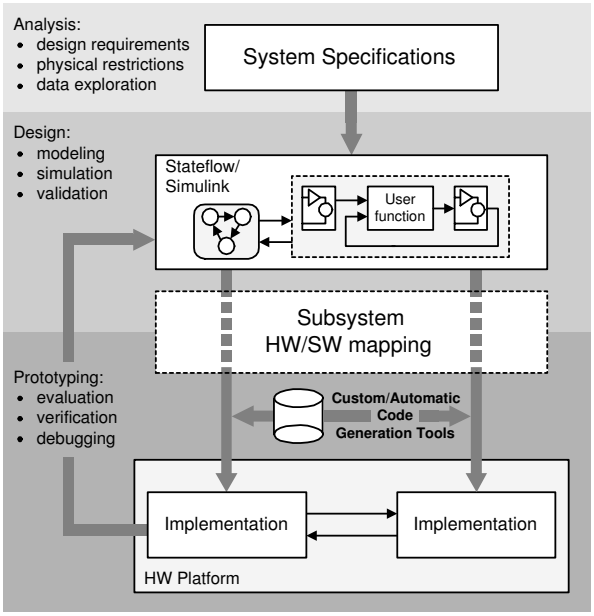

Figure 1. The prototyping methodology.

and circuit modules, is also a complete functional system model, in which selected functions are executed in the prototyping platform. Moreover, this mixed-level model provides a complete and analytical testbench for each of the prototyping modules, since the testing tools and procedures determined at the verification of the analytical high-level system, can be reused.

3. Stateflow/Simulink Modeling

The Stateflow/Simulink tools supplied by The Mathworks Inc. enable the development of a complete transmission system model that includes multi-domain design blocks corresponding to the system's data-pump functions, communication protocol units as well as other emulated modules, e.g. transmission channel.

For the implementation of the protocol procedures as well as for the supervisory logic controlling the complete communication system, Stateflow is used. Stateflow is a graphical design tool for modeling and simulating event-driven systems, using advanced finite state machines (FSM) that support representation of both hierarchical and parallel states. These FSMs can also be tightly integrated with other continuous or discrete-time standard and/or custom Simulink blocks to form a complex dynamic systems.

In general, the protocol FSMs implement sequential procedures related to the initialization and management of the system operation and provide to the data-pump circuits essential control and data information depending on the trans-

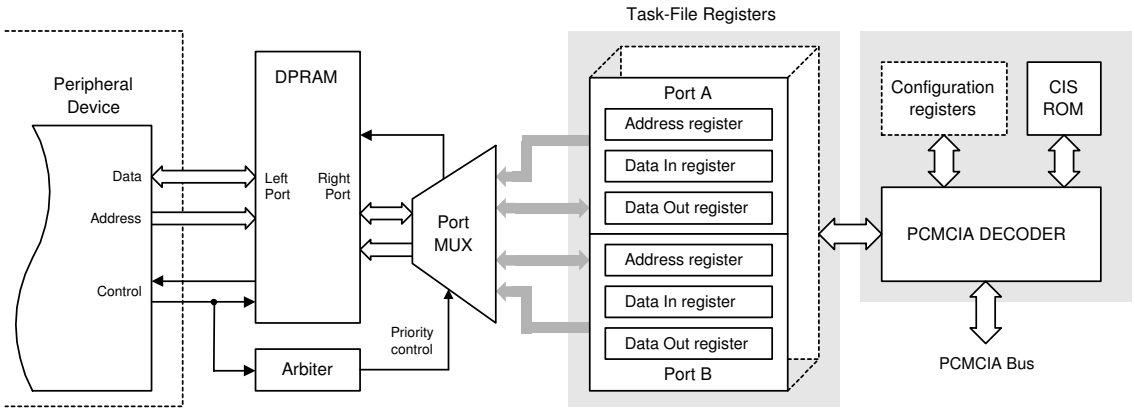

Figure 2. Simulation model and hardware platform integration.

mission and reception status. This interaction is described using a custom interface that is implemented using events and data. We distinguish the following types:

- *Timing and synchronization events*, which ensure that the procedures and state transitions occur precisely as defined in the protocol specifications.
- *Circuit control events*, which are related to the initialization and control of the data-pump circuits.
- *System signals*, which carry the several types of data and the parameters that the protocol units exchange with the system peripherals.

On the other hand, Simulink enables the design of sophisticated data and signal processing algorithms using complex build-in or custom functions, so that the complete data-pump can be modeled and verified. In particular, Stateflow and Simulink provide powerful capabilities for verification and validation of a model's behavior, such as a flexible debugger and support of state-transition animation. Therefore, it is easy to consider various scenarios and iterate until the Stateflow diagrams and the simulation results perform the desired behavior. Regarding the system's protocol FSMs, this development environment enables the validation of the protocol implementation at a higher level of abstraction and at the early stages in the development cycle.

3.1. Data exchange and synchronization

Figure 2 presents the data exchange concept between the simulation model and the hardware platform. As described in Section 2, this communication is based on the PCMCIA interface. In the high-level model, the blocks that are translated into circuit 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 demonstrated in Figure 2. This custom module provides the necessary circuits for the PCMCIA initialization and data transactions and for extending the available I/O memory space. A dual-port memory (DPRAM) that is accessed by both the simulation workspace and a custom peripheral device contains all necessary user and control information for the mixed-level model dataflow. In Figure 2 a case of two distinct ports that enable two different host applications to access the memory is presented. A MUX is used in order to decode the address and data buses. This configuration is useful for debugging and diagnostic purposes. In such a case a custom application collects data from the DPRAM in order to process the statistics and also provide a graphical representation. The two-port approach resolves invalid data read and write operations in concurrent memory access, due to the multi-cycle PCMCIA transaction.

This data-exchange concept enables the integration of the simulation model and the prototyping platform into a functional model, in which selected functions are executed in hardware. Stateflow and Simulink blocks of the high-level model may interact with their hardware/software counterparts in the prototyping platform, using the library functions that utilize the PCMCIA driver.

3.2. Embedded microcode development

The blocks of the system model are translated into circuits in the hardware platform using either automated code generation tools or custom user designs. The Mathworks provides support for automated code generation of the Stateflow diagrams and the Simulink blocks through Stateflow Coder and Real-Time Workshop. Selected families of Texas Instruments, Inc., Motorola, Inc. and Xilinx, Inc., devices are supported. The code produced is not highly op-

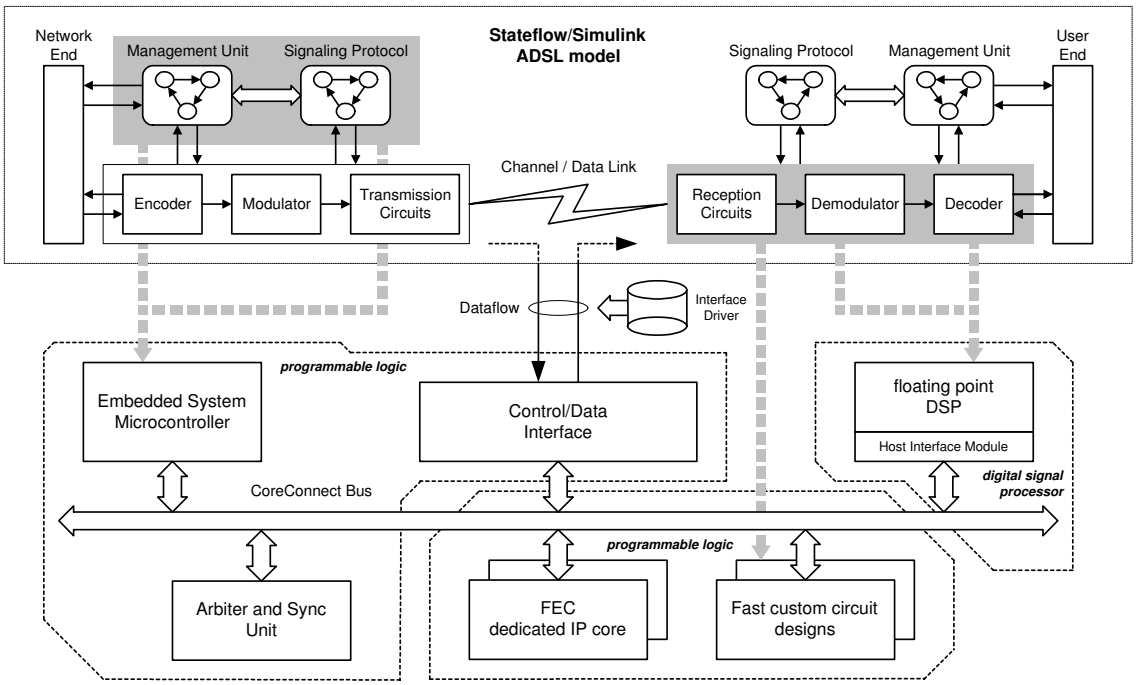

Figure 3. The modular environment for prototyping an ADSL communication system.

timized, however it is effective and can be embedded in the target platforms.

In demanding application, where custom design are preferred or new implementations are examined, the model blocks can be developed by the user using low-level programming languages (i.e assembly, C/C++) or hardware description languages (i.e VHDL).

4. The ADSL Modem Prototyping

In this section we present how the proposed methodology was used for prototyping an asymmetric digital subscriber line (ADSL) [2] data transmission system.

The ADSL systems perform multicarrier transmission over the local loop and support synchronous (STM) or asynchronous (ATM) data traffic. The ADSL data-pump incorporates advanced signal processing and coding techniques, i.e. Reed Solomon forward error correction (FEC), FFT-based modulation/demodulation, 4-D Trellis coding etc. The signal and data processing functions embedded in the ADSL data-pump are defined in Recommendation G.992.1 [3]. The establishment of the communication link between two far-end ADSL transceivers determines a specific set of initialization procedures including activation, transceiver training, channel analysis and parameters exchange. These procedures are accomplished using handshaking sequences between the two modems and comprise the signaling protocol for the asymmetric DSL devices. The detailed state sequences and transitions for the ADSL protocol are defined in Recommendations G.992.1 [3] and G.994.1 [4].

4.1. The prototyping environment

Figure 3 presents the prototyping environment for the ADSL system. Using standard and custom blocks of the Simulink and Stateflow tools, we developed a simulation system that models an end-to-end ADSL communication link. As described in Section 3, the data-pump functions were developed using build-in and/or user-defined algorithms embedded in Simulink blocks, while the subsystem's operation is controlled by state-machine modules developed using the Stateflow tool. The signaling protocol is also developed using hierarchical state-machines that control the transmission states of the transceiver models.

In order to build and verify the functionality of the simulation model, an appropriate interface is specified that describes the interaction between the signaling protocol state-machines and the signal processing units of the data-pump. This interface is implemented in the form of signals. We distinguish the following signal types:

- *Timing and synchronization signals*, which ensure that the procedures and state transitions occur precisely as defined in [3].

- *Protocol status signals*, which carry information about the protocol status and the initialization phases.

- *Data-pump control signals*, which are related to the initialization and control of the modem's signal processing circuits.

- *ADSL signals and messages*, which carry the several types of signals and the parameters that the modems exchange.

After the ADSL high-level model was verified and optimized using simulations, we used the top-down process presented in Figure 1, in order to map the model blocks into components of our prototype architecture and progressively build our system using the mixed-level design and test environment that consists of high-level model blocks and low-level circuit units. The integration between the simulation workspace and the hardware/software modules of the prototyping platform was realized using the interface described in Section 3.1.

4.2. Hardware architecture

In the current version, the prototyping platform is based on reprogrammable hardware and a floating-point DSP processor. In particular we use two Virtex-II FPGA devices with a total of 2 Million gates and a TMS320C6711 processor with 900 MFLOPS processing power.

Figure 3 presents the architecture of our prototype ADSL system. The embedded processor used in our implementation is the soft IP MicroBlaze core [5]. The hardware architecture is based on the CoreConnect bus [6], where the MicroBlaze is the bus master and all other peripherals are operating as On-chip Peripheral Bus (OPB) slave devices. The IBM's industry-standard OPB [6] is a 32-bit wide multi-master bus that is ideal for connecting custom and user-defined peripherals to the MicroBlaze processor core. This architecture provides the flexibility to develop a modular design and explore several configurations. Moreover this architecture satisfies the requirements of our application. Modern DSL systems are developed using programmable digital signal processing (DSP) platforms [7]. In particular, the functions embedded in the data-pump are implemented in high performance DSP cores or dedicated hardware. In Figure 3 the 6711 DSP is responsible for executing the data-pump functions, while dedicated hardware modules are used for the demanding forward error correction (FEC) encoding and decoding circuits and other custom circuits that require a speed-optimized implementation (i.e. interleaver/deinterleaver circuits, CRC check-sum engines).

Figure 3 shows the mapping of the simulation model blocks into the components of the prototype architecture. The protocol state-machines along with the modem's management logic are assigned to the MicroBlaze system controller, that supervises the modem's operation. The data and signal processing model units are mapped into software modules of the 6711 DSP or dedicated hardware circuits. The data exchange between the computer that hosts the simulation environment and the hardware platform is performed via the interface logic described in Section 3.1. The DPRAM interface module is implemented as a standard OPB slave peripheral.

The embedded dual port memory (see Figure 2) provides the storage area of data and control information exchanged between the simulation environment and the hardware platform, as well as a common-access memory space for the system components. In fact, the DPRAM provides the physical interface between the circuit modules translated from model blocks and their environment.

In Figure 4 we describe the general concept of the intercommunication between the simulation environment and the translated blocks of our prototype. Each system module is associated with a specific region in the DPRAM, which contains the interface signals between that module and its environment. In general, we distinguish the following interface signals:

- *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, however 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 and participate in the modem's dataflow.

Each discrete memory region is in fact the dedicated memory space of the corresponding peripheral of our prototype architecture, and therefore can be accessed by other system modules as the prototype integration proceeds. Figure 4 shows an example of the intercommunication between the analytical simulation model and three system components: the signaling protocol module executed in the system microcontroller, a software module implemented in the DSP processor and a hardware module implemented as dedicated RTL code. All modules consist of distinct components in the prototype architecture and communicate with the simulation model and each other over the OPB bus and their individual memory space. Although the entire memory space

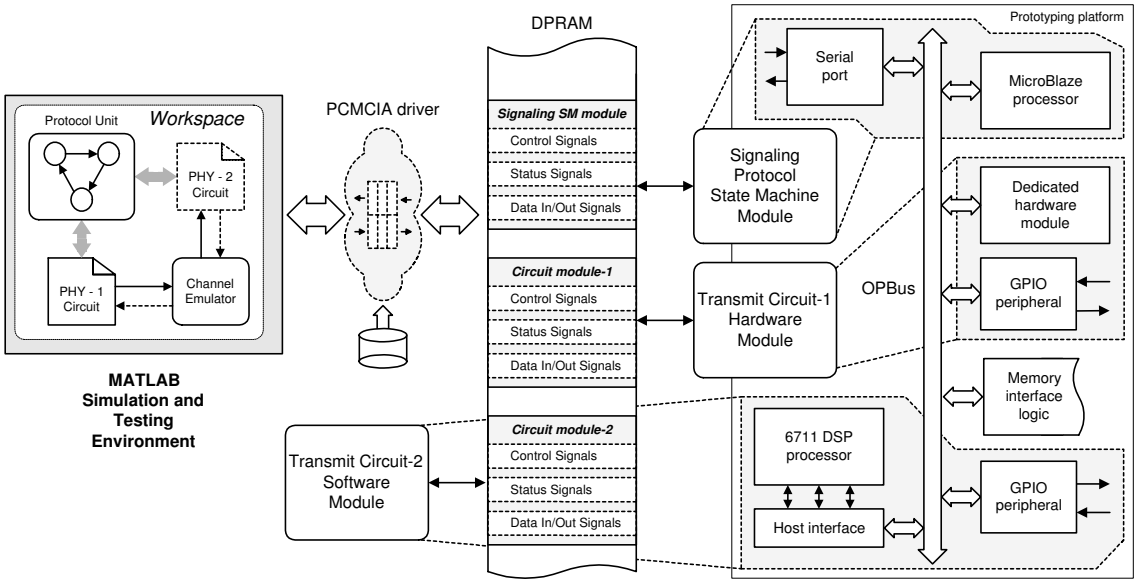

Figure 4. The intercommunication DPRAM organization.

behaves as a single memory module at the PCMCIA interface, it can be implemented as separate memories over the OPB bus according to the design complexity as the system is progressively build. We also observe that each module may include a general purpose peripheral module (i.e. I/O signals, serial port etc.) for internal testing and diagnostic purposes.

The mixed-level model that consists of the simulation blocks and the modules in the prototyping platform is a complete testing environment for each hardware component, since we can evaluate the functionality of each implementation, while the mixed-level model executes exhaustive testing of the ADSL specifications. After the verification of each component, we gradually extended our system by adding other hardware modules of the complete ADSL data-pump, either as standard OPB peripherals, or as standalone devices.

5. Conclusions

In this paper we have presented an efficient methodology for prototyping data communication systems using multi-domain Stateflow/Simulink models. We have presented the procedure for integrating a high-level simulation model with a hardware development platform and we have described how the analytical model is mapped and progressively translated into the components of the prototype architecture. The main advantage of the proposed approach it that the mixed-level system model provides a complete testing environment for each prototype module and enables the evaluation of each implementation in normal as well as corner cases of the system's operation. The implementation and testing steps provide a strong flexibility in the design process.

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.

References

[1] N. Papandreou, M.Varsamou, and T. Antonakopoulos, "xDSL Systems Prototyping using a Flexible Emulation Environment," *14th IEEE International Workshop on Rapid System Prototyping - RSP'2003*, June 2003, pp. 194-200.

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

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

[4] ITU, "Handshake Procedures for Digital Subscriber Line (DSL) Transceivers," ITU - G.994.1, March 1999.

[5] Xilinx Inc., MicroBlaze Hardware Reference Guide, Jan. 2002.

[6] IBM Inc., On-Chip Peripheral Bus: Architecture Specifications, ver 2.1, Apr. 2001.

[7] B. R. Wiese and J. S. Chow, "Programmable Implementations of xDSL Transceiver Systems," *IEEE Commun. Mag.*, vol. 50, pp. 114–119, May 2000.