Efficient Implementation of Low Density Parity Check Codes for Satellite Ground Terminals

Nareder Kumar[#], Chandra Prakash[$], S.N. Satashia, Virender Kumar, K.S. Parikh
Space Applications Centre (ISRO),
Ahmedabad (India)
Email: narender@sac.isro.gov.in[#], chandra@sac.isro.gov.in[$]

***Abstract*— Low Density Parity Check (LDPC) codes have gained lot of importance in the channel coding arena, because these can provide excellent performance close to Shannon limit & can easily beat the best known turbo codes for large block lengths. This paper explains generic algorithms for hardware efficient implementation of Encoder & Decoder for LDPC codes adopted by Digital Video Broadcast-Satellite-Second Generation (DVB-S2) standard. Low complexity & high throughput architectures have been proposed for LDPC Encoder & Sum Product Algorithm based LDPC Decoder. MATLAB & HDL simulation results are presented for proposed encoder & decoder architectures. Satellite link test results of a scaled down FPGA based implementation with IESS satellite modem are also presented.**

Keywords*— *Channel Coding; LDPC Encoder; LDPC Decoder; DVBS2; Sum Product Algorithm; Satellite Communication; Irregular Repeat & Accumulate LDPC Codes; HDL Implementation.

I. INTRODUCTION

LDPC codes belong to a class of highly efficient linear block codes which can provide performance very close to the channel capacity using iterative soft-decision decoding approach. Due to strong advantages LDPC codes have been adopted by many standards such as DVB-S2, DVB-T2, DVB-C2, WLAN(802.11) , WPAN(802.15) , Mobile Broadband Wireless Access (802.20), Broadband Wireless Metropolitan Area networks(802.16) etc. [1][2].

LDPC Codes are the block codes with a sparse parity check matrix which can also represented in form of a bi-partite graph known as "Tanner Graph". Tanner graph consist of two type of nodes i.e. Bit Nodes (columns of H) and Check Nodes. (rows of H). Each nonzero element $H(i,j)$ of parity check matrix H corresponds to edge connecting j^{th} bit node to i^{th} check node.

LDPC codes can be broadly classified into two categories i.e. binary and non binary codes. Non binary LDPC Codes are defined on Galois field GF(q), where all symbols of code word belongs to GF(q) & binary LDPC codes which are defined in a binary field GF(2) i.e. {0,1}. Further there are two more categories of LDPC codes based on their parity check matrix i.e. Regular & Irregular LDPC Codes. Regular LDPC codes are characterized by constant degree bit nodes & check nodes. Minimum distance between code words increases linearly with code length [3], which means they require very large code length for better performance Whereas Irregular LDPC codes are characterized by variable degree bit nodes & check nodes & can be specified by corresponding degree distribution polynomials $\lambda(x)$ & $\rho(x)$.

$$\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1} \qquad \rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1} \tag{1}$$

Where, λ_i is the fraction of edges connected to bit nodes of degree i & ρ_i is the fraction of edges connected to check nodes of degree i . Irregular LDPC codes have better minimum distance properties but they have high encoding complexity, because their generator matrix is not sparse [4][1].

Recent Digital Video Broadcast Standard Second Generation (DVB-S2) has adopted a powerful category of LDPC codes known as Irregular Repeat & Accumulate (IRA) LDPC Codes which are a type of structured LDPC codes with constrained degree distribution polynomials. These codes have strictly concentrated check node degree & variable bit node degree. Parity check matrix of these codes is of following form

$$\mathcal{H}_{(n-k)\times n} = \left[A_{(n-k)\times k} \mid B_{(n-k)\times(n-k)}\right] \tag{2}$$

Where n is codeword length & k is message block length. Matrix A is a sparse and pseudo random matrix with circulant periodicity constraints. These periodicity constraints lead to significant reduction in storage & design complexity of the encoder & decoder. B is lower triangular staircase (dual diagonal) matrix. Further all bit nodes & check nodes are divided into groups of M consecutive nodes. List of check nodes connected to leading bit node in every information bit node group has been specified in tables, these check node indices can be used to find indices of check nodes connected to remaining $M-1$ information bit nodes by cyclic shifting of check node indices of previous bit node by $q = ((n-k))/M$.

DVB-S2 standard has specified IRA LDPC Codes with block lengths i.e. Normal Frame (N=64800) & Short Frame (N=16200) and M=360. Eleven code rates are specified for normal frame & ten code rates are specified for short frame [5].

Section II describes encoding algorithm & architecture. Section III discusses about decoding algorithm & decoder architecture. While section IV & V discusses implementation & simulation results. Sections V conclude the paper.

II. LDPC ENCODING ALGORITHM & ARCHITECTURE

The parity check matrix of IRA LDPC codes has been structured in such a way that message can be directly encoded by using parity check matrix itself & hence eliminates the need of calculating Generator matrix G. The equations for calculating parity bits from binary parity check matrix can be derived from tanner graph.

$$p_0 = \bigoplus_{z \in IN(0)} m_z = S_0$$

$$p_1 = \bigoplus_{z \in IN(1)} m_z \oplus p_0 = S_1 \oplus S_0$$

These equations can be generalized as

$$p_r = \bigoplus_{z \in IN(r)} m_z \oplus p_{r-1} = S_r \oplus S_{r-1} \cdots S_1 \oplus S_0 \quad (3)$$

Where $r = 0,1,2..n-k-1$ and $IN(r)$ denotes the information nodes connected to check node CN_r. These parity bits are appended with information bits to form complete code word.

One of the simplest method for encoding would be sequential check node processing (commonly called horizontal processing) as per equation (3) but it will require a priori knowledge of all information bits which will lead to very high encoding latency (less throughput) for large messages which is not good for low bit rate applications.

However another approach suggested by Gomes et al [6] overcome this limitation by partially-parallel bit node (vertical) processing. This technique exploits the periodic structure of parity check matrix & process M information nodes simultaneously & updates the corresponding temporary parity bits. After processing complete message, these temporary parity bits require post processing as per staircase matrix B to get final parity bits.

A. Encoder Archtecture

Figure-1 shows the basic block diagram of the proposed architecture of the generic LDPC encoder which is based on partially-parallel bit node processing scheme suggested by Gomes [6] & supports all DVB-S2 code rates for long frames & short frames (excluding special code rates with non integer check node degree). Encoder accepts the information bits serially into M bit shift register. Encoder Control block contains the state machine which implements the memory access scheme as per encoding algorithm described in following pseudo code.

```
KVAL = K/M
NVAL = N/M

At start of each message block

Initialize PARITY_RAM with all 0;
ptr=0;
accum_reg=0;
 (for every M information bits acquired in
                data_reg)
for i=0:KVAL-1
  CODE_RAM[i] = data_reg;
  W = WEIGHT_ROM(i);

 for j=0:W-1
   row = ROW_ROM[ptr]
   shift = SHIFT_ROM[ptr]
   I_rot = rot(data_reg,shift)
           (right circular shifting)

   PARITY_RAM[row]=PARITY_RAM[row]⊕ I_rot
        accum_reg = accum_reg ⊕ I_rot
   ptr = ptr + 1
  end

end

accum_val = proc_func(accum_reg)
(Triangulation    &    right    shifting
operation)[6]

for k=0:q-1
  accum_val = accum_val ⊕ PARITY_RAM[j]
  CODE_RAM[K_VAL +j]= accum_val
end
```

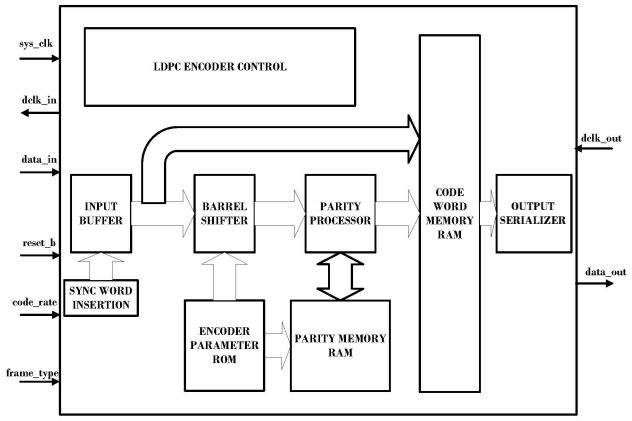

Figure 1: Block diagram of LDPC Encoder

Barrel shifter performs quick cyclic shifting operations. Parity processor implements the accumulation, triangulation & right shifting operations. Encoder ROM holds the row & shift parameters which are calculated from the location of ones in leading columns of every information bit node group specified in form of tables in appendix C in DVB-S2 standard document [5] for every code rate & frame type. 90 bit Unique Sync word is inserted periodically in the code for decoder synchronization. Parity RAM holds the temporary parity bits organized in form

of $q \times M$ memory. Code RAM holds the information bits & all parity & it is also organized in form of $N\_VAL \times M$ memory. After encoding is completed CODE_RAM is read sequentially & output is shifted serially at faster clock rate (f_{clkin}/R $where$ $R = K/N$).

III. LDPC Decoding Algorithm & Architecture

The algorithm used for decoding LDPC codes is called message passing algorithm, where messages (real numbers) are passed iteratively between two sets of nodes called bit nodes & check nodes, across the edges of tanner graph. Message passing algorithm operates on received code word bits from demodulator. At the cost of hardware complexity, LDPC decoder provides better performance for soft decision bits as compared to hard decision. Optimal decoding algorithm which is also known as the Sum product algorithm (Belief Propagation algorithm) is chosen for implementation as it achieves the theoretical performance bound. It refines the likelihood probabilities of the received code word bits using the messages exchanged between bit nodes & check nodes iteratively for satisfying parity check constraints.

Sum Product Algorithm (Soft Decoding)

Soft decision bits from demodulator are represented in terms of log likelihood ratios (LLR) for simplifying calculations in decoder. Let c_i be the transmitted bit & y_i be the received soft decision bit. Then corresponding LLR can be given by

$$\mu_i = log\left(\frac{p(c_i = 0/y_i)}{p(c_i = 1/y_i)}\right) \tag{4}$$

μ_i represents the conditional probability that received bit is more likely to be 1 or 0. Larger positive value of LLR means received bit is more likely 0 whereas larger negative value means received bit is more likely 1. Let BN(i) be the set of BN indices connected to CN_i & CN(i) be the set of check node indices connected to BN_i. Let $Q_{jk}[l]$ be the messages sent from BN_j to CN_k & $R_{jk}[l]$ be the messages sent from CN_j to BN_k during the l^{th} iteration & d_b & d_c be the bit node & check node degrees respectively.

1. (Initialization) All bit nodes are initialized with the received LLR values μ_j & broadcasted to neighboring check nodes connected to every bit node.

$$Q_{jk_i}[0] = \mu_j$$

2. (Check Node Update) Every check node processes the messages received from neighboring bit nodes $Q_{k_{ji}}[l-1]$ during previous iteration & replies to neighboring bit nodes with the message $R_{ik_j}[l]$.

$$R_{ik_j}[l] = \left(\left(\sum_{m\in BN(i)} \psi(Q_{mi}[l-1])\right) - \psi(Q_{k_{ji}}[l-1]\right) \times \left(sgn\left(Q_{k_{ji}}[l-1]\right).\prod_{m\in BN(i)} \text{sgn}(Q_{mi}[l-1])\right) \tag{5}$$

$$\psi(x) = -\ln\left(tanh\left(\left|\frac{x}{2}\right|\right)\right) \quad \text{sgn(x)} = \begin{cases} 1 \text{ if } x \geq 0 \\ -1 \text{ if } x < 0 \end{cases}$$

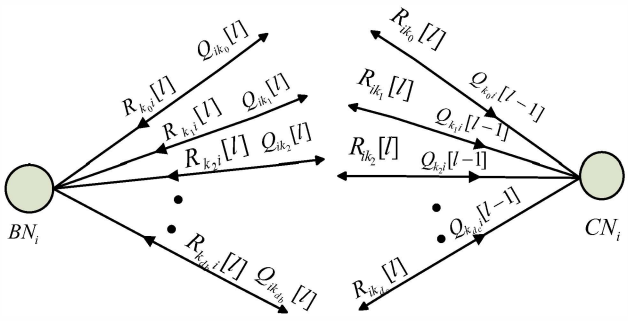

Figure 2: Message passing during Bit Node & Check Node Update steps in SPA Algorithm

3. (Bit Node Update) Thereafter, every bit node processes the message received from neighboring check nodes during check node update step & replies to neighboring check nodes with the message $Q_{ik_j}[l]$.

$$Q_{ik_j}[l] = \mu_i + \left(\sum_{m\in CN(i)} \left(sgn(R_{mi}[l]).\psi(R_{mi}[l])\right)\right) - \left(sgn\left(R_{k_{ji}}[l]\right).\psi(R_{k_{ji}}[l])\right) \tag{6}$$

4. After Bit Node Update the soft decision codeword can be given by

$$S_i[l] = \mu_i + \sum_{m\in CN(i)} \left(sgn(R_{mi}[l]) \times \psi(R_{mi}[l])\right) \tag{7}$$

Hard decision codeword can be given by

$$z_i = \begin{cases} 0 & if\ S_i \geq 0 \\ 1 & if\ S_i < 0 \end{cases}$$

This hard decision codeword is checked for parity check constraints by using following parity check equation

$$zH^T = 0 \tag{8}$$

Steps 2, 3 and 4 are repeated until iteration limit is reached or parity check constraints are satisfied [2].

Different simplified versions of Sum product algorithm like Minimum Sum Algorithm, $\lambda - min$ algorithm, Soft-In-Soft-Out (SISO) algorithm etc. have been proposed in literature which exploit the hyperbolic nature of $\psi(x)$ & approximate it in form of simpler functions for saving computations & hardware resources [9][10].

LDPC Decoder Architecure

Among various architectures which have been published in literature partially parallel architectures [7] are more popular as compared to fully parallel which provide high throughput at the cost of hardware complexity & fully serial architecture [8] which have extremely low throughput. Partially parallel architectures provide high throughput by exploiting the circulant structure of parity check matrix for IRA LDPC codes.

Figure-3 shows basic block diagram of the proposed architecture of generic LDPC decoder which supports all DVBS2 code rates for long frames & short frames (excluding special code rates with non integer check node degree).

The proposed decoder operates on LLR messages which can be obtained by augmenting soft decision bits from demodulator with estimated SNR. These LLR messages have been quantized in 6 bits using variable precision quantization with finer resolution in the unreliable region & coarse resolution in the reliable resolution [9][10].

Sync Word Detector module synchronizes the decoder to the code word boundaries & helps to resolve the phase ambiguity in demodulation. It keeps monitoring the sync word periodically, so that whenever it starts missing the sync words more frequently, it resets the decoder. Input buffer stores the LLR messages when decoder is busy & it converts the serial messages into parallel M messages for initializing the LLR buffer Memory.

Functional Units implement the bit node update equation (5) & check node update equations (5) of Sum Product Algorithm. $\psi(x)$ is critical part of functional unit & it has significant impact on LDPC decoder's performance. It is an hyperbolic function & can be implemented using direct base 2 arithmetic or piece-wise linear approximation or Look-Up Table (LUT). $\psi(x)$ is implemented as LUT because it has less implementation complexity as compared to other approaches.

Zhang et al. [9] proposed a scheme for finite precision implementation of $\psi(x)$ using LUT. This scheme proposed $2^6 \times 6$ LUT for 6-bit messages with input messages having variable quantization & output messages having uniform quantization. Further Oh & Parhi [11] proposed a variable quantization scheme for $\psi(x)$ LUT which helps to reduce LUT size by 75% to $2^4 \times 6$ & also lead to reduction in width of message memory to 5 bits (1 sign+4magnitude). Different FIFOs have been used for pipelining to ensure continuous flow of messages through functional unit & maximizing the throughput. Message passing algorithm has been tweaked such that maximum resources can be shared among bit node & check node update steps [12].

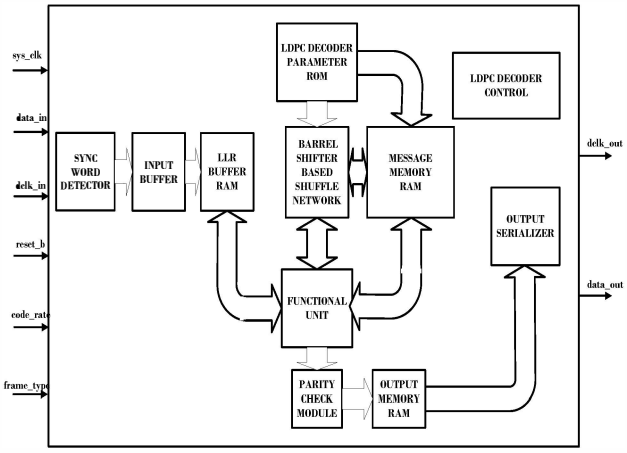

Figure 3: Block Diagram of LDPC Decoder

The proposed LDPC decoder has four types of memories viz. Parameter ROM, LLR Buffer RAM, Message Memory & Output Memory. Decoder Parameter ROM holds the row, shift & invert-shift parameters which are obtained from the location of ones in leading columns of every information bit node group specified in form of tables in appendix C in DVBS2 standard document [5][2]. LLR buffer memory has been organized as $^{N}/_{M} \times M$ 6-bit wide memory & is initialized by M parallel 6-bit messages from input buffer. Message memory, which holds the messages exchanged between bit nodes & check nodes across edges of tanner graph, is divided into two groups. First group contains edges corresponding to location of ones in matrix A (TOP RAM) & second group contains edges corresponding to location of ones in matrix B (BOTTOM RAM). Both memories have been organized as $q(d_c - 2) \times M$ & $2q \times M$ 5-bit wide memories [7]. Finally, output memory, which holds the decoded data bits, has been organized in form of $^{N}/_{M} \times M$ 1-bit wide memory.

Memory mapping ensures that for check node processing both TOP RAM & BOTTOM RAM are accessed sequentially & for bit node processing, TOP RAM access is indexed & cyclically shifted as per corresponding row & shift coefficients & BOTTOM RAM access is sequential. This node processing scheme requires no shifting for check node processing & two shifting operations for bit node processing (one shifting at the time of reading & another inverted shifting at the time of writing the updated messages). Hence, two shuffling networks are required for achieving maximum throughput. However, same throughput can also be achieved by offloading one shifting operation to check node processing step, which means that updated messages can be cyclically shifted with invert-shift coefficients & written back sequentially to TOP RAM & BOTTOM RAM to set up for bit node processing. This offloading improves the throughput & eliminates the need for second shuffling network. The new shift coefficients are obtained by sorting the row coefficients sequentially & inverting associated shift coefficients.

Decoder Control block generates the memory access commands for Sum Product Algorithm which is realized in form of state flow diagram in figure 4.

In INIT state LLR Buffer memory is initialized with the codeword LLR values received from demodulator & are simultaneously compressed in Functional Units & written to message memory for check node processing step through shuffle network.

Iterative decoding starts with check node update (CNUP) step, in which the messages corresponding to every check node ($dc - 2$ consecutive rows of TOP RAM & 2 consecutive rows of BOTTOM RAM) are read from message memory sequentially as M parallel messages at a time. These are passed on to M Functional Units without any cyclic shift for processing as per check node update equation (5) for computing the return messages to corresponding bit nodes.. After processing the outputs from Functional Units are shifted by the value ishift from ROM & written back to TOP RAM & BOTTOM RAM in same sequence for reducing burden in Bit Node Update step.

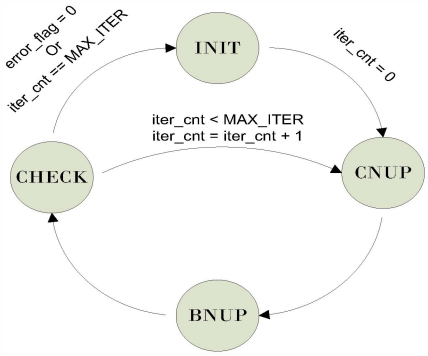

Figure 4: State Flow diagram of LDPC Decoder

After processing all check nodes, the decoder proceeds to Bit Node Update (BNUP) step, in which all messages corresponding to every bit node are read from message memory & fed to FUs without any shifting. Simultaneously respective LLR values are read from LLR buffer memory & fed to FUs for calculating updated LLR values & return messages as per equation (6) & (7). After processing, the outputs from FU's written back to TOP RAM on locations pointed by values from row ROM after cyclic shifting by shift coefficients & to BOTTOM RAM on consecutive locations without any cyclic shift.

The Parity Check step proceeds simultaneously with bit node update step and all parity check equations are checked as per equation & hard decision code word is written to Output memory.

At the end of bit node update step if all parity check equations are satisfied then decoder aborts the decoding process & takes next frame for decoding otherwise it continues for next iteration until iteration limit (50 iterations) is reached.

IV. Implementation Results

LDPC Encoder & Decoder's Verilog HDL models has been synthesized & ported on Xilinx Virtex-6 FPGA (XC6VLX240T) device. The overall resource utilization is summarized in the table-1 & table-2. LDPC Encoder & Decoder implementation results have been compared with equivalent IP cores from industry [14] [15].

Table 1 : FPGA Resource Utilization of LDPC Encoder

FPGA Resources	SoftJin DVB-S2 LDPC Encoder[14]	Proposed LDPC Encoder
Slice Flip Flops	7447	3400
Slice LUT	10862	3070
BRAM	15	20
Maximum frequency	167 MHz	250 MHz

The BRAM requirements have increased due to the serial interface proposed in this implementation. Throughput of encoder & decoder depends on maximum operating clock frequency & number of clock cycles required for encoding/decoding & number of iterations. The proposed decoder can achieve very high throughput at high SNR because number of decoding iterations decreases as SNR increases.

Table 2 : FPGA Resource Utilization of LDPC Decoder

FPGA Resources	SoftJin DVB-S2 LDPC Decoder[15]	Proposed LDPC Decoder
Slice Flip Flops	75K	48K
Slice LUT	194K	113K
BRAM	151	150
Maximum frequency	140 MHz	100 MHz

V. Simulation & Test results

MATLAB & Verilog HDL models based on proposed architectures have been realized for LDPC Encoder & Decoder for the codes adopted by DVB-S2 standard. MATLAB model have been simulated for both fixed point & floating point precisions.

Simulation environment involves generation of random bit stream, encoding using LDPC encoder model for different DVBS2 code rates. The encoded bit stream has been modulated using BPSK/QPSK/8-PSK modulator & passed through AWGN channel. Thereafter received signal has been demodulated using BPSK/QPSK/8-PSK demodulator with LLR output & decoded using LDPC decoder Model. The

LDPC Decoder performance has been analyzed by measuring Bit Error Rate (BER) for different SNR values.

Figure-5, 6, 7 and 8 shows noise performance of different code rates for normal frame (n=64800) for different modulations. It is evident from floating point simulation results that these codes can provide coding gain as high as 10dB at BER of 10^{-6}.

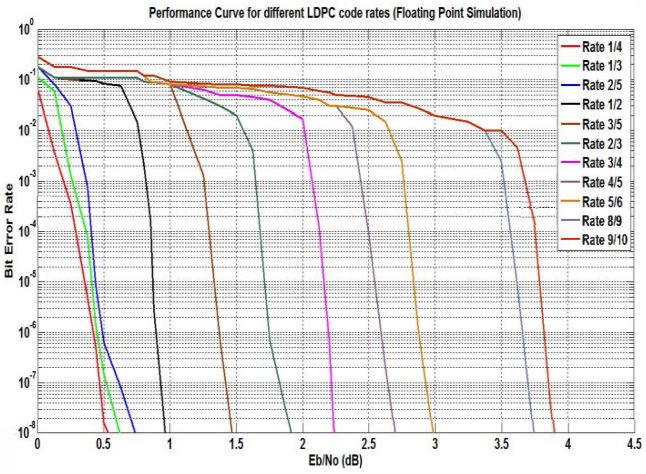

Figure 5 MATLAB Simulation results for floating point model of LDPC Decoder for BPSK/QPSK modulation

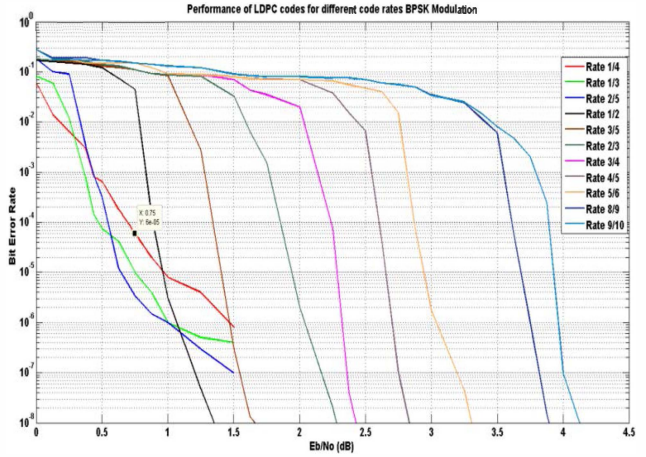

Figure 6 Simulation results for Fixed point MATLAB Model or HDL model of LDPC Decoder for BPSK/QPSK modulation

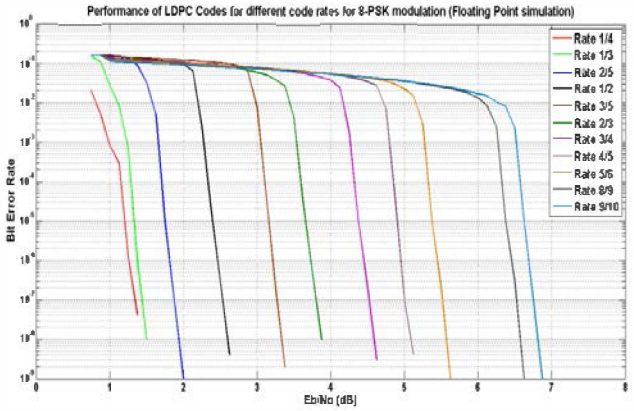

Figure 7 : Simulation results for MATLAB floating point model of LDPC decoder with 8PSK modulation

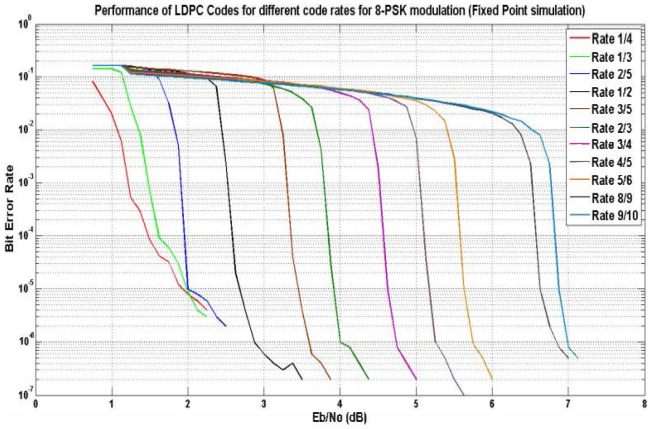

Figure 8 : Simulation results for MATLAB/HDL fixed point model of LDPC decoder with 8PSK modulation

Verilog HDL models have been verified by behavioural simulation in Xilinx ISIM. LDPC Encoder Models have been verified by comparing encoded data with MATLAB proprietary software function. LDPC Decoder has been verified by providing quantized LLR's from MATLAB & comparing the decoded data with original transmitted data for bit error rate calculation. However in fixed point simulation, performance degradations have been observed for rate 1/4, 1/3 & 2/5 and error floors have been observed for other code rates. But these error floors are much below than 10^{-6} & can be easily removed by outer BCH codes. Also in fixed point simulation performance loss of 0.2dB has been observed which is due to quantization & message compression [11]. Behavioral simulation results have been found similar to fixed point simulation results.

VI. Conclusion & Future Work

High throughput has been achieved in the LDPC Encoder & Decoder implementation by using partially parallel processing & FIFO based pipelining. Further 75% reduction in logarithmic hyperbolic tangent function ($\psi(x)$) LUT Size & 1-bit reduction in message memory width has been achieved by using proposed LUT compression scheme. The Verilog HDL

models of LDPC Encoder & Decoder has been ported to Xilinx Virtex-6 FPGA & tested in satellite link with hard decision bits interface from IESS demodulator. Coding gain of 6dB has been recorded for code rate 3/5 at 10^{-8} BER with BPSK modulation up to 10 Mbps. This decoder is being integrated with soft demodulator providing 6-bit soft decision.

REFERENCES

[1] P. H. Siegel, *An Introduction to Low-Density Parity-Check Codes*. University of California, San Diego, 2007.

[2] K. Loi, "Field-Programmable Gate-Array (FPGA) Implementation of Low-Density Parity-Check (LDPC) Decoder in Digital Video Broadcasting Second Generation Satellite (DVB-S2)," M.Sc. Thesis, University of Saskatchewan, 2010.

[3] F. Daneshgaran, M. Laddomada, and M. Mondin, "An algorithm for the computation of the minimum distance of LDPC codes," *European Transactions on Telecommunications*, vol. 17, pp. 57-62, 2006.

[4] R. G. Gallager, "Low-density parity-check codes," 1963.

[5] ETSI, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, satellite Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)," EN 302 307 V 1.2.1, 2009.

[6] M. Gomes, et al., "High Throughput Encoder Architecture for DVB-S2 LDPC-IRA Codes," in *IEEE ICM*, 2007.

[7] M. Eroz, F. W. Sun, and L. N. Lee, "An Innovative Low-Density Parity-Check Code Design With near-Shannon-limit Performance and Simple Implementation," *IEEE Transactions on Communications*, vol. 54, pp. 13-17, Jan. 2006.

[8] A. Howland and C. Blanksby, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 404-412, Mar. 2002.

[9] T. Zhang, Z. Wang, and K. K. Parhi, "On Finite Precision Implementation of Low Density Parity Check Codes Decoder," *IEEE International Symposium on Circuits*, vol. 4, pp. 202-205, 2001.

[10] L. H. Sheryl, C. G. Vincent, and C. Schlege, "Soft-Bit Decoding of Regular Low-Density Parity-Check Codes," in *University of Alberta*, 2007.

[11] K. K. Parhi and D. Oh, "Low Complexity Implementations of Sum-Product Algorithm for Decoding Low-Density Parity-Check Codes," *IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 262-267, Oct. 2006.

[12] Z. Zhang, L. Dolecek, M. Wainwright, V. Anantharam, and B. Nikolić, "Quantization Effects in Low-Density Parity-Check Decoders," in *ICC proceedings*, 2007.

[13] Xilinx, "Virtex-6 FPGA Family Overview", January 2012. [Online]. Available : http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf

[14] SoftJin Technologies Pvt. Ltd., "LDPC Encoder for DVB-S2", May 2010. [Online]. Available : http://www.softjin.com/IP_Datasheet_PDF_Version/LDPC_Encoder_datasheet.pdf

[15] SoftJin Technologies Pvt. Ltd., "LDPC Decoder for DVB-S2", May 2010. [Online]. Available : http://www.softjin.com/IP_Datasheet_PDF_Version/LDPC_Decoder_datasheet.pdf