From Basic Concept to Real-Time Implementation: Prototyping WCDMA Downlink Receiver Algorithms – A Case Study

M. Guillaud, A. Burg, L. Mailaender, B. Haller, M. Rupp, and E. Beck
Lucent Technologies, Wireless Research Laboratory
791 Holmdel-Keyport Road, Holmdel, NJ 07733-0400, USA
{maxime,burg,lm,bhaller1,rupp,ericbeck}@lucent.com

Abstract

In this paper we present an approach to rapid prototyping of advanced signal processing techniques for future wireless applications currently being adopted within Bell Labs Research. The aim of the "Bell Labs Algorithm Development and Evaluation" (BLADE) initiative is to devise a design framework specifically targeting the needs (and capabilities) of high-level algorithm designers (viz. communication engineers), which enables them to quickly "translate" a research idea into a working real-time system for practical experimentation purposes. The mixed DSP/FPGA implementation of a WCDMA testbed is used as an example to describe our initial experience with the proposed design methodology, which is based on a set of commercially available software tools and a platform consisting of off the shelf hardware modules.

1. Introduction

Future wireless systems aim to provide higher data rates, improved spectral efficiency and greater capacity. This can be achieved at the cost of increased signal processing complexity. The required algorithms are usually derived analytically from mathematical models based on many simplifying assumptions [1]. Following this, the original, "optimal" procedures nearly always need to be modified (i.e., re-engineered) in order to reduce their computational complexity. The performance of the resulting schemes is then typically obtained from computer simulations. Unfortunately, the employed simulation models rarely represent the actual system in sufficient detail, so that the designers are often left with a high degree of uncertainty regarding the real-world behavior of the developed solution. This is especially true when studying multiple antenna systems (such as BLAST [2]) which rely heavily on specific properties of the propagation environment. Therefore, researchers in wireless communications are now increasingly confronted with the problem of validating their ideas under realistic conditions and challenged with providing a proof of concept (i.e., they are required to demonstrate that a proposed novel scheme is practically feasible). Consequently, rapid prototyping is becoming essential to evaluate the true performance and accurately assess the implementation cost of new signal processing techniques for wireless applications prior to actual product development in order to minimize the risk of failure [3], [4], [5].

Traditionally, the people involved in generating these new algorithms are communication engineers, who have little knowledge of real-time system design, i.e., they typically possess limited experience in DSP programming and in writing VHDL code for an FPGA implementation. Our goal was to identify a set of tools and a design flow that would allow researchers with this type of background to implement their ideas on a flexible real-time prototyping platform. In this paper we present the proposed "Bell Labs Algorithm Development and Evaluation" (BLADE) methodology and describe how it was applied to building a WCDMA testbed, which was selected as an initial test case.

2. The BLADE Approach to Prototyping

Most algorithm designers start out by using interactive mathematical software packages such as Matlab, Maple, Mathematica or MathCAD to investigate a novel idea, because these tools provide a wealth of built-in data processing and analysis functions as well as extensive graphics capabilities. The next step typically consists of writing a floating-point C program—mainly to speed up the simulation runs. In principle, this C code could then be cross-compiled onto a digital signal processor (DSP), but often a single floating-point DSP is unable to meet the stringent real-time constraints imposed by the application. Therefore, it is common practice to employ multiple DSPs and/or map those parts of the system requiring very high speed processing onto hardware such as field-programmable gate arrays (FPGAs). These reconfigurable devices are especially well suited for prototyping since they allow to do multiple design iterations quickly. A hardware platform suitable for rapid prototyping of radio systems will therefore consist of

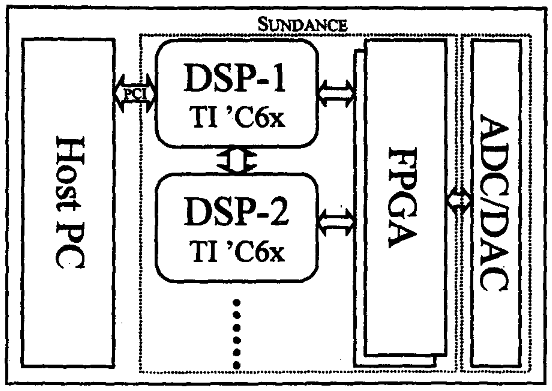

Figure 1. Basic BLADE hardware setup.

a mixture of DSPs and FPGAs, along with a C-based set of software tools which permit a smooth transition from a hardware independent, behavioral-level simulation code to a DSP and FPGA specific structural (i.e., register transfer level) code. Both the hardware and software components of the BLADE prototyping environment will be described in detail in the following two subsections.

2.1. Hardware Platform

A hardware platform appropriate for prototyping purposes must be able to accommodate a broad class of domain specific applications. Hence, the setup must be scalable, i.e., it has to be flexible, modular and extensible. We have chosen to use commercially available off the shelf (COTS) hardware components. The basis of our system is a development platform from SUNDANCE that consists of carrier boards with a PCI bus interface, each capable of hosting up to four TIM[1] modules. A variety of such hardware modules exist, including TI 'C6x series DSPs and XILINX Virtex series FPGAs. Each module is equipped with four asynchronous bidirectional 20 Mbyte/s links (= comm-ports), and some of them have additional 16-bit parallel interfaces (= SDB, Sundance Digital Bus) for synchronous transmission at speeds up to 200 Mbyte/s. Evaluation boards from ANALOG DEVICES are used to perform A/D and D/A conversion. **Figure 1** shows a typical setup for implementing a wideband radio communication system. The FPGAs contain the fixed, high sample rate front end processing, i.e., the regular data path (e.g., filters), which is both very computationally intensive and requires high I/O bandwidths [6], whereas the DSPs handle the symbol-level, data dependent processing. Functions such as the medium access controller (MAC), which operate on data packets are assigned to the Pentium processor in the host PC.

[1]Module standard originally specified by TI for multiprocessor systems based on their TMS320C4x DSP.

2.2. Software Tools and Design Flow

The main goal of the BLADE methodology is to provide the algorithm designer with a "painless" path to migrate from a high-level mathematical model to a bit-true, cycle-accurate description which can be compiled to run in real-time on a combination of DSPs and FPGAs. Ideally, a fully integrated design environment should support a top down development process, which allows incremental refinement of individual blocks. To achieve this, the suggested BLADE design methodology relies on the use of C as the primary description language, which is commensurate with the EDA industry's current general trend towards using C as a system-level design language [7]. This approach relieves the algorithm designer (i.e., the communication engineer) from the tedious tasks of having to write assembler code for the DSPs and/or use hardware description languages (HDLs), such as VHDL or Verilog, to implement those portions of the system which will be mapped onto FPGAs.

To support the C-based design paradigm, assist team-oriented development efforts as well as promote the concept of code reuse, THE MATHWORKS' Simulink was chosen as the system-level modeling and simulation tool. We have found that Simulink appeals far more to algorithm designers who currently use Matlab than other system-level design tools such a COSSAP or SPW. Furthermore, the latter tools are far more expensive and therefore often less accessible. In Simulink a design is represented as a (hierarchical) block diagram, with a piece of C or Matlab code associated with each block. Simulink comes with a large library of predefined blocks, which can be extended by users who write their own so-called S-functions. Since Simulink defines the block/S-function interfacing mechanism, code can easily be encapsulated and shared. Our method of system design in this environment is done as follows. Initially the system is modeled at the behavioral-level with all the signal processing being performed using floating-point arithmetic and predefined Simulink blocks (if available). Subsequently, those blocks that will be implemented on the hardware platform need to be gradually refined. The first refinement step consists of replacing all the floating-point data types and operations with finite-precision processing employing limited wordlengths (i.e., these Simulink blocks have to be replaced with custom S-functions). The fixed-point C++ class library provided by A|RT Library from FRONTIER DESIGN is used to model the bit-true system. In a second step, both the hardware structure as well as the scheduling of operations are introduced into the description of those blocks that will be mapped into FPGAs, resulting in a bit-true and cycle-accurate RTL-level model. To achieve this a special RTL-type C coding style (= RTL-C) is employed which can be automatically translated into synthesizable VHDL by the A|RT Builder from FRONTIER DESIGN. A|RT Builder

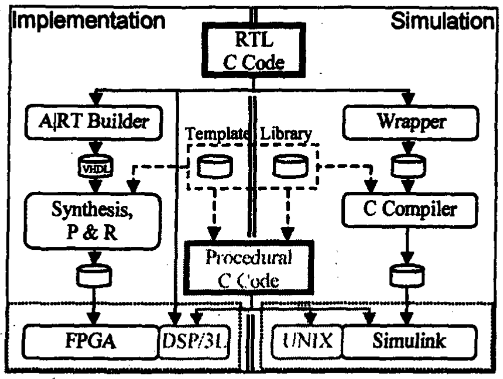

Figure 2. BLADE design flow.

does not perform any kind of behavioral synthesis, hence the development of an appropriate hardware architecture is left to the designer. It should be noted that the RTL-C code is ANSI C compliant and can hence still be used in the Simulink system simulation. Regarding blocks that are to be mapped into a DSP, the C code from the simulation can be compiled to run on either a floating-point DSP 'C67 or (with some modifications) on its fixed-point counterpart the 'C62. The 3L Diamond real-time operating system (RTOS) is employed to handle inter-processor communication, whereby each block runs as a separate task and comm-ports are utilized as links between devices. The user writes a configuration file to tell the RTOS how to assign the tasks to multiple DSPs and to specify the links between these processors.

The proposed BLADE design flow is outlined in **Figure 2**. The right side of the figure shows the simulation flow and the left side depicts the steps that lead to the implementation. A wrapper is automatically generated by a Perl script to create an S-function required to be able to simulate a block of floating/fixed-point or RTL-level C within Simulink. Non-block oriented functions that execute asynchronously with respect to the rest of the system can be co-simulated, running as separate tasks in a multitasking environment, in parallel to the cycle oriented Simulink engine. Interface functions and memories or other macros can be included from a template library.

3. WCDMA Downlink Design Example

In order to gain experience with the proposed design methodology, the implementation of a WCDMA downlink was chosen for a case study. The system complies with the 3GPP/UMTS physical layer specifications [8] in terms of the employed chip rate, spreading codes and scrambling sequences. The base station transmitter generates a primary synchronization signal, a fixed-rate pilot and a number of user signals with selectable rate and power level. The motivation behind putting this system onto the hardware platform was to evaluate the performance gain and assess the cost of employing a chip-level equalizer followed by a combined descrambler/despreader instead of a conventional rake receiver in a WCDMA mobile terminal. Since the multiple access interference (MAI) in a synchronous DS-CDMA downlink using orthogonal spreading codes is essentially due to the multipath, the in-cell MAI can be suppressed by channel equalization [9]. The intersymbol interference (ISI) caused by the delay spread especially becomes the limiting factor for high data rate users who are employing a short spreading factor (i.e., 4 or 8). The potential benefits of using an equalizer are an increase in capacity on the one hand as well as an improvement in throughput and greater coverage for high data rate services on the other. Since this system contains many of the basic building blocks found in typical broadband radio systems, this project also allowed us to develop a reusable library of frequently employed modules.

3.1. Transmitter

The transmitter hardware comprises a XCV400 400k gate Virtex FPGA and a 'C67 floating-point DSP. The DSP runs an ACELP voice encoder whose output is mapped onto the data stream of the user of interest. It also configures the FPGA when the system is turned on and controls the parameters of the transmitter which resides on the FPGA. Apart from the desired user's signal, up to 15 random data users can be turned on to act as interferers. Furthermore, a synchronization and a pilot signal are added on top of the user signals before the resulting signal is applied to a 25-tap RRC pulse shaping filter and subsequently digitally up-converted to an IF of 70 MHz, where it is bandlimited by a SAW filter. For experimentation in the laboratory the analog IF signal is sent through a programmable radio channel emulator after which Gaussian noise is added to it. Alternatively, the setup can be connected to an RF front end in order to perform "over the air" measurements.

3.2. Receiver

The receiver hardware consists of two 'C67 floating-point DSPs and a XCV1000 1000k gate Virtex FPGA. The following functions have been implemented:

- IF to baseband down-conversion
- slot/symbol/chip synchronization
- carrier frequency offset compensation
- channel estimation
- rake finger assignment and tracking
- 4-finger rake
- $(4\times)$8-tap polyphase equalizer + descrambler/despreader
- data interface/buffer and BER measurement

	Criterion	Function
FPGA	data path operations sampling/chip rate processing	IF down-conversion, filters correlators for sync. and channel estimation rake, equalizer
DSP	control flow complex and irregular operations complex data path operations lower rate processing with low I/O requirements floating-point or high precision fixed-point operations	synchronization and tracking control LS-based channel estimation (i.e., matrix-vector mult.) rake finger management equalizer coefficient computation (i.e., matrix inversion)

Table 1. DSP/FPGA partitioning of the receiver functions.

Table 1 lists the criteria that were applied for partitioning the receiver functions between the first DSP and the FPGA, and shows to which device the main blocks of the system have been allocated. All the blocks operating at the sampling rate (= 4× chip rate) are mapped onto the FPGA, because of the tremendous amount of computations that have to be performed in parallel. The relative size of the different receiver functions implemented on the FPGA is given in **Table 2**. The whole design consumes 87% of the XCV1000's logic resources (i.e., CLBs). In order to facilitate making quick design changes, without having to redo the time consuming VHDL synthesis and FPGA place and route over and over again (turnaround time approx. 6 hours!), almost all of the control operations are initially best realized on a DSP. Both the rake and the equalizer rely on channel impulse response (CIR) estimation. Originally, the CIR estimation procedure was implemented on one of the floating-point DSPs. A least-squares (LS) algorithm was chosen to determine the channel coefficients, which requires a large matrix-vector multiplication (512×64). Unfortunately, this computation is rather slow on the 'C67 (update rate approx. 50 Hz), so that the rake/equalizer is unable to track the channel, which leads to long error bursts. Therefore, an alternative solution based on a code matched filter was implemented on the FPGA. This method is able to update the coefficients 10 times as fast as before, and hence is capable of tracking the channel quickly enough in most practical situations. For the rake, four fingers are assigned to the four main peaks of the CIR. In the case of the equalizer, the tap coefficients of the FIR filter are computed according to the MMSE criterion based on the estimated CIR [9]. This computation relies on matrix inversion, which is very tedious to implement for high speed, real-time applications. The algorithm in [10] exploits the Toeplitz structure of the autocorrelation matrix to speed up the matrix inversion process. Further complexity reduction is achieved by breaking the output signal of the fourfold oversampled chip matched filter into four different phases, and implementing a separate equalizer for each phase. This so-called polyphase structure shows very good performance when compared to a straightforward equalizer realization. With this method, a 32-tap (i.e., 4-phases, each 8 chips/taps long) complex equalizer computation takes 1.18 ms on a 'C67 clocked at 166 MHz.

An ACELP voice decoder is implemented on the second floating-point DSP together with interface routines to the CDMA receiver and to an external audio codec. The decoder operates on a frame by frame basis and is embedded into the system using 3L Diamond to handle the data transfer. It was simulated independantly of the WCDMA receiver in Simulink and subsequently mapped (almost 1:1) onto the second DSP.

Digital down-converter	0.3%
Frequency offset estimation	1.8%
Rake-finger weighting & combining	6.4%
Timing reference	1.1%
Frequency offset compensation	5.9%
Chip matched filter	10.5%
Peak detection	1.2%
Polyphase equalizer (8 chips long)	29.1%
Correlators (4)	13.1%
Channel scope for LSE	1.3%
Channel estimation	26.9%
BER check	2.4%

Table 2. Relative size of different receiver functions within the FPGA.

3.3. Simulation and Measurement Results

A performance evaluation of the two schemes was done for a spreading factor of 32 using the outdoor-to-indoor channel model B defined in [11]. The in-cell interference level is gradually increased by turning on users (up to 15). The Gaussian noise level is negligible. **Figure 3** compares the measured performance of the rake with initial floating-point simulations. The gap of 1-1.5 dB between the two results is mainly due to the channel estimation latency, which is not taken into account in the simulations, where multiple fixed channels are used instead of a time-varying channel (with 5 Hz Doppler frequency). **Figure 4** shows preliminary results for the equalizer-based version of the receiver.

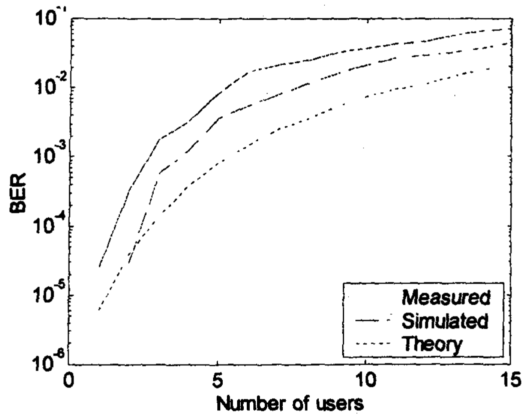

Figure 3. Performance of the 4-finger rake receiver.

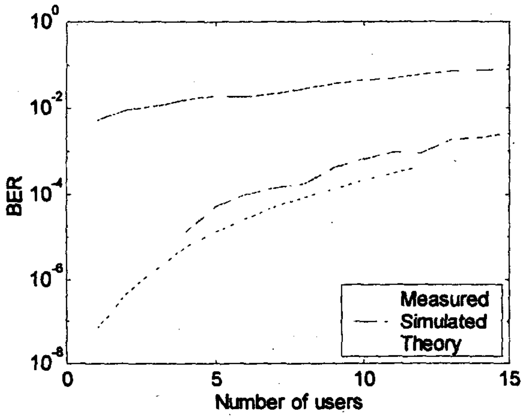

Figure 4. Performance of the equalizer-based receiver.

The slow equalizer update rate used in this initial setup is primarily responsible for the substantial performance loss of the current hardware implementation, which can be reduced by speeding up the coefficient computation for the equalizer.

4. Summary and Conclusions

We presented the BLADE design methodology for rapid prototyping of research ideas, which is tailored towards the needs and capabilities of system-level engineers. It uses C as the primary description language and Simulink as the simulation engine. This C-based approach is well suited for hardware/software co-design, allowing the designer to exploit the powerful capabilities of both DSPs and FPGAs, without having to be familiar with the details of the DSP architecture or the intricate structure of the FPGA, and ultimately avoiding the painstaking task of writing assembler and VHDL code. Evidently, this does not claim to be a design flow suitable for a production environment, since its does not lend itself to achieving performance optimized designs. However, the future incorporation of optimized blocks from vendor specific libraries, such as provided through the XILINX Core Generator, into our flow could improve performance substantially and further reduce design time and effort. A recent step in this direction was made through the introduction of the XILINX System Generator for Simulink [12]. A further limitation of the current FPGA flow is that only a single clock can be employed, which excludes the use of architectures based on time-sharing of hardware resources (e.g., bit-serial architectures) [13], a design style that is especially well suited for medium speed FPGA solutions.

References

[1] H. MEYR AND M. OERDER, "Systematic Derivation of Algorithms for Digital Receivers," in *Proc. 2nd Int. Workshop on DSP Techniques Applied to Space Comm.*, 1990.

[2] G. D. GOLDEN, ET AL., "Detection Algorithm and Initial Laboratory Results Using V-BLAST Space-Time Communication Architecture," *Electronics Letters*, 35(1):14–16, 1999.

[3] G. E. PRESCOTT AND S. TYLER, "Prototyping of Military Radio Systems Using Field Programmable Gate Arrays and DSP Microprocessors," in *Proc. ICSPAT '97*, 1997.

[4] M. RUPP, ET AL., "Rapid Prototyping for High Data Rate Wireless Local Loop," in *Proc. Asilomar Conf.*, 1999, Vol. 2, pp. 993–997.

[5] G. WRIGHT "The 'BiggaScale'/Berkeley Emulation Engine (BEE)," Project Presentation, 1999.

[6] R. BAINES, "The DSP Bottleneck," *IEEE Comm. Magazine*, 33(5):46–54, 1995.

[7] G. PROPHET, "System-Level Design Languages: To C or Not to C?" *EDN*, 44(21):135–146, 1999.

[8] 3GPP, *Physical Layer – General Description (Release 1999)*, Technical Report 3G TS 25.201 V3.0.2, 2000.

[9] I. GHAURI AND D. T. M. SLOCK, "Linear Receivers for the DS-CDMA Downlink Exploiting Orthogonality of Spreading Sequences," in *Proc. Asilomar Conf.*, 1998, Vol. 1, pp. 650–654.

[10] H. KRISHNA AND S. D. MORGERA, "The Levinson Recurrence and Fast Algorithms for Solving Toeplitz Systems of Linear Equations," *IEEE Trans. ASSP*, 35(6):839–847, 1987.

[11] ETSI, *Selection Procedures for the Choice of Radio Transmission Technologies of the UMTS*, Technical Report TR 101 112 V3.2.0, 1998.

[12] R. D. TURNEY, ET AL., "Modeling and Implementation of DSP FPGA Solutions," White Paper, 2000.

[13] XILINX INC., *FPGA Solutions for Next Generation Wireless Technology*, Workshop Notes.