ADSL Transmitter Modeling and Simulation

Department of Electrical and Computer Engineering
University of Texas at Austin

Kripa Venkatachalam

Qiu Wu

EE382C: Embedded Software Systems

May 10, 2000

Abstract

Asymmetric Digital Subscriber Line (ADSL) offers high-speed data transmission over existing telephone lines (POTS). It is gaining in prominence because of its features, which provide for optimal bandwidth usage, low bit error rate and ease in deployment. Its computational needs are realizable by the high-power DSPs available today. We have implemented a configurable and extensible simulation of an ADSL transmitter in Ptolemy environment. This simulation would allow tradeoff analysis. This simulation can also be used for evaluating the feasibility of single processor solution. This paper discusses the capabilities and limitations of the simulation developed by us. Computational and memory requirements have been summarized and analyzed.

1. Introduction

Asymmetric Digital Subscribe Line (ADSL) is used to deliver high-rate digital data over existing telephone lines. ADSL facilitates use of normal telephone services, ISDN and high-speed data transmission on the same line. ADSL has been finding widespread application because of its features that provide for low bit error rate ($10e^{-7}$), optimal channel bandwidth usage, and ease in deployment [1]. ADSL allows data rates of the order of 1 Mbps downstream and about 100 Kbps upstream. ETSI and ANSI have defined the three different ADSL standards.

One of the significant problems in ADSL modem design is the inherent complexity of the system. This project was aimed at developing a high level abstraction of the design of an ADSL transmitter. A subset of ANSI G.lite standard [2] for ADSL was implemented. The standard provides lot of flexibility to the user and most of the parameters are programmable. Implementation details are specified only where necessary. The primary contribution of this project is a highly configurable and extensible simulation platform for building an ADSL transmitter.

Applications

- This simulation platform would help analyze the performance of an ADSL transmitter in terms of data throughput and reliability.
- This simulation can also be used to provide an estimate of the computational and memory requirements of the design, which could help in deciding the feasibility of a single chip solution.

- This platform, in conjunction with the receiver and channel model, could provide tradeoff analysis of Noise Vs Bit error rate and Noise Vs Bit rate.

This report describes the simulation developed, identifying both its capabilities and limitations. First, we summarize the operation of an ADSL transmitter, as specified in the ANSI G.lite standard [2], emphasizing the programmable parameters. Then, we describe the model developed. We discuss the system metrics of a reference design in terms of memory, simulation time and computation. We conclude by discussing the limitations of our simulation and possible enhancements to the present work.

2. ADSL Transmitter

Modulation Scheme

Discrete multi-tone (DMT) [3][6][9] modulation is chosen as an ANSI standard line code for ADSL system. In a *N* sub-channel DMT system, modulation is implemented with a *2N* size inverse fast Fourier transform (IFFT).

Data packet description

The input bitstream is grouped and processed in terms of structures known as superframes [2]. Each superframe is composed of 68 data frames, which are encoded and modulated into DMT symbols and 1 sync frame, which carries no user or overhead bit-level data and is inserted by modulator to establish superframe boundaries.

Transmitter Operation

ADSL G.lite standard [2] specifies 128 sub channels of 4KHz bandwidth each. The operational details of a transmitter are described below. The description is intended to provide the necessary background for understanding the model implementation and is not

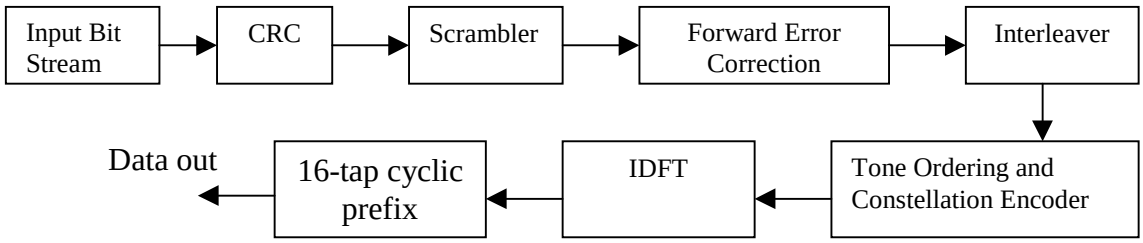

Figure 1: Functional block diagram of an ADSL Transmitter

meant to provide a functional overview of the transmitter [5][12]. Figure 1 shows a block diagram of an ADSL transmitter.

Cyclic Redundancy Code (CRC): 8-bit CRC is computed and added to the first byte of the frame 0 of the next super frame. The crc bits are computed from the *k* message bits using the equation:

$$crc(D) = M(D)\, D^8 \ \ modulo\ G(D)$$

where:

$$G(D) \ = D^8 + D^4 + D^3 + D^2 + 1$$

$$M(D) \ = m_0 D^{k-1} + m_1 D^{k-2} + \ldots\ldots + m_{k-2} D + m_{k-1}$$

$$crc(D) = c_0 D^7 + c_1 D^6 + \ldots\ldots + c_6 D + c_7$$

Scrambler: The binary data stream is scrambled using the following algorithm:

$$d_n = D_n \oplus d_{n-18} \oplus d_{n-23}$$

where D_n is the n^{th} input to the scrambler and d_n is the n^{th} output of the scrambler.

Forward error correction (FEC): Reed Solomon encoder is implemented for forward error correction. The Reed Solomon encoder appends *R* bytes of Reed Solomon FEC redundancy bytes to every *S* data frames. *R* and *S* are programmable. *R* could take on values from {0,4,8,16} and *S* could take values on values from {1,2,4,8,16}. *R/S* has to be

an integer. *R* redundancy bytes $c_0, c_1,\ldots,c_{R-1}$ are appended to *KxS* (where *K* is the number of message bytes per dataframe) message bytes using the following rule:

$$C(D) = M(D)\, D^R \text{ modulo } G(D)$$

Where, *G(D)* is the generator polynomial and *M(D)* is the message polynomial.

Interleaver: The Reed Solomon codewords are convolutionally interleaved. The interleaver depth can be 1,2,4,8 or 16. Bytes are interleaved according to the rule:

Byte B_I (with index *I*) is delayed by $(D-1) * I$ bytes, where *D* is the interleave depth.

Tone ordering and Constellation encoder: Based on the bit allocation table computed during initialization (dependent on the channel conditions), bits are allocated to the 128 sub channels. Bits allocated to each channel are modulated as quadrature amplitude modulated (QAM) [10] symbols. The number of bits per channels can be between 2 and 15 and the maximum value is programmable between 8 and 15. Specific rules for constellation encoding are defined in the standard [2]. 256 point IFFT and 16 point cyclic prefix are used.

3. Implementation and Analysis

Modeling: The model was developed in Ptolemy [4]. The scope of the model defines the operational specifications of the blocks of the transmitter described above and models the downstream dataflow function of the transmitter without handshake with the receiver. Restricting the scope of the model to describe dataflow allows the blocks to be mapped on to synchronous dataflow (SDF) [8][11] stars. Including the initialization model would have necessitated the use of dynamic dataflow (DDF) [8] domain for modeling, which would slow down the simulation. Table 1 shows the number of tokens consumed and produced by the actors. All tokens are represented in bits unless specified otherwise.

Actor	Tokens Consumed	Tokens Produced
CRC	69*(Total Number of bits* – *R**8)	69*(Total Number of bits – *R** 8)
Scrambler	1	1
FEC	*S**(Total Number of bits– *R**8)	*S**(Total Number of bits)
Interleaver	8	8
QAM	Total Number of bits	256 complex numbers
IFFT	256 complex numbers	256 real numbers
Cyclic Prefix	256 real values	272 real values

*As determined after initialization.

Table 1: Model of the transmitter specifying the number of tokens consumed and produced by the actors.

Schedule and Mapping to a multiprocessor system: As can be seen from table 1, CRC block operates in groups of bytes that are a multiple of 69 and FEC block operates in groups of bytes that are a multiple of *S*. The schedule shown below in Figure 2 displays two groups of actors: the first group that is executed *S* times followed by the second group that is executed 69 times. This implies that the data output is not continuous. *S*69* DMT symbols are calculated followed by a silence period, which is considerable as we shall see in the next section. So this schedule does not yield itself to uniform data rate output and sampling as is needed by the ADSL system. Mapping the system to a two-processor system will eliminate this problem. CRC could be mapped on to a general-purpose processor and the rest of the blocks to a DSP. Since the CRC occupies most computational requirement, as will be shown in the next section, the two processors would be sharing the load uniformly and would be functioning optimally too.

```
{
  { scheduler "Bhattacharyya's Loop SDF Scheduler" }
{ repeat 4 {
  { repeat 8694 {
    { fire modem.ConstInt1 }
    { fire modem.IntToBits2 }
  } }
  { fire modem.Crc1 }
} }
{ repeat 69 {
  { repeat 504 {
    { repeat 8 {
      { fire modem.Scrambler1 }
    }
    { fire modem.BitsToInt1 }
  } }
  { fire modem.Reedencode1 }
  { repeat 4 {
    { repeat 127 {
      { fire modem.Interleaver1 }
      { fire modem.IntToBits1 }
    } }
    { fire modem.Qam1 }
    { fire modem.Ifft1 }
    { fire modem.Cyclicprefix1 }
    { repeat 272 {
      { fire modem.XMgraph.input=11 }
    }
  } }
} }
}
```

Figure 2: Schedule generated with R = 4 and S = 4

4. System Metrics

The system was analyzed for simulation time, computational load distribution between its actors, buffer requirement and computational complexity. All data has been specified for a reference design of 128 channels, 8 bits per channel, *R=4, S=4, D=2* system.

Number of bits to be transmitted	**Simulation time in secs**	**Average load in m/c**
280416 (Avg. load: 8 bits/ channel)	11.5643	4.04
560832(Worst case :16bits/ channel)	22.902	4.04
140208 bits (Low load:4 bits/ channel)	6.3228	4.04

Table 2: Number of bits to be transmitted Vs Simulation time

SDF Actors	**Percentage simulation time**
CRC	45.05

Scrambler	10.17
Reed Solomon Encoder	15.56
Interleaver	9.32
Constellation Encoder	9.8
IFFT	10.08

Table 3: Percentage simulation time for each SDF actor

SDF Actors	**No. of multiplications**	**No. of additions**	**No. of bitwise operations**
CRC	0	70656	211968
Scrambler	0	0	17
Interleaver	1	0	0
QAM	0	128	1024
FEC	0	4096	4096

Table 4: SDF Actors and their computational complexities

Maximum buffer requirement = 280416.

The CRC block implemented operates on bits and is targeted for mapping on to hardware. A faster algorithm that would operate on bytes could be used instead, which would not be to hardware friendly. As can be seen from Table 4, number of multiplications is reduced by maximizing the use of shift operations. IFFT block of Ptolemy was used and its computational complexity was not analyzed.

Conclusions and possible enhancements

A data flow model of transmitter was developed in the SDF domain. The simulation is configurable, extensible and provides for tradeoff analysis. System metrics in terms of computational load, complexity, buffer requirement and simulation time were computed.

Integrating the transmitter with the receiver and channel model and integrating the initialization model to the dataflow model would certainly prove to be interesting enhancements.

References:

1. ANSI T1.412-1995, "Network and customer installation interfaces: Asymmetric digital subscriber line (ADSL) metallic interface."
2. ITU-T G.992.2, "Splitterless asymmetric digital subscriber line (ADSL) transceivers."
3. J. A. C. Bingham, "Multi-carrier Modulation for Data Transmission: An idea Whose Time has Come," *IEEE Communication Magazine*, vol. 28, no.5, pp. 5-14, May 1990.
4. J. Buck, S. Ha, E. A. Lee and D.G. Masserschimitt, " Ptolemy: A framework for Simulating and Prototyping Heterogeneous Systems," International Journal of Computer Simulation, special issue on Simulation Software Development, vol. 4, 1994.
5. W. Y. Chen, *DSL Simulation Techniques and Standards Development for Digital Subscriber Line System.*, Macmillan Technical Publishing, 1998.
6. J. S. Chow, "A Discrete Multi-tone Transceiver System for HDSL Applications," *IEEE Transaction on Selected Areas in Communications*, vol. 9, no.6, Aug. 1991.
7. J. M. Cioffi, "A Multicarrier Primer," *Stanford University/Amati T1E1 contribution*, I1E1.4/91-157(November 1991)
8. B. L. Evans, "Class notes for ee382c: Embedded software systems, spring 2000," The University of Texas at Austin.
9. I. Kalet, "The Multi-tone Channel," *IEEE Transactions on Communications*, vol 37, no.2, pp: 119-124, Feb. 1989.
10. E. A. Lee and D.G. Masserschimitt, *Digital Communication*, 2nd Edition, Kluwer Academic Publishers, ISBN 0-7923-9391-0, 1994.
11. E. A. Lee and D.G. Masserschimitt, " Synchronous Dataflow," Proceedings of the IEEE, vol. 75, no. 9, 1987.
12. T. Starr, J. M. Cioffi, and P. J. Silverman, *Understanding Digital Subscriber Line Technology*., Prentice-Hall PTR, 1999.