

# Fixed Point DSP Implementation of Low-Density Parity Check Codes

Tejas Bhatt, Krishna Narayanan, Nasser Kehtarnavaz

DSPLab, Department of Electrical Engineering  
Texas A&M University  
College Station, TX 77843-3128  
email: [tejas@ee.tamu.edu](mailto:tejas@ee.tamu.edu)

**Abstract:** It has been shown earlier and rediscovered recently that Low-Density Parity Check Codes can achieve bit error rate near to Shannon limit. They can be decoded using soft decision iterative decoding scheme. As low cost, high performance DSPs are widespread, DSP implementation for LDPC decoder is a viable option. Floating point implementation has higher costs and soft decision iterative decoding is considered. Various optimal and sub-optimal implementations of the algorithm are considered. Various algorithms are compared for performance loss and low complexity tradeoff. It is shown that there is no performance loss as compared to floating point software approach. A lower complexity and higher speed implementation of the LDPC decoder can thus be achieved using fixed point DSP implementation.

high speed decoding of this algorithm can be achieved by implementing it on the fixed-point DSP.

In this paper, we propose implementation of iterative decoding of LDPC codes on fixed point DSP. In particular, the algorithm is implemented on the TI fixed-point DSP, TMS320C6201. The decoding is done for an AWGN channel using fixed-point approach and comparison with floating point software implementation is given. We also present the comparison between various sub-optimal implementations of the algorithm and study the error rate vs. complexity tradeoff. Results described in later sections show almost no performance loss due to fixed-point approach. The gain is increased speed of decoding and lower complexity. This makes the code a good competitor for uplink in mobile communications.

## I. INTRODUCTION

Low-Density Parity-Check (LDPC) codes are receiving much attention because of their near Shannon limit performance [1]. It has been shown in some of the recent work that their bit error rate is comparable to turbo codes and sometimes better. Although these codes were first proposed by Gallager [2] in 1963, they were forgotten for almost three decades until they were rediscovered in [1]. LDPC codes are linear block codes with a sparse parity check matrix. They can be sub-optimally decoded with an iterative algorithm like turbo codes. Decoding complexity for LDPC codes can be lower than that of turbo codes. While, much research has been done for the analysis of these codes, we are unaware of any fixed point DSP implementations of decoding schemes for these codes.

The sum-product algorithm proposed by Gallager in [2] is a computationally intensive algorithm. In depth analysis of the same is available in the literature [2, 3]. Various low complexity implementations of the algorithm are also widespread. The low complexity and

The paper is organized as follows. In section II the system model and decoding algorithm is discussed. The code structure and construction are explained in detail. The specifics of fixed point DSP implementation are described in section III. We present various techniques to lower the complexity and memory requirements. In section IV, we present the simulation results for both floating-point software implementation and fixed-point DSP implementation. In section V, a brief summary of the work is presented.

## II. SYSTEM MODEL

System model we consider is shown in Fig. 1. The information bit sequence is encoded using a rate  $1/2$  LDPC encoder and the coded output is transmitted over an AWGN channel. At the receiver end, output of matched filter are given to soft decision iterative

decoder, the output of which is estimated information sequence.

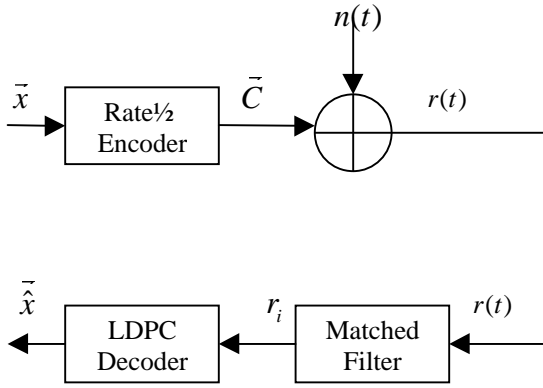


Fig.1–System Model

### A. Construction of LDPC Codes

Low-Density Parity-Check codes can be specified by a sparse parity check matrix  $H$  having 1's in very few positions. LDPC codes are linear  $(N, K)$  block codes with  $K$  information bits mapped to a codeword of block length  $N$ . The parity check matrix for the code has exactly 'j' 1's in each column and 'k' 1's in each row. Irregular Gallager codes having different number of 1's in different columns and rows are also possible. For any two columns, positions of 1's cannot overlap in more than one row position.

Any LDPC code can be represented as  $\{N, j, k\}$  code. If  $H$  has fixed number of 1's in each column only we can represent the code as  $\{N, j, \_ \}$  code. Similarly, if  $H$  has fixed number of 1's in each row only, we can represent the code as  $\{N, \_, k\}$  code. When  $j \geq 3$  and  $k > j$ , bit error rate for the code decreases exponentially with root of code length  $N$  [2].

In particular, we have considered rate 1/2 LDPC code with code length  $N=200$  and  $j=3$ , i.e.  $(200, 100)$  code of type  $\{200, 3, \_ \}$ . The algorithm can be easily extended for both regular and irregular LDPC codes of any arbitrary length.

### B. Decoding Algorithm

The iterative decoding algorithm is same as one described by [3]. Let  $r_i$  be information available from the channel for the  $i$ th coded bit position and  $\sigma^2$  be the variance of the white gaussian noise. Let the dimension

of the code be  $(N, K)$ . Let  $R_j$  be the set of positions of columns having 1's in  $j$ th row and  $L_i$  be the set of the positions of the row having 1's in the  $i$ th column, i.e.  $R_j = \{ i | H_{j,i} = 1 \}$  and  $L_i = \{ j | H_{j,i} = 1 \}$ ,  $\forall i \& j$ . Let,  $\rho_j$  = Number of elements in  $R_j$  and  $v_i$  = Number of elements in  $L_i$

The log likelihood ratio is given as

$$L = \sigma \ln \left( \frac{\Pr(C_i = -1)}{\Pr(C_i = +1)} \right) \quad (1)$$

Then, addition of two log likelihood ratios  $a$  and  $b$  can be defined as,

$$a \oplus b = \sigma \ln \left( \frac{e^a + e^b}{1 + e^{a+b}} \right) \quad (2)$$

Subtraction of two log likelihood ratios  $a$  and  $b$  can be defined as

$$a \ominus b = \sigma \ln \left( \frac{e^a - e^b}{1 - e^{a+b}} \right) \quad (3)$$

The decoding algorithm is now presented. It is same as given by [3], but some modifications have been made for the ease of implementations.

**Step:1** : Initialization

$$L_j^{(0)}(x_i) = \frac{2}{\sigma} r_i, \quad \forall j \in L_i, \quad i=0,1,2,\dots,N-1$$

**Step:2** : For  $q=1,2,\dots,Q$

*Part A*: For  $j=0,1,2,\dots,K-1$

$$tmp = L_j^{(q-1)}(x_{R_{j,0}}) \oplus \dots \oplus L_j^{(q-1)}(x_{R_{j,\rho_j-1}}) \quad (4)$$

for  $i \leq \rho_j - 1$ :

$$L_e^{(q)}(x_{R_{j,i}}) = tmp \ominus L_j^{(q-1)}(x_{R_{j,i}}) \quad (5)$$

*Part B*: For  $i=0,1,2,\dots,N-1$

$$L_j^{(q)}(x_i) = \frac{2}{\sigma} r_i + \sum_{l \in C_i, l \neq j} L_e^{(q)}(x_i) \quad (6)$$

Part C :Soft Output:

$$L(x_i) = \frac{2}{\sigma^2} r_i + \sum_{j \in C_i} L_{e_j}^{(q)}(x_i) \quad (7)$$

Part D:Hard Decision:

For  $i=1,2, \dots, N-1$ ,

If  $L(x_i) > 0$ ,  $\hat{x}_i = 1$ , else  $\hat{x}_i = 0$ .

Part E:Stopping Criteria:

If,  $\bar{S} = \hat{x}^T \bar{H} = \bar{0}$  or  $q > Q$ , stop, else continue.

### III. DSP IMPLEMENTATION

In this section the DSP implementation details are presented. The scope of the fixed point DSP implementation is primarily related to  $j$  additions and subtractions of the log likelihood ratios per iteration. Due to the higher cost associated with the floating point implementation, the fixed point implementation is considered here. The iterative decoding scheme for LDPC is implemented on a high performance DSP, namely the TIDSP TMS320C6201@. Such an implementation demands some approximations that are presented below.

The addition and subtraction operation for the log likelihood ratio is done using uniform quantization and lookup table. As per the following equation,

$$\log(e^a + e^b) = \max(a, b) + \log(1 + e^{-|a-b|}) \quad (8)$$

the second term on the right hand side dies down quickly and can be ignored after a large enough  $|a-b|$  value.

The addition operation is efficiently implemented by using lookup table with almost no performance loss if the interval size is sufficiently small.

$$a \oplus b = \max(a, b) + \log(1 + e^{-|a-b|}) - \max(a + b, 0) - \log(1 + e^{-|a-b|}) \quad (9)$$

The implementation of equation (9) requires one lookup table for the addition operation. The subtraction operation can also be implemented using the lookup

table approach. There will be two different cases for the subtraction operation as follows:

$$a \ominus b = a + \log(1 - e^{-|a-b|}) - \log(1 - e^{-|a+b|}), \text{ if } (a + b) < 0 \quad (10)$$

and

$$a \ominus b = -a + \log(1 - e^{-|a-b|}) - \log(1 - e^{-|a+b|}), \text{ if } (a + b) > 0 \quad (11)$$

The second term on the right hand side of equation (10) and (11) is again approximated using a lookup table. Selecting an appropriate interval size, one can clip the values of the resulting extrinsic information. The performance loss due to this is negligible. Converting the log, exponents and division operations via table lookups speeds up the decoding algorithm and reduces the complexity. A much more involved coding effort for log and exponent routines is thus avoided.

As far as memory requirements are concerned, the sparse matrix implementation, using  $C_i$  and  $R_j$  for the extrinsic information calculations, saves a considerable amount of memory by allowing the entire codeword to reside on the 128K bytes on-chip memory. The use of on-chip memory avoids accesses to slower off-chip memory.

A so called Q-format representation consisting of 16 bits for both integer and fractional parts is employed for the fixed point implementation. Overflow is avoided via clipping. Reduced complexity is achieved by avoiding division or multiplication operation even for the lookups. Interval size is specified as the power of two to replace multiplications with shift operations. The parallelism features of this DSP chip is fully exploited to avoid any wait state for memory loading.

We intend to consider the hyperbolic tangent implementation of the iterative decoding algorithm and report the results in the final version of the paper. This can reduce the number of lookups and thus can make the algorithm more efficient. For improved bitrate we plan to examine the sub-optimal approach of the max log mapping. This can improve the speed by removing the necessity of memory loads for the lookups and can also save the memory storage requirements. This lowered complexity and higher speed, though come at the cost of increased SNR for same bit error rate.

## IV. RESULTS

In this section we present the results obtained for {200,3,1} LDPC code over AWGN channel. Fig. 2 shows the graph of bit error rate vs SNR in dB for both fixed point DSP and floating point software implementations. The results obtained considering same number of iterations for both implementations.

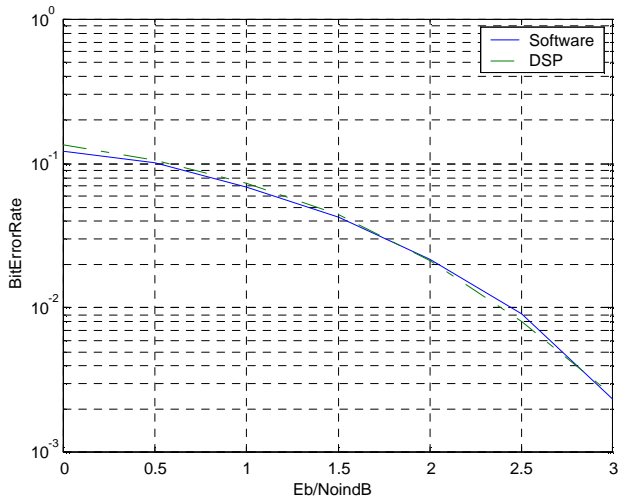


Fig. 2 BER Comparison

So far the above fixed point implementation is done in approx. 30,000 instruction cycles per iteration. Considering five iterations on average, 150,000 instruction cycles are needed. With a clock rate of 200 MHz or 5 ns cycle time, a bitrate of 133.33 kbps has been obtained at this point in time.

The hyperbolic tangent and max log map are expected to lead to higher bit rates. We are in the process of the fixed-point implementations of these techniques. Since max log map results in performance degradation, we will be examining the bit error rate vs complexity tradeoff. The outcome will be reported in the final version of the paper.

## V. SUMMARY

In this paper we look at fixed point DSP implementation of iterative decoding algorithm of LDPC codes. An increase in bitrate is obtained by efficient implementation. A superior bitrate can be achieved by considering hyperbolic tangent and sub-optimal implementation like max log map. Since this algorithm reduces to lower complexity implementation and

considering near Shannon limit performance of LDPC codes, fixed point DSP implementation looks a promising approach, making these codes a viable option for data transmission through wireless channel.

## REFERENCES

1. D.J.C. MacKay and R.M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes", IEEE Electronics Letters, vol. 32, no. 18 pp. 1645-1646, Aug 1996.
2. R.G. Gallager, "Low-Density Parity-Check Codes", M.I.T. Press, Cambridge, Massachusetts, 1963.
3. J. Hagenauer, E. Offer, C. Measson, M. Morz, "Decoding and Equalization with Analog Non-linear Networks", European Transactions on Telecommunications, Oct 1999.
4. N. Kehtarnavaz, B. Simsek, "C6X-Based Digital Signal Processing", Prentice Hall, 2000.
5. SPRU189C, "TMS320C62X/C67X CPU and Instruction Set", Reference Guide, Texas Instruments, 1998.
6. SPRU301A, "TMS320C6000 Code Composer Studio", Tutorial, Texas Instruments, 1999.
7. SPRU186C, "TMS320C6X Assembly Language Tools", User's Guide, Texas Instruments, 1998.