LOW COMPLEXITY DEMAPPING FOR LDPC RECEIVERS USING HIGH ORDER MODULATION SCHEMES

by

Suhas Prahalada

B.E. Visvesvaraya Technological University, INDIA, 2011

A Project Submitted in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

University of Victoria

LOW COMPLEXITY DEMAPPING FOR LDPC RECEIVERS USING HIGH ORDER MODULATION SCHEMES

by

Suhas Prahalada

B.E., Visvesvaraya Technological Univeristy, India 2011

Supervisory Committee

Dr. Hong-Chuan Yang, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Amirali Baniasadi, Department Member
(Department of Electrical and Computer Engineering)

ABSTRACT

In this project soft decision demapping for Low Density Parity Check (LDPC) code receivers using high order modulation schemes is investigated in detail and an improvement to low complexity demapping scheme is proposed. In order to achieve better bit error rate (BER) performance of LDPC code decoder, the received signal should be soft decided rather than hard decided. However, for high order modulation schemes soft decision demapper with conventional log likelihood ratio (LLR) typically involves high computational complexity. The proposed demapper reduces computation complexity by 50% and offers BER performance comparable to conventional LLR method. The proposed method has linear complexity by avoiding multiple square and exponential operations. The proposed method is applied to 8-Phase Shift Keying (8PSK) modulation scheme in digital video broadcasting via satellite (DVB-S2) standard and its performance is compared against conventional LLR and max-log approximation.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to thank

My parents for providing education since my childhood

Dr. Hong-Chuan Yang for guiding me through graduate studies and

Dr. Amirali Baniasadi for serving on the supervisory committee.

Chapter 1 INTRODUCTION

Communication systems are becoming an essential component of every computing platform from wireless sensors, mobile phones, to data servers. Throughputs of future communication systems are expected to increase beyond hundreds of Mbps and even Gbps. Error correction plays a major role in communication systems to increase the transmission reliability and achieve a better error correction performance with less signal power. Forward error correction (FEC) took on a key role in ensuring reliable transmission of data and gradually became mainstream technology for wireless and optical communication systems. FEC decoding can be carried out with either of two decision methods: hard decision and soft decision. The input to a hard-decision FEC decoder consists of binary bits 0 and 1. On the other hand, soft decision decoding requires a stream of 'soft bits' which include not only the 1 or 0 decision but also an indication of how certain we are about the correctness of the decision. When operating at the same coding rate, soft decision typically offers higher coding gain, but with higher processing complexity.

FEC using Low-Density Parity–Check (LDPC) codes with soft decoding has rapidly gained acceptance in the wireless communication community. LDPC code was first developed in 1962 [1] as an error correction code that allowed communication over noisy channels very close to the Shannon limit. With advancements in VLSI, LDPC codes have recently received a lot of attention because of their superior error correction performance with iterative decoding using a message passing algorithm [1]. Recently, they have been adopted as the FEC method for many emerging standards, such as digital video broadcasting via satellite (DVB-S2) [3], WiMAX

standard (802.16e) [4], and G.hn/G.9960 standard [5]. While many previous works focus on complexity reduction of LDPC decoding algorithm, the complexity for reliability initialization at the entry of LDPC decoding algorithm has received little attention. Increasing the order of modulation increases the computational complexity of soft decision demapping scheme. In recent communication standards high order modulation schemes are frequently used to fulfill higher data rate and throughput requirements.

Second generation of DVB-S (DVB-S2) (EN 302 307) was standardized in 2003. The bandwidth efficiency was improved by up to 30% with the usage of powerful coding schemes like LDPC and BCH. DVB-S2 is envisaged for broadcast services including standard HDTV, interactive services including Internet access, and data content distribution (professional). DVB-S2 standard supports four different modulation schemes: quadrature phase shift keying (QPSK), 8PSK, 16-Amplitude and Phase shift keying (16APSK) and 32APSK. By choosing appropriate code rates and modulation schemes, spectral efficiency can be improved dependent on the capabilities of the satellite transponder. However, in order to achieve the performance required by this iterative channel coding method, a received signal should be soft-decided rather than hard-decided.

Therefore, efficient technique for soft-deciding received symbols in high order modulation scheme to log likelihood ratio (LLR) of information bits is needed. In this work we investigate LLR computation for high order modulation over Additive White Gaussian Noise (AWGN) channel. We also present the simulated performance of demapper for different LLR calculation methods.

1.1 Challenges and Literature Survey

In digital communication systems, the LLR has been used as reliability metric in soft decision decoding. Computation of LLR metric involves square, logarithm, exponential and addition operations. For lower order modulation schemes like binary phase shift keying (BPSK) and QPSK using a gray bit-to-symbol mapping, the LLRs of the codebits are independent and identically distributed (i.i.d.). Hence, LLR equation can be easily simplified and accurately approximated. For higher order modulation schemes as number of constellation points involved in LLR computation increases, approximation and simplification becomes more difficult.

The conventional LLR method has been used in most of the soft decision demappers [6], [7], but this method involves high hardware complexity and power consumption due to the complicated operations involved. The max-log approximation [8] eliminates the exponential and logarithm computation steps in LLR calculation, Hence it's more economical in terms of hardware resources and power consumption. In the Euclidean method proposed in [8] multiplication by channel estimation value is removed, But this requires square and square root operations. The square and square root operations lead to increased hardware complexity. In the phase selection method proposed in [8] the constellation is divided into 8 regions and LLR are computed based on phase angle of received symbol. This method eliminates squaring operations involved in LLR calculation and thus is less complex. However, its Bit Error Rate (BER) performance deteriorates as compared to conventional LLR and max-log methods. In [10] a method based on symmetry of constellation points is described for calculation of LLR for QPSK and 8PSK modulation schemes. In [10] and phase selection method in [8], the 8PSK constellation of DVB-S2 standard is rotated by $\pi/8$ and $-\pi/8$ for symmetry purpose. In [11] a

polar based demapper is proposed, where conventional LLR computation is approximated in the polar domain.

1.2 Contributions

1. In this project the complexity of different soft decision demapping schemes are studied and compared.
2. We also study performance of different demapping schemes by considering 8-PSK system in AWGN channel with DVB-S2 LDPC error correction code.
3. Finally we propose a low complexity demapping scheme and study the hardware implementation of the proposed demapping scheme.

1.3 Organization

The main goal of this project is an improvement to low complexity, low power consumption demapping scheme for soft decision decoding and for that we start by studying different demapping schemes for LDPC decoder, propose a low complexity demapping scheme, then study its performance, complexity and finally go on to estimate the hardware complexity of proposed scheme.

This report is organized as follows: Chapter 2 gives an overview of communication system, LDPC codes, and soft decision decoding. In chapter 3 computation of LLR, different approximations are investigated in detail. Low complexity LLR computation method is proposed and compared with conventional methods in terms of number of operations involved. In chapter 4 hardware

implementation complexity of proposed demapping scheme is studied. Chapter 5 provides concluding remarks.

Chapter 2 COMMUNICATION SYSTEM WITH LDPC

Encoding of information is accomplished by multiplying, in GF (2), an information vector by a generator matrix. Let S = $\{s_0, s_1, \ldots, s_i \ldots, s_{M-1}\}$ denote the set of possible transmit symbols. $log_2 M$ encoded bits $b_0, b_1, \ldots, b_j$ are grouped and mapped to a sequence of corresponding complex constellation symbols. During each symbol period, a symbol x from the symbol set S will be transmitted with identical probability to send $log_2 M$ bits information.

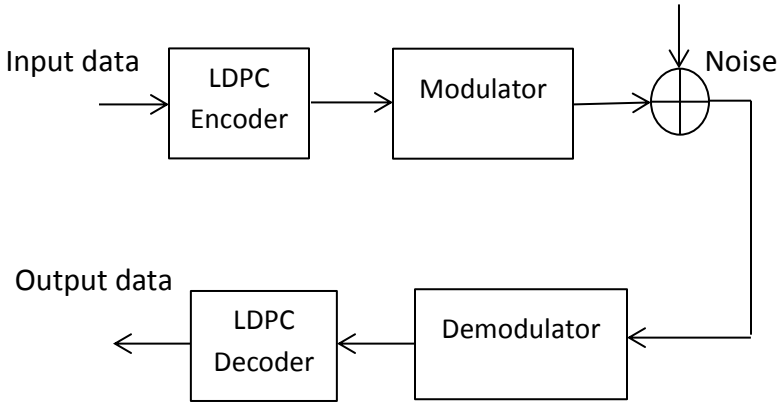

Figure 1. LDPC coded system block diagram.

In order to isolate the soft demapping performance from other effects, this project assumes AWGN as the only channel impairment. The probability that s_i was transmitted conditioning on noise-contaminated received signal r can be calculated as

$$Pr(x = s_i|r) = \frac{e^{\frac{-|r-s_i|^2}{2\sigma^2}}}{\sum_{i=1}^{M} e^{\frac{-|r-s_i|^2}{2\sigma^2}}} \tag{2.1}$$

The process of retrieving signal from corrupted received waveform is called detection. In soft-decision decoding, reliability metrics are calculated at the receiver based on the channel output r. The decoder uses these reliability measures to gain knowledge of the transmitted

codewords. The decoder inputs are codebit log-likelihood ratios computed from the received (noisy) modulation symbols.

LDPC codes are defined by an M x N binary matrix called the parity check matrix. The number of columns, N, denotes the codeword length. The number of rows, M, denotes the number of parity check constraints in the code. The message vector length K is K = M – N. The LDPC code in DVB-S2 has normal frames with the codeword length of 64800 bits and short frames with the codeword length of 16200 bits. Priori probabilities (LLR) for the received bits are the input to the decoder. The decoding algorithms for LDPC are iterative, and the procedure consists of variable message and check message and their updating. For each code bit the decoder iteratively computes an approximation of the maximum a posteriori (MAP) value known as a posteriori probability (APP).

In Hard decision decoding, the received codeword is compared with the all possible codewords and the codeword which gives the minimum Hamming distance is selected. In hard-decision the Hamming weight of the codeword is used as the branch metric, which is simply the number of positions in which the received codeword differs from the ideal codeword. Soft decision decoding makes use of relative magnitude of received symbols. In Soft decision decoding, the received codeword is compared with the all possible codewords and the codeword which gives the minimum Euclidean distance is selected. Thus the soft decision decoding improves the detection performance by utilizing additional reliability information (calculated Euclidean distance or calculated log-likelihood ratio). This fact delineates the improvement that will be seen when this soft decision scheme is used in combination with forward error correcting (FEC) schemes like convolution codes, LDPC etc.

Chapter 3 Demapper design

3.1 Log Likelihood Ratio

The likelihood ratio is the ratio of two probabilities of the same event, but under different hypotheses. As logarithm is a monotonically increasing function, the logarithm of a function achieves its maximum value at the same points as the function itself. The LLR is an important statistic that is easier to work with than the likelihood ratio itself and is particularly convenient with Gaussian noise statistics. Log Likelihood ratio is given by the formula below.

$$L_j = ln\frac{\Pr(b_j = 0|r)}{\Pr(b_j = 1|r)} \tag{3.1}$$

where L_j is log likelihood value of jth bit, numerator and denominator denote probability of jth bit being 0 or 1 given received signal r. The sign of L_j is the hard decision and magnitude of $|L_j|$ is the reliability of this hard decision. By applying Bayes rule above expression can be simplified as follows.

$$L_j = ln\frac{p(r|b_j = 0)\,\frac{Pr(b_j = 0)}{p(r)}}{p(r|b_j = 1)\,\frac{Pr(b_j = 1)}{p(r)}} \tag{3.2}$$

Where Pr denotes probability and p indicates probability density function (PDF). $p(r)$ is probability density function of r being received. If we assume Pr (b_j = 0) = Pr (b_j = 1) = ½. For $b_j \in (0, 1)$, LLR reduces to equation below

$$L_j = ln\frac{p(r|b_j=0)}{p(r|b_j=1)} \tag{3.3}$$

In the above expression $p(r|b_j)$ is Gaussian PDF given by

$$p(r|b_j) = \frac{1}{\sqrt{2\pi}\,\sigma}\; e^{\frac{-|r-\mu|^2}{2\sigma^2}} \tag{3.4}$$

Where μ is mean, σ^2 is noise variance and $p(r|b_j)$ is the likelihood r is received given b_j was sent. Group of bits b_0, $b_1, \ldots, b_j$ are mapped to symbol s_i. Generalizing above equation for multiple constellation points, we get

$$LLR(b_j) = ln\frac{\sum_{K_j,0}\; e^{\frac{-|r-s_i|^2}{2\sigma^2}}}{\sum_{K_j,1}\; e^{\frac{-|r-s_i|^2}{2\sigma^2}}} \tag{3.5}$$

where s_i is the i[th] constellation point, r is received symbol, $K_{j,0}$ is a set of M/2 constellation symbols in which bit b_j is 0 and $K_{j,1}$ is a set of M/2 constellation symbols in which bit b_j is 1.

The numerator involves a subset of M constellation points in which bit b_j is 0 and denominator involves a subset of points in which bit b_j is 1. The calculation of LLR involves following 2 steps.

- The calculation of Euclidean distance between the received symbol and each of the constellation points ($\frac{-|r-s_i|^2}{2\sigma^2}$). The calculation involves multiple multiplications and subtractions.
- The calculation of exponentials and logarithms ($log^{e^{-a}+e^{-b}+e^{-c}}$ - $log^{e^{-d}+e^{-e}+e^{-f}}$). The calculation is expensive in terms of hardware resources.

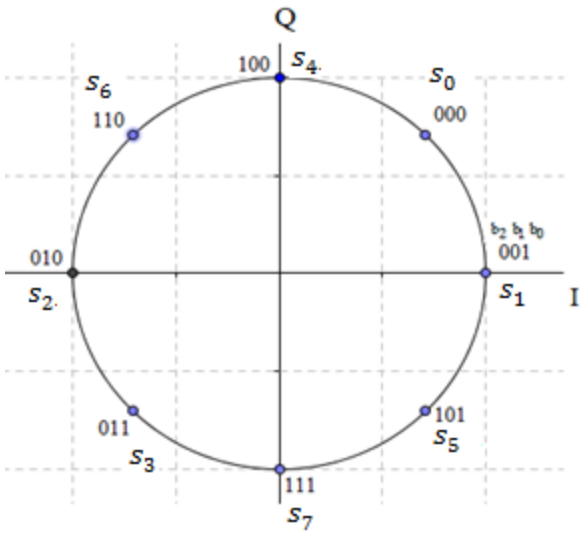

Figure 3.1. 8PSK modulation constellation diagram as per DVB-S2 standard.

In case of 8PSK modulation constellation above, 3 bits are grouped together and mapped onto complex symbolS_i . S_i is coordinate of constellation dots, b_2, b_1 and b_0. I and Q denote inphase (real) coordinate and quadrature (imaginary) coordinate of signal constellation. Log Likelihood ratio of bits b_2, b_1 and b_0 is given by equations below.

$$LLR(b_2) = ln(\frac{P_0 + P_1 + P_2 + P_3}{P_4 + P_5 + P_6 + P_7}) \tag{3.6}$$

$$LLR(b_1) = ln(\frac{P_0 + P_1 + P_4 + P_5}{P_2 + P_3 + P_6 + P_7}) \tag{3.7}$$

$$LLR(b_0) = ln(\frac{P_0 + P_2 + P_4 + P_6}{P_1 + P_3 + P_5 + P_7}) \tag{3.8}$$

$$P_i = \frac{1}{\sqrt{2\pi}\,\sigma}\, e\frac{-|r - s_i|^2}{2\sigma^2}, i = 0, \dots\dots, 7 \tag{3.9}$$

Where P_i is the likelihood of ith symbol sent given received signal r, σ^2 is variance in AWGN channel environment.

It can be noted from above equations that conventional LLR method requires multiplication operation on each symbol to calculate euclidean distance between received symbol r and constellation s_i. Once euclidean distance is computed further computation of exponential and logarithm requires additional hardware. In the above conventional approach all the constellation points are considered which increases number of complex operations that need to be performed. A common approximation to conventional LLR calculation is to replace multiple terms in the numerator and denominator by respective largest terms, i.e., by using only the nearest constellation point that has $b_j = 0$ in the numerator, and the nearest neighbor that has $b_j = 1$ in the denominator. Max-log approximation eases receiver implementation as logarithm and exponential computations are changed to minimum distances calculations.

The max log approximation makes use of the following approximation

$$ln(\sum P_i) \approx ln(\max(P_i)). \tag{3.9}$$

Using max-log approximation Log Likelihood ratio for 8PSK modulation is given below.

$$LLR(b_2) = max(Q_0, Q_1, Q_2, Q_3) - max(Q_4, Q_5, Q_6, Q_7) \tag{3.10}$$

$$LLR(b_1) = max(Q_0, Q_1, Q_4, Q_5) - max(Q_2, Q_3, Q_6, Q_7) \tag{3.11}$$

$$LLR(b_0) = max(Q_0, Q_2, Q_4, Q_6) - max(Q_1, Q_3, Q_5, Q_7) \tag{3.12}$$

$$Q_i = \frac{-|r-s_i|^2}{2\sigma^2}, i = 0, \dots\dots, 7 \tag{3.13}$$

3.12 Proposed demapping scheme

In order to simplify the LLR computation, we need to reduce the number of constellation points considered and try to eliminate the exponential and logarithm operations.

$$LLR(b_j) = ln \frac{\sum_{I_{j,0}} e^{\frac{-|r-s_i|^2}{2\sigma^2}}}{\sum_{I_{j,1}} e^{\frac{-|r-s_i|^2}{2\sigma^2}}} \tag{3.14}$$

where s_i is the i[th] constellation point, M denotes total number of constellation points, r is received symbol, $I_{j,0}$ is a set of M/4 constellation symbols in which bit b_j is 0 and $I_{j,1}$ is a set of M/4 constellation symbols in which bit b_j is 1.

The formula above is used for LLR computation in the proposed algorithm which uses $M/2$ constellation symbols. To identify the closest $M/2$ constellation points, one could normally compute the distances to all M constellation points and choose those ones that are closer. This is unnecessary, however. The constellation diagram can be divided into smaller regions, thus number of constellation points considered for LLR calculation can be reduced by simple comparison operations. In the proposed demapping scheme four closest points to the received symbol are considered for LLR calculation rather than eight points in 8PSK modulation. The constellation diagram is divided into 4 regions R1, R2, R3, and R4. The figure below shows decision regions for least significant bit b_0. I and Q denote inphase (real) coordinate and quadrature (imaginary) coordinate of signal constellation.

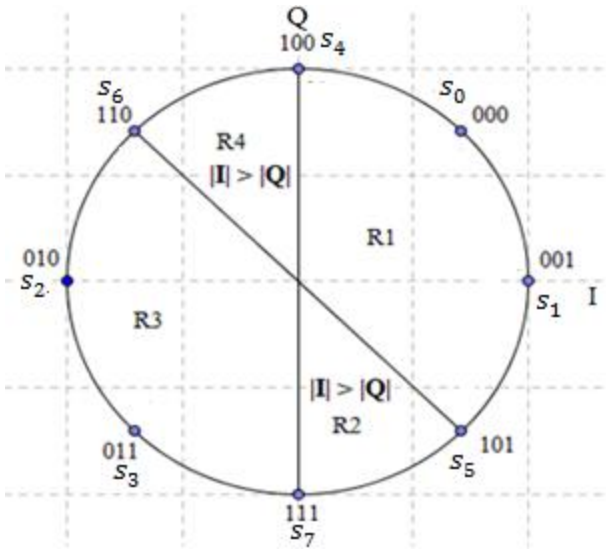

Figure 3.2. 8-PSK constellation diagram divided into multiple regions. Decision regions for bit LSB b0.

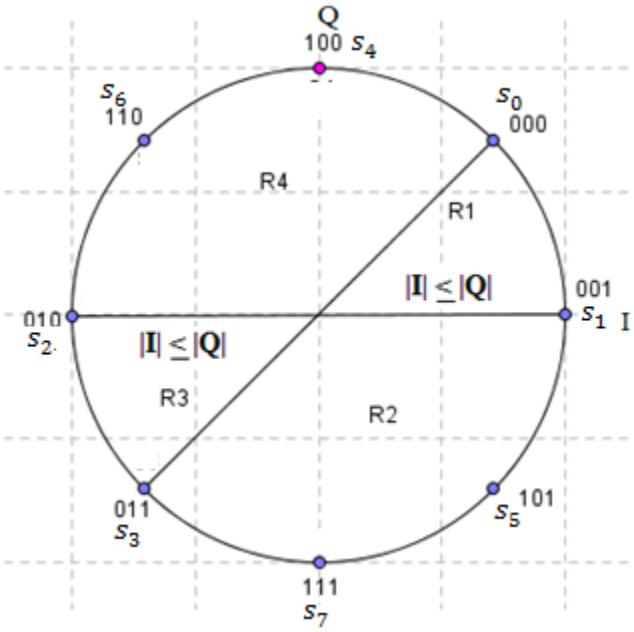

Figure 3.3. Decision boundary for bit b1.

The figure above shows decision regions R1, R2, R3, and R4 for middle bit b_1. For bit b_2, decision boundaries of b_0 and b_1 can be reused for computation of LLR. Decision boundaries of b_0 are used to determine closest symbols with bit 0 and decision boundaries of b_1 are used to determine closest symbols with bit 1 for computation of LLR of bit b_2. The computational

complexity can be greatly reduced as the receiver only needs to calculate the reliability using symbols in the truncated vectors got from decision regions. Based on the decision regions mentioned above, an algorithm for LLR calculation is described below.

Decision rules for bit b_0

If (I > 0)

If ((Q < 0) And (|I| > |Q|)

$$LLR(b_0) = ln(\frac{P_0 + P_2}{P_5 + P_7})$$

$$b_{2_{numerator}} = ln(P_1 + P_3)$$

Else

$$LLR(b_0) = ln(\frac{P_0 + P_4}{P_1 + P_5})$$

$$b_{2_{numerator}} = ln(P_0 + P_1)$$

Else

If ((Q> 0) And (|I| > |Q|))

$$LLR(b_0) = ln(\frac{P_4 + P_6}{P_1 + P_3})$$

$$b_{2_{numerator}} = ln(P_0 + P_2)$$

Else

$$LLR(b_0) = ln(\frac{P_2 + P_6}{P_3 + P_7})$$

$$b_{2_{numerator}} = ln(P_2 + P_3)$$

End

For bit b_1 decision rules are as follows.

If (Q > 0)

If ((I >0) And (|I| <= |Q|)

$$LLR(b_1) = ln(\frac{P_0 + P_1}{P_7 + P_6})$$

$$b_{2_{denominator}} = ln(P_4 + P_5)$$

Else

$$LLR(b_1) = ln(\frac{P_0 + P_4}{P_2 + P_6})$$

$$b_{2_{denominator}} = ln(P_4 + P_6)$$

Else

If ((I< 0) And (|I| <= |Q|))

$$LLR(b_1) = ln(\frac{P_4 + P_5}{P_2 + P_3})$$

$$b_{2_{denominator}} = ln(P_6 + P_7)$$

Else

$$LLR(b_1) = ln(\frac{P_1 + P_5}{P_3 + P_7})$$

$$b_{2_{denominator}} = ln(P_5 + P_7)$$

End

Finally, the calculation of LLR for is b_2

$$LLR(b_2) = b_{2_{numerator}} - b_{2_{denominator}}$$

$$P_i = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{\frac{-|r-s_i|^2}{2\sigma^2}},\ i = 0, \ldots\ldots, 7.$$

Where P_i is the likelihood of ith symbol sent given r received, σ^2 is variance in AWGN channel environment.

3.3 Performance and Complexity Evaluation

In this section the complexity of conventional method, max-log approximation and proposed method are reviewed in terms of number of operations. Let $r = r_x + jr_y$ and $s_i = s_x + js_y$ denote the received symbol and complex constellation point, then Euclidean distance calculation can be simplified as follows.

$$|r - s_i|^2 = \left(\sqrt{(r_x - s_x)^2 + \left(r_y - s_y\right)^2}\right)^2 \tag{3.15}$$

$$= (r_x - s_x)^2 + \left(r_y - s_y\right)^2 \tag{3.16}$$

The above operation involves 2 square operations and 3 addition/subtraction operations. Channel estimation value $\frac{1}{2\sigma^2}$ can be precalculated, if variance of noise is assumed constant and this operation involves 1 multiplication. Therefore calculation of Euclidean distance for each constellation point involves 3 multiplication and 3 addition operations. Based on above analysis number of operations per symbol in LLR calculation is shown in table below.

	Compare	**Exp/Log**	**Addition**	**Multiplication**
Conventional LLR		**8 / 2**	**25**	**24**
Max-log	**6**	**0**	**25**	**24**
Proposed approach	**6**	**4 / 2**	**13**	**12**

Table 3.1. Comparison of computation complexity of LLR computation methods.

It can be noted from the above table that in proposed approach we reduce number of multiplication and exponential operations by half as compared to conventional method.

However, in the proposed approach there are additional comparison operations. The comparison operations are much more economical in terms of hardware than multiplication and exponential operations.

In order to compare the performance of proposed method against that of max-log and conventional LLR method MATLAB simulation was performed. In the simulation DVB-S2 LDPC coding (normal frame and code rate 1/2) and 8PSK modulation scheme was used to evaluate the proposed demapping algorithm. The results, obtained by computer simulations, are given in terms of BER versus SNR in dB, where SNR is signal to noise ratio in decibels. The figure below shows performance of the proposed, max-log and conventional 8PSK soft decision demapping algorithm.

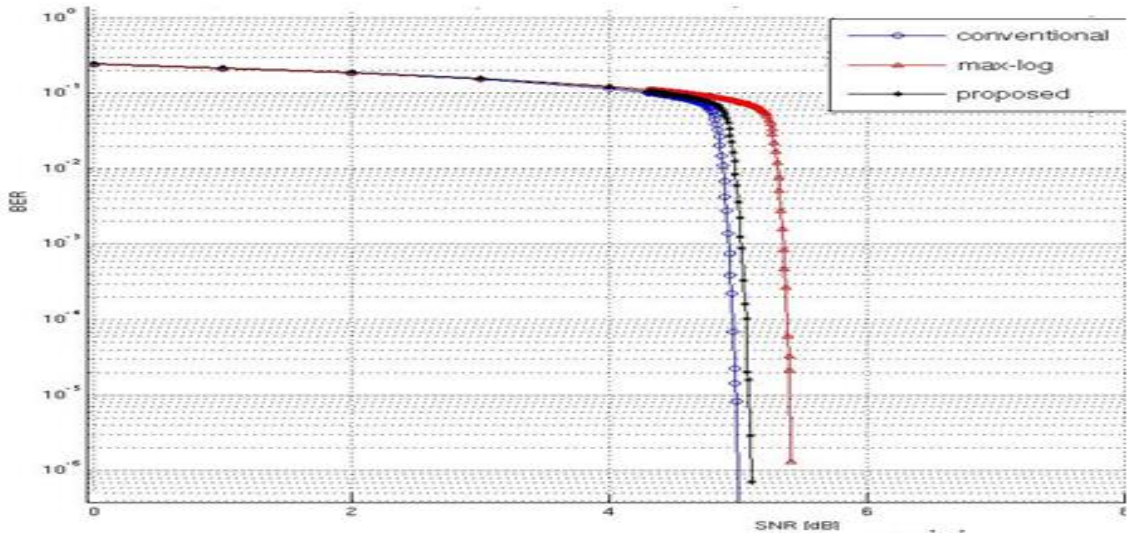

Figure 3.4 Performance comparison.

The curves have been generated by simulating over 5000 packets in AWGN channel model. As it can be observed the BER sharply decreases after 4.5 dB. The BER of proposed approach very

closely follows the conventional approach and performs better than max-log approximation by 0.5 dB.

Chapter 4 Study of Hardware Implementation

In this section we study the hardware implementation complexity of proposed demapping scheme against conventional demapping scheme. The demapper accepts one frame at a time performs demapping and outputs LLR of bits. High Level block diagram of demapper is shown.

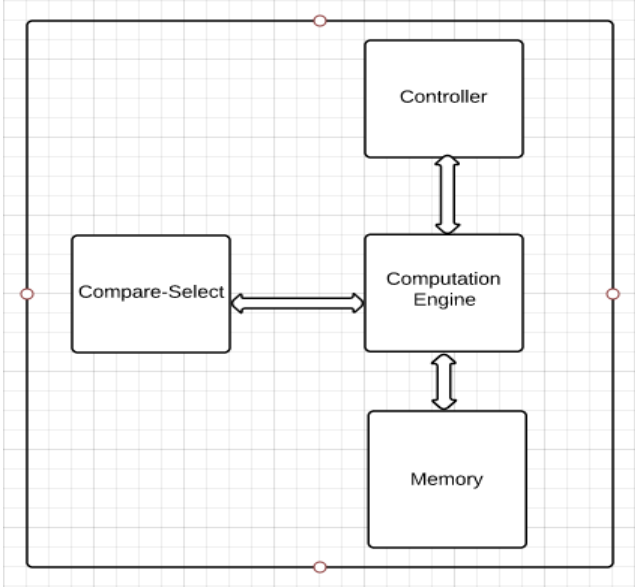

Figure 4.1. High Level Demapper Block diagram

Hardware resources of only comparison and computation block are estimated in this project. The comparison block was coded in VHDL and synthesized to get an estimate of hardware resources needed. A 10-bit fixed point representation of signal was assumed. The comparison block accepts received symbol r and produces constellation symbols to be used for LLR calculation. Latency of comparison block is 1 clock cycle per received symbol. The result of synthesis of comparison block is given below.

Hardware Resources	Number
Number of look up tables (LUT's)	24
Number of Register Slices	3
Power in mW @ 100MHz	26.13

Table 4.1 Hardware Resources for comparison block.

To get an estimate of hardware resources needed for computational blocks, fully optimized Xilinx IP's were assumed. Xilinx adder, multiplier and Coordinate Rotation Digital Computer (CORDIC) IP's were used. Both multiplier and CORDIC blocks were optimized for performance. Heavy parallelism was used to achieve a latency of one clock cycle for multiplier and CORDIC blocks. Adder block as well has a latency of one clock cycle.

Hardware Resources	Adder	Multiplier	CORDIC
Number of LUT's	11	111	497
Number of Register slices	10	20	14
Power in mW @ 100MHz	35.36	56.19	102.87

Table 4.2. Hardware cost estimates of computation blocks.

By comparing tables 4.1 and 4.2, it can be noted that hardware resources needed for comparison block is much less than multiplier and CORDIC blocks. In the proposed demapping scheme multiple multiplication and exponential operations are eliminated. Thus by eliminating multiple multiplication and exponential operations the throughput could be increased and power consumption could be reduced.

In max-log approximation exponential and logarithm operations are eliminated, thus CORDIC block in hardware is unnecessary. Thus max-log approximation is the most hardware efficient, low complexity implementation and proposed scheme can be thought of as an improvement to low complexity implementation.

A detailed dataflow diagram of computation branch is given below. No pipelining was assumed in the following computation branch.

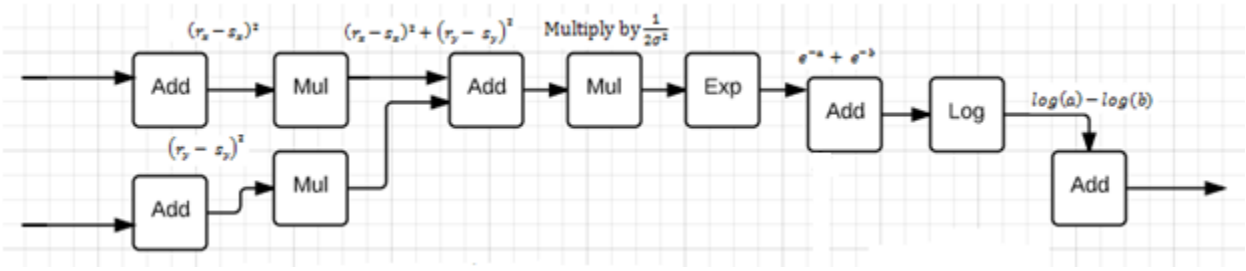

Figure 4.2 Computation branch dataflow diagram

As mentioned above each block in computation branch has latency of 1 clock cycle, hence latency of computation branch is 8 clock cycles. The computation branch here computes one likelihood term in equation (3.6). In a parallel implementation computation of equations (3.6), (3.14) would require 8 and 4 such computation branches respectively. The latency would be 8 clock cycles in a parallel implementation. In serial implementation the latency for computing equations (3.6), (3.14) would be 64 (8 * 8 constellation points) and 32 (8 * 4 constellation points) clock cycles.

Chapter 5 Concluding Remarks

This project proposed an improvement to low complexity (max-log) soft decision demapping algorithm for 8PSK constellation in DVB-S2 standard. The proposed algorithm reduces computation complexity, and thus, it is more economical in terms of hardware resources and power needed for computation. The proposed algorithm offers better performance than max-log approximation and negligible performance degradation compared to conventional LLR method. Even though there is slight performance degradation with respect to conventional LLR method, the proposed method can be implemented in hardware with less number of complex operations. The proposed soft decision demapping algorithm can be applied to other digital communication systems that use 8PSK modulation scheme.

Soft decision decoding is appearing in an increasing number of applications, which have strict power and complexity constraints and require good error performance. Thus, it is very important to have better soft demapping algorithms that provides good error performance and reduces hardware complexity in order to achieve the high throughput and high efficiency requirements of future applications. This technique can be further extrapolated for other higher order modulation schemes, such as 16-APSK and 32-APSK.

Appendix

A.1 Matlab simulation code

```
function information_bits = info_source(message_length)
    % -- draw random bits
    information_bits = round(rand(1,message_length));

end
```

```
================================================================================
% Title       : Simulator for LDPC codes
% ------------------------------------------------------------------------------
% Description:
%   This file performs the main Monte-Carlo simulation procedure.
%   Encodes LDPC codes described by the codes found in the codes/
%   folder transmits bits over an AWGN channel and calls the decoding
%   algorithm.
% ------------------------------------------------------------------------------
function sim_LDPC(RunID,TxRx,LDPC)
  randn('state',RunID)
  rand('state',RunID)

  % -- initialize
  BER = zeros(3,length(TxRx.Sim.SNR_dB_list));
  FER = zeros(3,length(TxRx.Sim.SNR_dB_list));
  tic; %for calculating computation time

  for trial=1:TxRx.Sim.nr_of_channels

    c = info_source(LDPC.inf_bits);
    bitpwr = (sum(c.^2))/(length(c));
    hEnc = comm.LDPCEncoder(LDPC.H);
    encodedData = step(hEnc,c');  %LDPC encoding
    s = psk_8(encodedData');      %8psk modulation

    for k=1:length(TxRx.Sim.SNR_dB_list)

      % -- AWGN channel
      SNR = 10^(TxRx.Sim.SNR_dB_list(k)/10); %Snr converted from dB.
      Sigpwr = (sum(abs(s).^2))/(length(s));
      sigma2 = Sigpwr/SNR;
      y = add_awgn(s,TxRx.Sim.SNR_dB_list(k),sigma2);

      % -- compute LLRs & decode
      LLR_A2 = psk8_detect(y,sigma2);    %8psk soft demapping
      hDec = comm.LDPCDecoder(LDPC.H);   %LDPC Decoding
      bit_output_classic = (step(hDec, LLR_A2(1,:)'))';
      bit_output_maxlog = (step(hDec, LLR_A2(2,:)'))';
      bit_output_proposed = (step(hDec, LLR_A2(3,:)'))';
```

```
      % -- calculate BER
      ref_output = (c==1);
      tmp = sum(abs(ref_output-bit_output_classic))/LDPC.inf_bits;
      BER(1,k) = BER(1,k) + tmp;
      FER(1,k) = FER(1,k) + (tmp>0);

      ref_output = (c==1);
      tmp = sum(abs(ref_output-bit_output_maxlog))/LDPC.inf_bits;
      BER(2,k) = BER(2,k) + tmp;
      FER(2,k) = FER(2,k) + (tmp>0);

      ref_output = (c==1);
      tmp = sum(abs(ref_output-bit_output_proposed))/LDPC.inf_bits;
      BER(3,k) = BER(3,k) + tmp;
      FER(3,k) = FER(3,k) + (tmp>0);
    end

    if mod(trial,10)==1
      disp(sprintf('Estimated remaining time is %1.1f
minutes.',(toc)/trial*(TxRx.Sim.nr_of_channels-trial)/60));
    end
  end

  % -- save results to disk
  Results.TxRx = TxRx;
  Results.LDPC = LDPC;
  Results.BER_classic = BER(1,:)/trial;
  Results.FER_classic = FER(1,:)/trial;
  Results.BER_maxlog = BER(2,:)/trial;
  Results.FER_maxlog = FER(2,:)/trial;
  Results.BER_proposed = BER(3,:)/trial;
  Results.FER_proposed = FER(3,:)/trial;
  Results.FileName = sprintf('results/%s_%d.mat',TxRx.Sim.name,RunID);
  save(Results.FileName,'Results');

  % -- generate BER plot
  figure(1);
  semilogy(TxRx.Sim.SNR_dB_list,Results.BER_classic,'bo-')
  hold on
  semilogy(TxRx.Sim.SNR_dB_list,Results.BER_maxlog,'r^-')
  hold on
  semilogy(TxRx.Sim.SNR_dB_list,Results.BER_proposed,'gh-')
  xlabel('SNR [dB]')
  ylabel('BER')
  grid on
  axis([ -1 8 1e-8 1])

  Figure_FileName = sprintf('results/%s_%d',TxRx.Sim.name,RunID);

  saveas(gcf, Figure_FileName, 'fig');% here you save the figure
return
```

```
function [decoded_llrs] = psk8_detect(rx_symbol,noise_variance)
    s0 =  1/sqrt(2)+(1i/sqrt(2));
    s1 =  0+1i;
    s2 =  -1+0i;
    s3 =  (-1/sqrt(2))+(1i/sqrt(2));
    s4 =  1 + 0i;
    s5 =  (1/sqrt(2))-(1i/sqrt(2));
    s6 =  -1/sqrt(2)-(1i/sqrt(2));
    s7 =  0-1i;

    num_of_sym = length(rx_symbol);
    j = 1;
    for i = 1:num_of_sym

        classic_llrs = llr_approach(rx_symbol(i), noise_variance, s0,
s1, s2, s3, s4, s5, s6, s7);
        maxlog_llrs = max_log_approach(rx_symbol(i), noise_variance,
s0, s1, s2, s3, s4, s5, s6, s7);
        proposed_llrs = proposed_approach(rx_symbol(i),
noise_variance, s0, s1, s2, s3, s4, s5, s6, s7);
        decoded_llrs(1,j) = classic_llrs(1);
        decoded_llrs(1,j+1) = classic_llrs(2);
        decoded_llrs(1,j+2) = classic_llrs(3);

        decoded_llrs(2,j) = maxlog_llrs(1);
        decoded_llrs(2,j+1) = maxlog_llrs(2);
        decoded_llrs(2,j+2) = maxlog_llrs(3);

        decoded_llrs(3,j) = proposed_llrs(1);
        decoded_llrs(3,j+1) = proposed_llrs(2);
        decoded_llrs(3,j+2) = proposed_llrs(3);
        j = 3 * i + 1;
    end
end

function [Pri] = dist_cal(r_sym, con_sym, nois_var)
   compx = r_sym - con_sym;
   abs_val = (abs(compx))^2;
   Pri =  -(abs_val/(2*nois_var));
   %computes -(|r - s|^2)/(2*sigma2)
end

function [decoded_llr] = llr_approach(r_sym, noise_var, s0, s1, s2,
s3, s4, s5, s6, s7)
```

```
    p0 = exp(dist_cal(r_sym, s0, noise_var));
    p1 = exp(dist_cal(r_sym, s4, noise_var));
    p2 = exp(dist_cal(r_sym, s6, noise_var));
    p3 = exp(dist_cal(r_sym, s2, noise_var));

    p4 = exp(dist_cal(r_sym, s1, noise_var));
    p5 = exp(dist_cal(r_sym, s5, noise_var));
    p6 = exp(dist_cal(r_sym, s7, noise_var));
    p7 = exp(dist_cal(r_sym, s3, noise_var));

    num = p0 + p1 + p2 + p3;
    denum = p4 + p5 + p6 + p7;
    decoded_llr(1) = log(num/denum);   %for bit b0

    p0 = exp(dist_cal(r_sym, s4, noise_var));
    p1 = exp(dist_cal(r_sym, s1, noise_var));
    p2 = exp(dist_cal(r_sym, s0, noise_var));
    p3 = exp(dist_cal(r_sym, s5, noise_var));

    p4 = exp(dist_cal(r_sym, s6, noise_var));
    p5 = exp(dist_cal(r_sym, s2, noise_var));
    p6 = exp(dist_cal(r_sym, s3, noise_var));
    p7 = exp(dist_cal(r_sym, s7, noise_var));

    num = p0 + p1 + p2 + p3;
    denum = p4 + p5 + p6 + p7;
    decoded_llr(2) = log(num/denum);  %for bit b1

    p0 = exp(dist_cal(r_sym, s0, noise_var));
    p1 = exp(dist_cal(r_sym, s1, noise_var));
    p2 = exp(dist_cal(r_sym, s2, noise_var));
    p3 = exp(dist_cal(r_sym, s3, noise_var));

    p4 = exp(dist_cal(r_sym, s4, noise_var));
    p5 = exp(dist_cal(r_sym, s6, noise_var));
    p6 = exp(dist_cal(r_sym, s7, noise_var));
    p7 = exp(dist_cal(r_sym, s5, noise_var));

    num = p0 + p1 + p2 + p3;
    denum = p4 + p5 + p6 + p7;
    decoded_llr(3) = log(num/denum);  %for bit b2

end
```

```
function [decoded_llr] = max_log_approach(r_sym, noise_var, s0, s1,
s2, s3, s4, s5, s6, s7)

    q(1) = dist_cal(r_sym, s0, noise_var);
    q(2) = dist_cal(r_sym, s4, noise_var);
    q(3) = dist_cal(r_sym, s6, noise_var);
    q(4) = dist_cal(r_sym, s2, noise_var);

    r(1) = dist_cal(r_sym, s1, noise_var);
    r(2) = dist_cal(r_sym, s5, noise_var);
    r(3) = dist_cal(r_sym, s7, noise_var);
    r(4) = dist_cal(r_sym, s3, noise_var);

    decoded_llr(1) = max(q) - max(r);

    q(1) = dist_cal(r_sym, s4, noise_var);
    q(2) = dist_cal(r_sym, s0, noise_var);
    q(3) = dist_cal(r_sym, s1, noise_var);
    q(4) = dist_cal(r_sym, s5, noise_var);

    r(1) = dist_cal(r_sym, s6, noise_var);
    r(2) = dist_cal(r_sym, s2, noise_var);
    r(3) = dist_cal(r_sym, s3, noise_var);
    r(4) = dist_cal(r_sym, s7, noise_var);

    decoded_llr(2) = max(q) - max(r);

    q(1) = dist_cal(r_sym, s0, noise_var);
    q(2) = dist_cal(r_sym, s1, noise_var);
    q(3) = dist_cal(r_sym, s2, noise_var);
    q(4) = dist_cal(r_sym, s3, noise_var);

    r(1) = dist_cal(r_sym, s4, noise_var);
    r(2) = dist_cal(r_sym, s6, noise_var);
    r(3) = dist_cal(r_sym, s7, noise_var);
    r(4) = dist_cal(r_sym, s5, noise_var);

    decoded_llr(3) = max(q) - max(r);
end

function [decoded_llr] = proposed_approach(r_sym, noise_var, s0, s1,
s2, s3, s4, s5, s6, s7)

   real_r = real(r_sym);
   imag_r = imag(r_sym);
```

```
  if (real_r > 0)
       if ((imag_r < 0) & (abs(real_r) > abs(imag_r)))% Region R2 for
bit b0
           q0 = dist_cal(r_sym, s0, noise_var);
           q1 = dist_cal(r_sym, s2, noise_var);
           r4 = dist_cal(r_sym, s5, noise_var);
           r5 = dist_cal(r_sym, s7, noise_var);

           q2_0 = dist_cal(r_sym, s1, noise_var);
           q2_1 = dist_cal(r_sym, s3, noise_var);
       else
           q0 = dist_cal(r_sym, s0, noise_var); % Region R1 for bit b0
           q1 = dist_cal(r_sym, s4, noise_var);
           r4 = dist_cal(r_sym, s1, noise_var);
           r5 = dist_cal(r_sym, s5, noise_var);

           q2_0 = dist_cal(r_sym, s0, noise_var);
           q2_1 = dist_cal(r_sym, s1, noise_var);
       end
   else
       if ((imag_r > 0) & (abs(real_r) > abs(imag_r)))% Region R4 for
bit b0
           q0 = dist_cal(r_sym, s4, noise_var);
           q1 = dist_cal(r_sym, s6, noise_var);
           r4 = dist_cal(r_sym, s1, noise_var);
           r5 = dist_cal(r_sym, s3, noise_var);

           q2_0 = dist_cal(r_sym, s0, noise_var);
           q2_1 = dist_cal(r_sym, s2, noise_var);
       else                                      % Region R3 for bit b0
           q0 = dist_cal(r_sym, s2, noise_var);
           q1 = dist_cal(r_sym, s6, noise_var);
           r4 = dist_cal(r_sym, s3, noise_var);
           r5 = dist_cal(r_sym, s7, noise_var);

           q2_0 = dist_cal(r_sym, s2, noise_var);
           q2_1 = dist_cal(r_sym, s3, noise_var);
       end
   end
   num = exp(q0) + exp(q1);
   denum = exp(r4) + exp(r5);
   decoded_llr(3) = log(num/denum);  %LLR for bit b0
   if (imag_r > 0)
       if ((real_r > 0) & (abs(real_r) <= abs(imag_r)))%Region R1 for
bit b1
```

```
            q0 = dist_cal(r_sym, s0, noise_var);
            q1 = dist_cal(r_sym, s1, noise_var);
            r4 = dist_cal(r_sym, s6, noise_var);
            r5 = dist_cal(r_sym, s7, noise_var);

            r2_0 = dist_cal(r_sym, s4, noise_var);
            r2_1 = dist_cal(r_sym, s5, noise_var);
        else                                    %Region R4 for bit b1
            q0 = dist_cal(r_sym, s0, noise_var);
            q1 = dist_cal(r_sym, s4, noise_var);
            r4 = dist_cal(r_sym, s2, noise_var);
            r5 = dist_cal(r_sym, s6, noise_var);

            r2_0 = dist_cal(r_sym, s4, noise_var);
            r2_1 = dist_cal(r_sym, s6, noise_var);
        end
    else
        if ((real_r < 0) & (abs(real_r) <= abs(imag_r)))%Region R3 for
bit b1
            q0 = dist_cal(r_sym, s4, noise_var);
            q1 = dist_cal(r_sym, s5, noise_var);
            r4 = dist_cal(r_sym, s2, noise_var);
            r5 = dist_cal(r_sym, s3, noise_var);

            r2_0 = dist_cal(r_sym, s6, noise_var);
            r2_1 = dist_cal(r_sym, s7, noise_var);
        else                                      %Region R2 for bit b1
            q0 = dist_cal(r_sym, s1, noise_var);
            q1 = dist_cal(r_sym, s5, noise_var);
            r4 = dist_cal(r_sym, s3, noise_var);
            r5 = dist_cal(r_sym, s7, noise_var);

            r2_0 = dist_cal(r_sym, s5, noise_var);
            r2_1 = dist_cal(r_sym, s7, noise_var);
        end
    end
    num = exp(q0) + exp(q1);
    denum = exp(r4) + exp(r5);
    decoded_llr(2) = log(num/denum);    %LLR for bit b1
    num = exp(q2_0) + exp(q2_1);
    denum = exp(r2_0) + exp(r2_1);
    decoded_llr(1) = log(num/denum);    %LLR for bit b0
end
```

```
function [corrupted_signal] = add_awgn(tx_signal,SNR,sigma2)
    corrupted_signal = awgn(tx_signal,SNR);
end

function [output_vector] = psk_8(tx_vector)

    s0 =  1/sqrt(2)+(1i/sqrt(2));
    s1 =  0+1i;
    s2 =  -1+0i;
    s3 =  (-1/sqrt(2))+(1i/sqrt(2));
    s4 =  1 + 0i;
    s5 =  (1/sqrt(2))-(1i/sqrt(2));
    s6 =  -1/sqrt(2)-(1i/sqrt(2));
    s7 =  0-1i;
    symbols = [s0 s1 s2 s3 s4 s5 s6 s7];

    alphabets = symbols(1,:);
    q = tx_vector(1:3:end)*(2^0) + tx_vector(2:3:end) * (2^1) +
tx_vector(3:3:end) * (2^2) + 1; %collect msb and lsb separately
    output_vector = alphabets(q);
end

function ERR_LDPC_64800b_R12_LAYERED_SPA_I5(RunID)

  c = clock;
  disp(datestr(datenum(c(1),c(2),c(3),c(4),c(5),c(6))));
  % == LDPC SETTINGS ===================================
  TxRx.Sim.name = 'ERR_LDPC_64800b_R12_LAYERED_SPA_I5';
  TxRx.Sim.nr_of_channels = 100; % 1k for good results.
  TxRx.Sim.SNR_dB_list = [0:0.5:6];
  load('codes/dvb_s2/LDPC_DVBS2_64800b_R12.mat'); % load code

  % == EXECUTE SIMULATION ==============================
  sim_LDPC(RunID,TxRx,LDPC)
  c = clock;
  disp(datestr(datenum(c(1),c(2),c(3),c(4),c(5),c(6))));
return
```

A.2 VHDL code for compare select block

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity compare_select is
    Port ( Rx : in signed (9 downto 0);
           Ry : in signed (9 downto 0);
```

```
            clk : in STD_LOGIC;
            b0_num_symbols : out STD_LOGIC_VECTOR (5 downto 0);
            b0_denum_symbols : out STD_LOGIC_VECTOR (5 downto 0);
            b1_num_symbols : out STD_LOGIC_VECTOR (5 downto 0);
            b1_denum_symbols : out STD_LOGIC_VECTOR (5 downto 0);
            b2_num_symbols : out STD_LOGIC_VECTOR (5 downto 0);
            b2_denum_symbols : out STD_LOGIC_VECTOR (5 downto 0));
end compare_select;

architecture Behavioral of compare_select is

begin
    Process(clk) begin

    If(clk'event And clk = '1')then

       If(Rx > 0)then
          If((Rx > 0) And (Rx > Ry)) then
             b0_num_symbols <= "000010"; -- S0 and S2
             b0_denum_symbols <= "101111";  -- S5 and S7
             b2_num_symbols <= "000010"; -- S0 and S2
          else
             b0_num_symbols <= "000100"; -- S0 and S4
             b0_denum_symbols <= "001101";  -- S1 and S5
             b2_num_symbols <= "000001"; -- S0 and S1
          end if;
       else
          If((Ry > 0) And (Rx > Ry)) then
            b0_num_symbols <= "100110";  -- S4 and S6
            b0_denum_symbols <= "001011"; -- S1 and S3
            b2_num_symbols <= "000010"; -- S0 and S2
          else
            b0_num_symbols <= "010110"; -- S2 and S6
            b0_denum_symbols <= "011111";  -- S3 and S7
            b2_num_symbols <= "010011"; -- S2 and S3
          end if;
       end if;

       If(Ry > 0)then
           If((Ry > 0) And (Rx <= Ry)) then
               b1_num_symbols <= "000001"; -- S0 and S1
               b1_denum_symbols <= "110111";  -- S6 and S7
               b2_denum_symbols <= "100101";  -- S4 and S5
           else
               b1_num_symbols <= "000100"; -- S0 and S4
               b1_denum_symbols <= "010110";  -- S2 and S6
               b2_denum_symbols <= "100110";  -- S4 and S6
           end if;
        else
           If((Ry < 0) And (Rx <= Ry)) then
               b1_num_symbols <= "100101"; -- S4 and S5
               b1_denum_symbols <= "010011";  -- S2 and S3
```

```
                b2_denum_symbols <= "110111";  -- S6 and S7
            else
                b1_num_symbols <= "001101"; -- S1 and S5
                b1_denum_symbols <= "011111";  -- S3 and S7
                b2_denum_symbols <= "101111";  -- S5 and S7
            end if;
        end if;
    end if;
  end process;

end Behavioral;
```

A.3 AWGN (Additive White Gaussian noise)

Additive white Gaussian noise (AWGN) is a basic noise model used to mimic random processes that occur in communication medium. Each word denotes specific characteristic of noise:

- Additive means effect of noise on the signal is additive.
- White means the power of noise is uniform over a wide range of frequencies. In the frequency domain graph, the power remains constant for wide range of frequencies.
- Gaussian means the amplitude of noise follows Gaussian distribution in the time domain with mean μ and variance σ^2.

Bibliography

1. R. G. Gallager, “Low-Density Parity-Check codes,” Ph.D. dissertation, MIT press, Cambridge, MA, 1963.
2. I. Land, P. A. Hoeher, and U. Sorger, “Log-likelihood values and Monte Carlo simulation: some fundamental results,” in Proc. Int. Symp. on Turbo Codes and Related Topics, Brest, France, Sept 2000.
3. ETSI EN 302 307, v. 1.1.2 Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications June 2006.
4. IEEE 802.16 Working Group, “Part 16: Air in-terface for fixed and mobile broadband wireless access systems”, IEEE P802.16e/D8, May 2005.
5. ITU-T G.9960, Unified high-speed wire-line based home networking transceiver.
6. S. Allpress, C. Luschi, and S. Felix, “Exact and approximated expressions of the log-likelihood ratio for 16 QAM signals”, in Proc. 38th Asilomar Conference on Signal, System and Computer, Nov.2004.
7. Seung H. Choi, Cheon In Oh, D. Oh, D. Change, “The Mapping and demapping algorithms for higher modulation of DVB-S2 systems,” in Proc. Asia-Pacific Conference on Communications, Aug. 2006.
8. I. Lee, D. Chang and D. Oh, “Multi-level modulation LDPC decoding algorithm for new generation DVB-S2 system” in Proc. 24th AIAA International Communications Satellite Systems Conference, June. 2006.

9. Jang Woong Park; Myung Hoon Sunwoo; Pan Soo Kim; Dae-Ig Chang, Low Complexity Soft-Decision Demapper for High Order Modulation of DVB-S2 system International SoC Design Conference, ISOCC'08, vol. 02, pp. 37-40, Nov. 2008.
10. Jianing Su; Zhenghao Lu; Xiaopeng Yu; Changhui Hu, A Novel Low Complexity Soft-decision Demapper for QPSK 8-PSK Demodulation of DVB-S2 system, International Conference of Electron Devices and Solid-State Circuits (EDSSC'11), pp. 17-18 Tianjin (China), Nov. 2011.
11. Barre, A.; Boutillon, E.; Bias, N.; Diaz, D. "A polar-based demapper of 8PSK demodulation for DVB-S2 systems", Signal Processing Systems (SiPS), 2013 IEEE Workshop on Signal Processing Systems.
12. Jea Hack Lee; Myung Hoon Sunwoo; Pan Soo Kim; Dae-Ig Chang, "Low Complexity soft-decision demapper for DVB-S2 using phase selection method", International Conference on Ubiquitous Information Management and Communication (ICUIMC'11), Article No. 45.
13. Qi Wang, Qiuliang Xie, Zhaocheng Wang, Sheng Chen, Lajos Hanzo, "A Universal Low – Complexity Symbol-to-Bit Soft Demapper", IEEE Transactions on Vehicular Technology, VOL. 63, NO. 1, JANUARY 2014.
14. C. Roth, P. Meinerzhagen, C. Studer, and A. Burg, "A 15.8 pJ/bit/iter Quasi-Cyclic LDPC Decoder for IEEE 802.11n in 90nm CMOS," Proc. IEEE Asian Solid-State Circuit Conference (A-SSCC), Nov. 2010.

Suhas Prahalada suhasp47@yahoo.co.in +1 778 679 9630

Profile Summary

- Sound knowledge of logic design and digital circuit design.
- Programming experience with C, C++, Java, MATLAB.
- Working experience with post silicon validation and ASIC verification.

Education

M.Eng Electrical Engineering Jan 2013 - Present
University of Victoria Victoria, BC
GPA (To date) **7.86/9**

B.Eng Electronics and Communication Sept 2007- Sept 2011
Vishveshvaraya Technological University Bangalore, India
Percentage **70% (First Class)**

Professional Experience

BROADCOM CORPORATION, RICHMOND, BC
Hardware IC Design Intern (Aug 2013 to April 2014)
ASCENDUM SOLUTIONS, BANGALORE, INDIA
Software Engineer (Oct 2011 to Nov 2012)
BOSCH Pvt Ltd, Bangalore
Intern (Jan 2011 to May 2011)
Project: **Calibration equipment for FEM-AI & FEM-P using CAN Interface**

Technical Skills

Electronics

- Solid understanding of logic design and digital electronics.
- Good knowledge of FPGA and ASIC design flow.
- Good understanding of synthesis, Static Timing Analysis.
- HDL's: VHDL, Verilog.
- Assertion Language: PSL.
- Good exposure to CAD tools like Xilinx ISE, MATLAB and Modelsim.

Computer Science

- Good understanding of OOP and other programming paradigms.
- **Programming Language and Scripting:** C, C++, Perl, shell scripting, PHP, JAVA, VBScript.
- **Platforms/OS:** Windows, Linux.
- **Tools:** Netbeans, Eclipse, GIT, SVN.