ADSL Receiver Modeling - Final Report

Vikas Agarwal, Rajagopalan Desikan,

and Karthikeyan Sankaralingam

EE 382C - Embedded Software Systems

Spring 2000

May 10, 2000

Abstract

Our project models the receiver part of an ADSL (Asymmetric Digital Subscriber Line) modem as per the draft specifications of the G.Lite ADSL draft standard [1]. The receiver is modeled as a Synchronous Dataflow graph built using SDF stars in Ptolemy, and receives input from the channel model. The channel model and transmitter are also built in the SDF domain. The primary aim of the project is to build a system to simulate an ADSL modem. The primary application of ADSL modems is for high speed internet access and transmission of rich multimedia content like audio and video.

1 Introduction

In this project we model the receiver part of an ADSL (Asymmetric Digital Subscriber Line) modem. ADSL uses separate channels to transmit data upstream and downstream, and transmits data at a high rate downstream and a lower rate upstream by frequency division multiplexing. It has the ability to transfer data at up to 9 Mb/s downstream and up to 1 Mb/s upstream. [2]. Analog voice is transmitted at baseband frequencies and combined with the passband data transmission via a low-pass filter that is commonly called a "splitter". The fundamental concepts behind the operation of ADSL are 1) near-end crosstalk is reduced by having the upstream bit rate and bandwidth much less than the downstream bit rate, and 2) simultaneous transport of voice and data by transmitting data in a frequency band above voice telephony. The lower bit-rate upstream band is transmitted at lower frequency and the high bit-rate downstream band is transmitted at high frequencies. A guard band is 0Φnecessary to prevent POTS noise from interfering with the digital transmission. See Figure 1.

2 Objectives

The objective was to model the working of the receiver in an ADSL modem based on the specifications in the G.Lite draft standard [1]. We implemented the receiver using the SDF domain in Ptolemy. The secondary objective was to combine our receiver model with the channel and transmitter model to provide a simple yet complete model for an ADSL system. By integrating the transmitter, channel and receiver we planned to simulate the behavior of an ADSL modem.

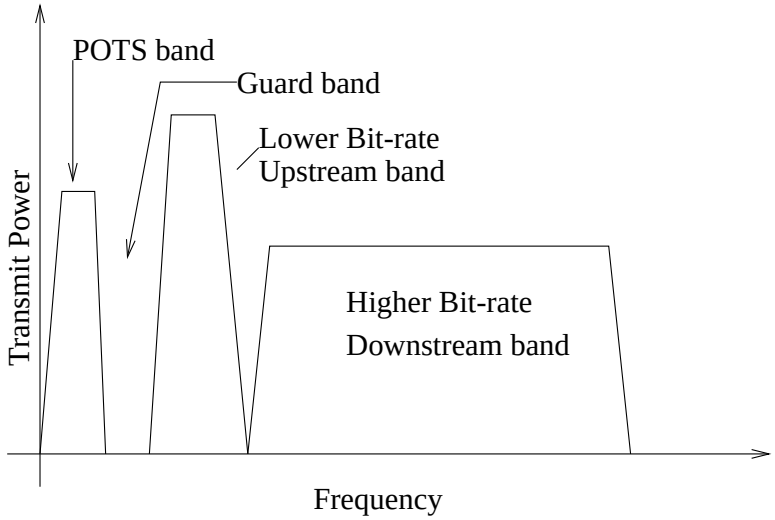

Figure 1: Frequency Division Multiplexing

3 Formal Modeling

The receiver is modeled as a synchronous dataflow graph [3]. The transmitter and channel are also modeled as SDF blocks and when connected together we simulate the entire ADSL modem as a single synchronous dataflow graph.

Different blocks in the receiver require different number of inputs and there are rate changes in some places. Figure 2 gives the different blocks in the receiver. Each of the blocks is implemented as a single SDF actor.

Such an SDF graph models the behavior of the receiver and is sufficient for simulation studies. Each of the actors, namely - the time domain equalizer (TEQ), FFT, Reed-Solomon Decoder, descrambler, and cyclic redundancy check (CRC) can be implemented in hardware as independent blocks. The TEQ is an FIR filter whose taps are determined during initialization. Hardware implementation of FFTs has been well documented and researched. There are several hardware implementation of Reed-Solomon decoders as they are extensively used in storage devices like

tape drive, compact disks , and DVDs. A Xilinx Reed-Solomon decoder core is available as an FPGA. There are several such commercial available hardware implementations. The Reed-Solomon code is a forward error correcting that can correct up to $R/2$ errors with R correction words [4]. The descrambler is implemented with a feedback shift register. Several CRC implementations exist in hardware using a shift register. The CRC is able to detect all single error bursts up to the number of bits in the CRC and most random errors [5].

The graph is scheduled using the Bhattacharrya loop scheduler in Ptolemy.

4 Implementation

For simulation, we assume that the receiver has performed all of the startup and handshaking with the transmitter. This makes the simplifying assumption that all of the channel parameters are known and do not change. We have narrowed the interpretation of the standard as needed to simplify the modeling. The parameters that we used in the receiver are given in the table below. All the values mentioned in the specification are supported.

Parameter	Values supported
Bitloading	Up to 16 bits on each of 126 channels
R (number of Reed-Solomon parity bytes)	0, 2, 4, 8, 16
S (Number of frames per RS codeword)	1, 2, 4, 8, 16
Interleave Depth	1, 2, 4, 8, 16

We specifically have not implemented:

1. Receiver Equalization updating: Once the taps of the equalizer are computed they are not updated dynamically. In a real modem the equalizer is adaptive because there may be changes in the transmission line or noise characteristics. [6] discusses methods to do this.

2. Transmitter adjustment: The bit distribution and symbol rates can be optimized dynamically based on the channel response and noise information in a real modem.

The reason for not implementing these features is that characteristics of the channel are determined once during initialization and passed as parameters to the SDF blocks. Since parameters of blocks in Ptolemy cannot change at runtime, we could not integrate these features in the current model.

Figure 2 is a block diagram of the receiver. Each of the blocks is explained in the remaining subsections.

4.1 Time Equalizer and Cyclic Prefix

The Time equalizer is implemented with a multitap FIR filter. The coefficients of the filter are determined during initialization. The Time equalizer is added to reduce the impulse response of the channel [7]. A Cyclic Prefix of length 16 is used to overcome the impulse response of the channel.

4.2 Synchronization Frame (SF)

The ADSL standard specifies that every 69th frame is a synchronization frame inserted by the transmitter. This frame contains no user data and is used by the receiver for synchronization. In our implementation we ignore the contents of this frame and simply drop every 69th frame. The SF block in our implementation comes right after the Cyclic Prefix block and takes an input of 69 frames and outputs the first 68 of those frames, ignoring the 69th frame. The sync frame reduces bandwidth marginally.

4.3 Bit decoder

The Bit decoder takes in the input of 256 complex values after the FFT and determines the actual bit values in each channel. Each channel carries different number of bits determined by the bit loading [8]. The Bit decoder interprets the real and imaginary part of the complex number as set of coordinates of a lookup table. We implement the bit decoder by interleaving the bits of input in the manner specified in the standard. Channel 64 carries the phase, which is fed to the channel.

4.4 Interleaver and Reed-Solomon(RS) Decode

The interleaver block undoes the interleaving done by the transmitter. The transmitter delays the ith byte of each frame by $(D-1)*i$ bytes. The bits are interleaved in this fashion to reduce the number of errors that can occur in a single frame. Most of the errors in the channel are likely to be burst errors and interleaving the bytes this way reduces the errors in a single RS codeword and spreads them in different codewords.

The Reed-Solomon Decoder in the specification is a 255 byte long codeword. It can correct up to $R/2$ errors with R parity bytes. The details of the decoder parameters can be found in the specification. We built the SDF decoder and encoder block in Ptolemy using source code developed by Rockliff [9]. The Reed-Solomon code is a forward error correcting code which operates continuously without any interrupts and it ensures constant delay on the data transfer which is useful for real-time applications. We implement the decoder using the iterative algorithm developed by Berlekamp [10].

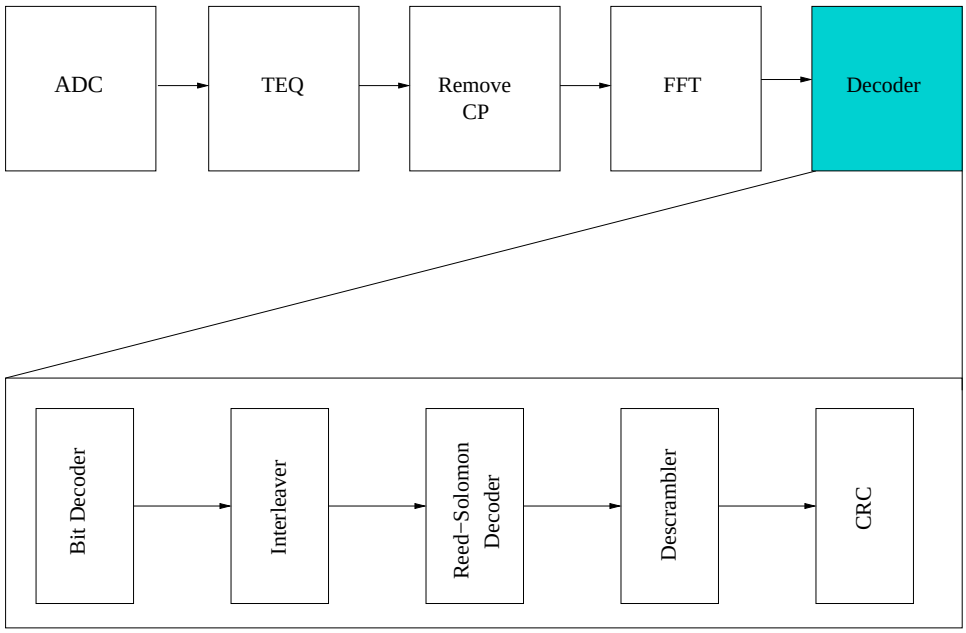

Figure 2: Block Diagram of the ADSL receiver

4.5 CRC

The first byte of the 0th data frame contains an 8 bit CRC of the previous super frame. We have implemented an 8 bit CRC block that takes in as input 68 data frames and outputs the 8bit CRC. This is compared with the transmitted CRC and an error is flagged if they differ.

5 Results

We have built an ADSL receiver which supports a subset of the parameters defined by the specification. We have tested the receiver stand alone with inputs fed from a Ptolemy source actor which reads data from a file. In the simulations, the data is decoded properly and errors are corrected, if they are within the range of errors tolerated ($< R/2$). If the CRC does not match we indicate an error and continue!

5.1 Buffer Requirements

Since we are modeling using SDF, we can determined memory requirements statically. The maximum buffer requirement on an arc is $69*256 = 17664$ float values. This is the buffer required to hold one super frame. The sync frame - last 256 samples are deleted and the remaining 68 frames are processed.

The different values of R and S do not make any difference in the simulation times. The different values of D (2 to 16) do not affect simulation time but increase the buffer requirement linearly on the arc from the interleaver to the RS decode block. The worst case buffer requirement will be $(16-1)*126*16 = 30240$ bits. (16 bits on each of 126 channels - theoretical maximum for bit loading).

6 Conclusions

The main impact of our project is that it provides a simple yet complete model of an ADSL system in conjunction with the transmitter and channel model developed by the other groups. Accurate simulation of the complete systems helps in reducing the time required for developing the corresponding hardware. Future work in this project will be to integrate the 3 models and run simulations of how the channel affects transmission and model upstream communication.

The two contributions of the project are a complete ADSL receiver and a few generic Ptolemy blocks which will be useful in modeling other systems - RS encoder and decoder, and an 8 bit CRC.

References

[1] "Splitterless asymmetric digital subscriber line (ADSL) transceivers - draft recommendation G.992.2." Feb. 1999.

[2] K. Maxwell, "Asymmetric digital subscriber line: Interim technology for the next forty years," *IEEE Communications Magazine*, vol. 3401, pp. 100–106, Oct. 1996.

[3] P. K. M. Shuvra S. Bhattacharyya and E. A. Lee, *Software Synthesis from Dataflow Graphs.* Kluwer Academic Publishers, 1996.

[4] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Indust. Appl. Math.*, vol. 8, pp. 300–304, 1960.

[5] F. Halsal, *Data Communication, Computer Networks and Open Systems, Fourth Edition.* Addison-Wesley Publishing Company Inc., 1996.

[6] P. J. S. Thomas Starr, John M. Cioffi, *Understanding Digital Subscriber Line Technology.* Prentice Hall Inc., 1999.

[7] P. J. W. Melsa, R. C. Younce, and C. E. Rohrs, "Impulse response shortening for discrete multitone transceiver," *IEEE Transactions on Communications*, vol. 44, pp. 1662–1672, Dec. 1996.

[8] P. S. Chow, J. M. Cioffi, and J. A. C. Bingham, "A practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels," *IEEE Journal of Communication*, vol. 43, pp. 773–775, Feb./Mar./Apr 1995.

[9] S. Rockliff", "http://imailab-www.iis.u-tokyo.ac.jp/r̃obert/,"

[10] R. E. Blahut, *Theory and Practice of Error Control Codes.* Addison-Wesley Publishing Company Inc., 1983.