CHAPTER 4

MODELING AND PERFORMANCE ANALYSIS

In a digital transmission system, an error occurs when a bit is alters in between transmission and reception in a channel. A binary 1 is transmitted and a binary 0 is received or a binary 0 is transmitted and a binary 1 is received. There are two types of errors that can occur: single-bit errors and burst errors [98]. A single-bit error is an isolated error that alters one bit but does not affect adjacent bits. A burst error is an error in a contiguous sequence of B bits in which the first and last bits and any number of intermediate bits are received in error.

A single-bit error may occur in the presence of white noise, due to a slight random deterioration of the signal-to-noise ratio, which is sufficient to confuse the receiver's decision of a single bit. Burst errors can be caused by impulse noise or fading in a mobile wireless environment and are more common and more difficult to deal with. The effects of burst errors are larger at higher data rates.

4.1 Errors in a Digital Communication System

Bit errors occur in digital communication systems due to intrinsic or extrinsic factors [99]. Intrinsic errors are due to the components, design and implementation of a link. They are caused due to internal noise sources, poor electrical connections, and sometimes receiver sampling error. In optical links the errors occur mainly because of the physical components such as optical driver, optical receiver, connectors, optical fiber, etc. Errors are also caused due to optical attenuation and optical dispersion. One of the prevalent causes of random or noise-induced errors is the optical receiver. The extrinsic sources of error are caused by external sources and influences. These include power supply noise, electrostatic discharge, electromagnetic interference and connector vibrations.

4.1.1 Shannon Capacity Formula

Nyquist's formula $C = 2B\log_2 M$ where M is the number of discrete signal or voltage levels, indicates that, doubling the band-width doubles the data rate, all other factors being equal [98]. Higher the data rate, more are the errors. These concepts can be explained using a

formula developed by the mathematician Claude Shannon. For a given level of noise, a greater signal strength would improve the ability to receive data correctly in the presence of noise.

SNR is the most common performance measure characteristic of a digital communication system. Typically, this ratio is measured at the output of the receiver and is thus directly related to the data detection process [100]. Signal-to-noise ratio is a term for the ratio of the power in a signal to the power present in the noise:

Thus, SNR in decibels can be expressed as

$$SNR(dB) = 10 \cdot \log\left(\frac{P_s}{P_n}\right) = 20 \cdot \log\left(\frac{A_s}{A_n}\right) \quad (4.1)$$

where P_s is average signal power and P_n is average noise power, and A is root mean square (RMS) amplitude for signal and noise. SNRs are usually expressed in terms of the logarithmic decibel scale, since many signals have a very wide dynamic range. Thus the SNR is 10 times the base-10 logarithm of the power ratio and if the signal and noise is measured across the same impedance then SNR can be obtained by calculating 20 times the base-10 logarithm of the amplitude ratio.

Shannon's result is given by $C = B\log_2(1+SNR)$, where C is the capacity of the channel in bits per second and B is the bandwidth. Although Shannon formula represents the theoretical maximum that can be achieved, in reality, much lower rates are achieved because the formula assumes only white noise and other sources of noise such as impulse noise, attenuation distortion or delay distortion is not accounted. Even in an ideal white noise environment, the present technology has not been able to achieve Shannon capacity due to encoding issues, such as coding length and complexity. The Shannon's formula provides a yardstick for the measurement of performance of practical communication schemes.

From the formula, it can be observed that the data rate could be increased by increasing either signal strength or bandwidth. However, as the signal strength increases, the effects of nonlinearities in the system also increases leading to an increase in intermodulation noise. Similarly, since the noise is assumed to be white, the wider the bandwidth, the more noise is admitted in the system. Thus, as B increases, SNR decreases.

The parameter E_b/N_0 is the ratio of signal energy per bit to noise power density per Hertz. It is a standard quality measure for digital communication system performance. The ratio E_b/N_0 is important because the bit error rate for digital data is a function of this ratio.

4.1.2 Bit Error Rate

BER is the most fundamental measure of system performance. It is defined as the number of bits received in error, divided by the total number of bits received [100]. It is usually expressed as 10^{-x}. For example, a transmission system might have a BER of 10^{-6}, meaning that on an average, 1 out of every of 1000000 bits transmitted exhibits an error. The BER performance is affected by factors such as signal-to-noise ratio and distortion. However, the quality of the link is ultimately defined by the ability to receive error-free information. The BER indicates of how often a packet or other data unit needs to be retransmitted because of an error.

4.1.3 Packet Error Rate

The PER is the number of incorrectly received data packets divided by the total number of received packets. A packet is declared incorrect even if one bit is erroneous.

4.1.4 Binary Symmetric Channel

The BSC is a binary channel, which can transmit only one of two symbols ether 0 or 1 [101]. It is one of the simplest noisy channels to analyze. The probability for having an error is denoted as p and the probability for no errors is 1 - p. It is known as symmetric because the probability of error is independent of the transmitted symbol. Figure 4.1.1 gives the graphical representation of BSC, where X is the input variable and Y is the output variable.

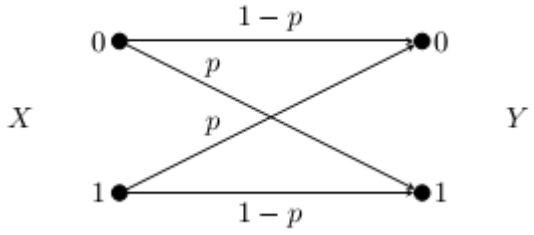


Figure 4.1.1 Graphical representation of BSC

4.1.5 Additive White Gaussian Noise Channel

AWGN is the commonly used and generally accepted model for thermal noise in communication channels [100]. It can be represented as in Figure 4.1.2 where s(t) is transmitted signal and n(t) is background noise and r(t) = s(t) + n(t), is the received signal.

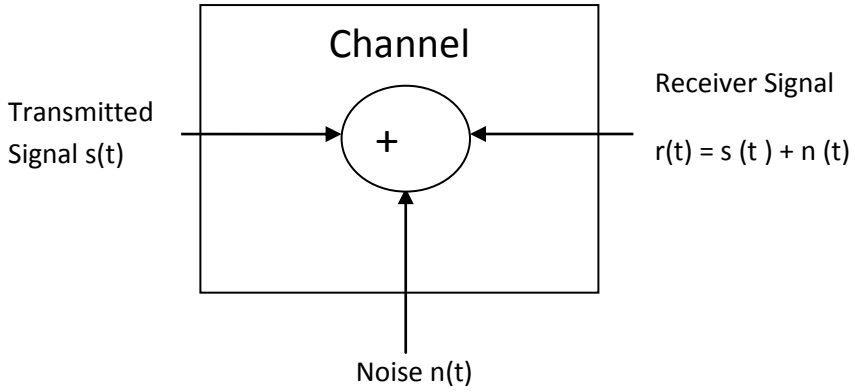


Figure 4.1.2 AWGN channel model

4.1.6 BER of BPSK (Binary Phase Shift Keying)

Phase-shift keying (PSK) is a technique of digital communication in which the phase of a transmitted signal is varied to convey information. There are several ways to accomplish PSK. BPSK is the simplest form of PSK. It uses two phases which are separated by 180° and hence also termed as 2-PSK [102]. This modulation is the most robust of all the PSKs since it requires serious distortion to make the demodulator reach an incorrect decision.

The BER of BPSK in AWGN can be calculated as:

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \text{ or } P_b = 1/2 erfc\left(\sqrt{\frac{E_b}{N_o}}\right) \quad (4.2)$$

4.2 Error Detection using CRC

CRCs have a long history of use for error detection in computing [103]. Error correction codes provide a means to detect and correct errors introduced by a transmission channel. Two main categories of code exist: block codes and convolutional codes. Both the codes introduce redundancy by adding parity symbols to the message data. Cyclic codes are an important class of linear (n,k) block codes 'C' which are said to be cyclic if for each of its code vectors c=(C_0, C_1, C_2, …….. C_{n-1}) in 'C' the i^{th} right cyclic shift c'= (C_{n-i}, C_{n-i+1}, … C_0 ,C_1, C_{n-i-1}) is a code vector also in 'C'. CRC codes are a subset of cyclic codes which are also a subset of linear block codes. CRC provides error control coding and protection to data by introducing some redundancy in the data in a controlled manner. It is a very effective way of detecting transmission errors during transmissions in various types of networks and it is also used in storage applications. Common CRC polynomials are able to detect following types of errors: i. All single bit error, ii. All double bit errors, iii. All odd number of errors having sufficient constraint length, iv. Any burst error for which the burst length is less than the polynomial length, v. Most large burst errors [104].

CRC uses binary alphabets, 0 and 1. Arithmetic is based on GF(2) i.e. modulo-2 addition (logical XOR) and multiplication (logical AND). The coding scheme uses systematic codes. Suppose m(x) is the message polynomial, c(x) the code word polynomial and g(x) the generator polynomial. We have c(x)= m(x)g(x) which is also written using the systematic form as c(x)=

$m(x)x^{n-k} + r(x)$, where r(x) is the remainder of the division of $m(x)x^{n-k}$ by g(x) and r(x) represents the CRC bits. The transmitted message c(x) contains k-information bits followed by 'n-k' CRC bits, for example $c(x) = m_{k-1}x^{n-1}+\ldots\ldots+ m_0x^{n-k}+ r_{n-k-1}x^{n-k-1}+\ldots\ldots+ r_0$. So encoding is straightforward: multiply m(x) by x^{n-k} , that is , append 'n-k' bits to the message, calculate the CRC bits by dividing $m(x)x^{n-k}$ by g(x), and append the resulting 'n-k' CRC bits to the message. The same algorithm can be used for decoding. If c'(x) is the received message, then no error or undetectable errors have occurred if c'(x) is multiple of g(x) , which is equivalent to determining that if $c'(x)x^{n-k}$ is a multiple of g(x), that is, if the remainder of the division from $c'(x)x^{n-k}$ by g(x) is 0[14]**.** An error is declared to have occurred if there is any discrepancy between these two sets of parity bits and thus indicates the presence of transmission errors in the received data.

However, as with all digital signature schemes, there is a small, but finite, probability that a data corruption that inverts a sufficient number of bits in just the right pattern will occur and lead to an undetectable error [105]. The minimum number of bit inversions required to achieve such undetected errors, known as Hamming Distance (HD) is a central issue in the design of CRC polynomials. However, there may be transmission errors residing in the received frame that are not detected by this procedure. Whenever channel noise affects both the information block ***i*** and the block *r* of parity bits in such a way that the received frame $\hat{c}=[\acute{r}, \hat{\imath}]$ satisfies $\acute{r}(x) = (x^p . \hat{\imath} (x))$ mod g(x), the errors cannot be detected by parity checking. Authors have proposed solution for the said problem using double CRC in the payload of the Ethernet frame [106].

There are four major CRC implementation solutions. Many researchers have done comparative studies over CRC implementation [107]. CRC implementation can use either hardware or software solutions. In the traditional hardware implementation, a simple shift register circuit performs the computations by handling the data one bit at a time and parallel implementation by handling data in one word(n-bit) at a time [108,109,37,41]. Software implementations of CRC encoding/ decoding do not resort to dedicated hardware requirements; their applicability, however, is limited to lower encoding rates. In traditional hardware implementations, a LFSR performs the computations by handling the data one bit at a time. With the increasing demands of high transmission rate requirement in various

applications, the need of a simple and systematic method to rapidly calculate the CRC has resulted. The most frequently used CRC digest computation algorithm in iSCSI and Ethernet, the Simultaneous Multiply and Divide (SMD), is derived from the long division algorithm. Sample implementations are provided of both algorithms in [110,111].

4.2.1 Software Solution

The CRC algorithm can always be implemented as a software algorithm on a standard CPU, with all the flexibility of reprogramming. Also, the software solution will be much cheaper, since a CPU is present in most communication network terminals. However, the computation speed will be lower, since no general purpose CPU can achieve the same throughput as dedicated hardware. Kounavis et al. implemented novel lookup table based algorithm based on 'slicing-by-x' algorithm which doubles the performance of existing software-based, table driven CRC implementations [112]. Software implementation, which requires many steps of the polynomial division, is typically slower than other codes. Nguyen et al. implemented fast CRCs as well as an effective technique to implement them without using lookup table, to eliminate or to greatly reduce many steps of the polynomial division [113].

The CRC algorithm is straightforward to implement in software. However, it requires considerable CPU bandwidth to implement the basic requirements, such as shift, bit test and XOR. Moreover, CRC calculation is an iterative process and additional software overhead for data transfer instructions puts enormous burden on the MIPS requirement of a microcontroller. The CRC engine in 'dsPIC33E/PIC24E' devices calculates the CRC checksum without CPU intervention and it is much faster than the software implementation. The CRC engine consumes only half of an instruction cycle per bit for its calculation as the frequency of the CRC shift clock is twice that of the dsPIC33E/PIC24E instruction clock cycle. For example, the CRC hardware engine takes only 64 instruction cycles to calculate a CRC checksum on a message that is 128 bits (16x8) long. If the same calculation is implemented in software, it will consume more than a thousand instruction cycles, even for an optimized piece of code.

4.2.2 Traditional Hardware Solution

LFSR with serial data feed has been used since the sixties to implement the CRC algorithm. Like all hardware implementations, this method simply performs a division and then the remainder which is the resulting CRC checksum, is stored in the registers after each clock cycle. The registers can then be read with the help of enabling signals. The main advantages are simplicity and low power dissipation. This method gives much higher throughput than the Software solution however, the speed requirements of today's network nodes cannot be fulfilled. Since fixed logic is used there is no possibility of reconfiguring the architecture or changing the generator polynomial[114].

4.2.3 Parallel Solution

Parallelism improves the computational speed in CRC generating hardware. The speed is increased by a factor between 4 and 6 when using a parallelism of 8 bits. By using fixed logic, implemented as parallelized hardware, CRC generation can achieve wire speed and therefore it is the pre-dominant method used in computer networks. Like any other combinatorial circuit, parallel CRC hardware could be synthesized with only two levels of gates. Thus, this implies a huge number of gates. Also, the minimization of the number of gates is an NP-hard optimization problem. Therefore, for realization of complex circuits, heuristics is used or one seeks customized solutions. Campobella et al. derive a recursive formula that can be used to deduce the parallel CRC circuits as in modern synthesis tools [41].

Also, Albertengo and his team designed the parallel CRC encoders based on digital system theory and z-transforms, which allows designers to derive the logic equations of the parallel encoder circuit for any generator polynomial [31].By inspecting the operations of the single-input LFSR, Pei[32], derived the state transition equation for the parallel CRC circuit by merging a number of the shift and modulo-2 (XOR) operations together within a single clock cycle [115]. Also some attempts of fast CRC computation, which is based on the observation that only a small portion residing in the beginning of an Ethernet frame is changed during the hops for updating the relevant source and destination address over the network. Here, the fast CRC update only calculates the changed portion of a frame and performs a single step update

afterwards to obtain the new CRC code. This method dramatically improves the throughput of CRC recalculation which can support a throughput of about 56 Gbps[116]. In above designs if the CRC polynomial is changed or a new protocol is added, new changed hardware must be installed in the network terminal. The lack of flexibility makes this architectures non suitable for use in a protocol processor. Hence, very often new hardware schemes which compute the transition and control matrix of a parallel cyclic redundancy checksum are proposed. This opens possibilities for parallel high-speed cyclic redundancy checksum circuits that reconfigure very rapidly to new polynomials [117].

4.2.4 Configurable Hardware

Configurable hardware are implemented using Look-Up-Tables (LUT). This implementation can be modified by using a larger or smaller LUT. If the size of the LUT is reduced the hardware-cost in terms of power consumption and area will be reduced but at the same time the Combinational Network will be increased so the effect will be cancelled. The optimal solution has not been derived. Another, novel implementation method is the Radix-16 Configurable CRC Unit. Toal and his team proposed architecture using field reprogrammable chips so that it is fully flexible in terms of the polynomial deployed and the input port width. They synthesized circuit and mapped it to 130-nm UMC standard cell ASIC technology which is capable of supporting line speeds of 5 Gb/s [118].

4.3 Simulation Model for Error Detection using CRC

When the errors are introduced in Ethernet data streams using the experimental setup, the report of the number of packet errors generated can be obtained from the Report menu of the TSE design, however the user application is not able to catch the Ethernet frame which is in error, since the frame is discarded. Also the CRC field of the Ethernet frame is not read. Hence it is difficult to implement direct methods of monitoring the packet loss for the performance study of the Ethernet for the particular CRC polynomial. We have adopted indirect methods of quantifying the performance of the protocol. We have developed a Simulation Model in Matlab for BER performance evaluation, as illustrated in Figure 4.3.1.

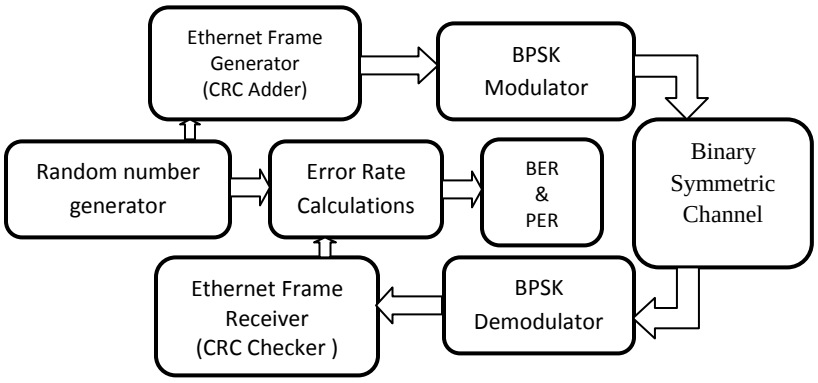


Figure 4.3.1 Matlab Simulation model for BER performance for BSC channel

This model can be used as a platform for Error Detection Analysis using CRC algorithms. A random number generator is used to generate a sequence of Ethernet frames of length varying from 64 bytes to 9600 bytes. The frames includes 4 bytes FCS, which is computed using Matlab code.

The transmitter modulates the frame bits using Binary Phase Shift Keying (BPSK), which is sent through a BSC. The amount of noise in the channel can be controlled. The receiver receives the noisy signal, demodulates it and produces a sequence of recovered bits. The FCS is recomputed at the receiver to find whether there was an error in the transmitted packet. Finally, we compare the received bits to the transmitted bits to get the BER and PER [119]. The model is user friendly and has a capability to vary the CRC polynomial, size of frames and number of frames. The platform can be used to test the performance of CRC error detection algorithms using different CRC polynomials. The Matlab code for this model is given in Appendix II. Also the study of BER performance for BPSK modulation in AWGN channel is performed.

4.4 Error Correction using LDPC codes

The data communication rates continue to increase due to bandwidth intensive applications such as video transport, data over IP, wireless base station, and medical applications[99]. Also increasing data rates entail more tight latency requirements, specifically

for full-duplex communication that involves streaming data like voice. Lower latency means less scope for error correction, which in turn puts more stringent requirements on the number of acceptable bit errors in the transmission system. LDPC codes have a performance close to Shannon limit coding gains when iteratively decoded [120]. They are used in several digital communication systems such as digital video broadcasting MIMO-WLAN (802.11n), (DVB-S2), WMAN (802.16e), mobile broadband wireless access (MBW A) (802.20) [121] and have a very good error correcting performance over a range of channels.

The main advantage of LDPC codes is that their performance is very close to the capacity for a lot of different channels and linear time complex algorithms are used for decoding. They are suited for implementations that make substantial use of parallelism. LDPC codes were first introduced by Gallager in his Ph.D. thesis in 1960. However, due to the computational effort in the implemention of coder and encoder and the introduction of Reed-Solomon codes, they were mostly ignored until 1990s. LDPC codes were independently rediscovered by MacKay [122,123] and by Luby et al. [124]. Since this time, much work has been carried out on the design of good codes and on the analysis of their performance [125,126,127]. Turbo codes and LDPC codes provide coding gains close to Shannon's limit [128,129,126]. LDPC codes however outperform turbo codes in terms of coding gain for large SNR [130,126] Also LDPC codes require less computational complexity and less number of iterations compared to turbo codes. Thus it results in higher throughput and lower power dissipation. Error correction algorithms are frequently implemented in hardware for fast processing to meet the real-time needs of communication systems [76].

4.4.1 Representation of LDPC Codes

LDPC can be represented in two ways. The first is using matrices and second is graphical representation [131].

Matrix Representation

Equation (4.3) defines a parity check matrix with dimension 'n × m' for a (8, 4) code.

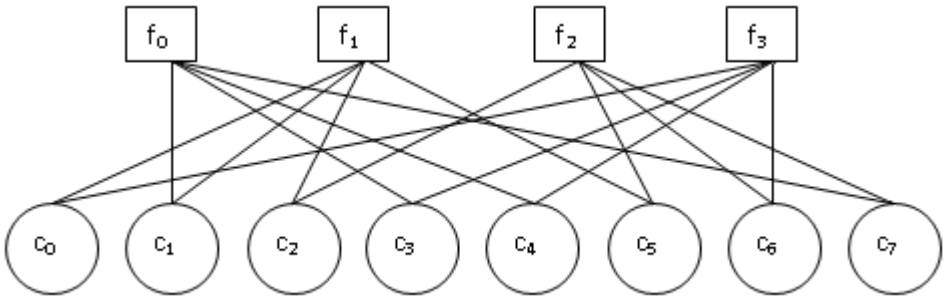
$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (4.3)$$

The number of 1's in each row is denoted by 'w_r' and 'w_c' represents the number of 1's in each for the column. The matrix can be called as low-density if the two conditions 'w_c << n' and 'w_r << m' are satisfied. Hence, in reality, the parity matrix should be very large.

Graphical Representation

Tanner introduced an effective graphical representation for LDPC codes. These graphs provide a complete representation of the code and they also help to describe the decoding algorithm. Tanner graphs are bipartite graphs. The nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The two types of nodes are called variable nodes (v-nodes) and check nodes (c-nodes). Figure 4.4.1 is an example for such a Tanner graph and represents the same code as the matrix in equation (4.3). The graph consists of 'm' check nodes (the number of parity bits) and 'n' variable nodes (the number of bits in a codeword). Check node 'f_i' is connected to variable node 'c_j' if the element 'h_{ij}' of 'H' is a 1.

c_nodes



v_nodes

Figure 4.4.1 Tanner graph corresponding to the parity check matrix in equation (4.3)

A LDPC code is called regular if 'w_c' is constant for every column and '$w_r = w_c \cdot (n/m)$' is also constant for every row. The example matrix given in equation (4.3) is regular with '$w_c = 2$' and '$w_r = 4$'. Graphical representation shows the regularity of this code. The number of incoming edges is same for every 'v-node' and also for all the 'c-nodes'. If the numbers of 1's in each row or column are not constant, the code is called an irregular LDPC code. Conditions for the decoding algorithm to be optimal is graph remains cycle free and all the information in updating the decoded bits or messages is unrelated and originating from the distinct received bits[132]. However, once a cycle is encountered the algorithm becomes suboptimal. Thus, the performance of the decoder is improved by maximizing the cycle lengths in the course of the design of the parity check matrix 'H' subject to the constraints of the block size and required column/row weight profiles[133].

4.4.2 LDPC decoding algorithms

The LDPC codes can be decoded by a local message-passing algorithm on the graph, the Sum Product Algorithm (SPA).

Sum-Product Algorithm – Probability Domain

The parity check matrix has a dimension of (N-K) x N and consists of N-K "Check nodes and N "Bit nodes" representing parity check equations and code bits, respectively. Decoding is performed iteratively [134]. In each iteration, every Bit node passes a message to the Check nodes that are connected to it. In the next half iteration, each Check node sends a message to the Bit nodes. This message is a function of all the extrinsic information that it has received from the Bit nodes in the last part. Then, the codeword is checked for validity. It performs the iterations until it finds the valid code word or reaches the maximum number of the iterations. The following steps should be performed for all the 'is' and 'js' for which the element in the parity check matrix is a one (h_{ij}=1).

Step 0: Initialize 'q_{ji}' by:

$$q_{ji}(0) = 1 - p_i = P_r(x_i = +1 \mid y_i) = \frac{1}{1 + e^{-2y_i/\sigma^2}} \tag{4.4}$$

$$q_{ji}(1) = p_i = P_r(x_i = -1 \mid y_i) = \frac{1}{1 + e^{2y_i/\sigma^2}} \tag{4.5}$$

This step initialized the values of $Q(x)_{ij}$ which are set to a priori estimates of the received symbols 'y_i' obtained after hard-decision decoding of the received decoded vector. Here 'σ' is standard deviation of the noise over the AWGN channel.

Step 1: Horizontal stepping on 'r_{ji}' by:

$$r_{ji}(0) = 1/2 + 1/2 \prod_{i' \varepsilon R_{j \setminus i}} \left(1 - 2q_{ji'}\ (1)\right) \tag{4.6}$$

$$r_{ji}(1) = 1 - r_{ji}(0) \tag{4.7}$$

Here, the coefficients '$r_{ji}(x)$' are the values which are estimates that go from parity check node to the symbols nodes, in the bipartite graph. The probability associated with each combinations are added to calculate the estimates of '$r_{ji}(x)$'.

Step 2: Vertical stepping on 'q_{ji}':

$$q_{ji}(0) = K_{ji}(1 - p_i) \prod_{j' \varepsilon C_i} r_{j'i}(0) \tag{4.8}$$

$$q_{ji}(1) = K_{ji} p_i \prod_{j' \varepsilon C_{i \setminus j}} r_{j'i}(1) \tag{4.9}$$

where the constants K_{ji} are chosen to ensure that $q_{ji}(0) + q_{ji}(1) = 1$. Here the normalizing constants 'K_{ji}' satisfy the normalizing condition $\sum_x Q(x)_{ij}=1$. The $(1-p_i)$ and (p_i) are the coefficients corresponding to the received hard-decision vector.

Step 3: For all the i's compute:

$$Q_i(0) = K_i(1-p_i)\prod_{j\varepsilon C_i} r_{ji}(0) \tag{4.10}$$

$$Q_i(1) = K_i p_i \prod_{j\varepsilon C_i} r_{ji}(1) \tag{4.11}$$

where the constants 'K_i' are chosen to ensure that $Q_i(0) + Q_i(1) = 1$.

Step 4: For every row index i:

$$\hat{c}_i = \begin{cases} 1 \; if Q_i(1) > Q_i(0) \\ 0 \; else \end{cases} \tag{4.12}$$

If $\hat{c}H^T=0$, or if maximum number of iteration is reached then stop, else continue iterations from Step 1.

Sum-Product Algorithm- Log Domain

SPA used in decoding of LDPC codes requires a large number of multiplications of probabilities which makes the algorithm numerically unstable, specially for very long codes. It may be noted that the best performance is obtained for very long codes. This long block length, combined with the need for iterative decoding, introduces latency which is unacceptable in many applications. Thus, a log-domain version of the algorithm is preferred. We define the following log likelihood ratios as part of the decoding algorithm:

$$Lc_i = \log\left(\left(P_r(x_i = +1 \mid y_i)\right)/\left(P_r(x_i = -1 \mid y_i)\right)\right); \tag{4.13}$$

$$Lr_{ji} = \log(r_{ji}(0)/r_{ji}(1)); \tag{4.14}$$

$$Lq_{ji} = \log(q_{ji}(0)/q_{ji}(1)); \tag{4.15}$$

$$LQ_i = \log(Q_i(0)/Q_i(1)). \tag{4.16}$$

This algorithm iterates over columns and rows of parity check matrix H, and operates on nonzero entries by performing the following steps:

Step 0: Initialize Lq_{ji} by: $Lq_{ji} = Lc_i = 2y_i/\sigma^2$; which initialized the values of coefficients $Q(x)_{ij}$ over logarithmic scale for the received symbols 'y_i' obtained after hard-decision decoder of the received decoded vector having 'σ' as standard deviation of the noise over the channel.

Step 1: Evaluate ***Lr$_{ji}$*** by: $Lr_{ji} = \left(\prod_{i'\varepsilon R_{j\backslash i}} \alpha_{ji'}\right).\phi\left(\sum_{i'\varepsilon R_{j\backslash i}} \phi(\beta_{ji'})\right)$; (4.17)

where, $\alpha_{ji} = sign(Lq_{ji})$, $\beta_{ji} = \left\|Lq_{ji}\right\|$, $\phi(x) = -\log(\tanh(x/2)) = \log\left(e^x + 1/e^x - 1\right)$.

Step 2: $Lq_{ji} = Lc_i + \sum_{j'\varepsilon C_i \backslash j} Lr_{j'i}$. Step 3: $LQ_i = Lc_i + \sum_{j\varepsilon C_i} Lr_{ji}$. (4.18)

Step 4: For every row index i: $\hat{c}_i = \begin{Bmatrix} 1 & if\ LQ_i < 0 \\ 0 & else \end{Bmatrix}$. (4.19)

If $\hat{c}H^T$=0, or if maximum number of iteration is reached then stop, else continue iterations from Step 1.

Min-Sum Algorithm

Consider the update equation for 'Lr_{ji}' in the Sum-Product algorithm:

$$Lr_{ji} = \left(\prod_{i'\varepsilon R_{j\backslash i}} \alpha_{ji'}\right).\phi\left(\sum_{i'\varepsilon R_{j\backslash i}} \phi(\beta_{ji'})\right) \quad (4.20)$$

The 'ϕ (x)' is a function which is decreasing for the values of x > 0. It is intuitive that the term corresponding to the smallest β_{ji} in the above summation dominates, so that the second term in above equation : $\phi\left(\sum_{i'\varepsilon R_{j\backslash i}} \phi(\beta_{ji'})\right) = \phi(\phi(\min_{i'} \beta_{ji'})) = \min_{i'} \beta_{ji'}$. The second equality follows from ϕ (ϕ (x)) = x. Thus the Min-Sum algorithm is the same as SPA in which Step(1) is replaced by this equation: Step 1': $Lr_{ji} = (\prod_{i'\varepsilon R_{j\backslash i}} \alpha_{ji'}).(\min_{i'\varepsilon R_{j\backslash i}} \beta_{ji'})$. Because of the approximation in this equation, there is a degradation in the performance of Min-Sum compared to SPA.

Modified Min-Sum Algorithm

Scaling the soft information during the decoding using min-sum algorithm, results in better performance. Scaling slows down the convergence of iterative decoding and reduces the overestimation error compared to SPA. In this algorithm, it is enough to change the Step 2 in Min-Sum Algorithm with

$$[Step2':]Lq_{ji} = (Lc_i + \sum_{j' \varepsilon C_i \setminus j} Lr_{j'i}) * \gamma \quad (4.21)$$

in which γ is the scaling factor. Min-Sum algorithm which is an approximation of the Sum-Product, suffers from some performance loss because of the approximations. On the other hand, Modified Min-Sum shows even better performance than SPA in some SNR ranges. Modified Min-Sum algorithm replaces the costly function evaluations with addition and shift operations. Although Modified Min-Sum has a few more additions than other algorithms, it is still preferred since nonlinear function evaluations are omitted. Also Modified Min-Sum algorithm requires less memory, hence better implemented in the hardware [134].

The direct implementation of the decoding algorithm for binary codes using the probability domain has several drawbacks as compared log-SPA, based on log-likelihood ratios (LLR). The direct implementation is more sensitive to quantization effects and requires more quantization levels than when using LLRs [135,136]. The SPA requires message multiplications, whereas the log-SPA implementation uses message additions. The latter is more efficient in fixed point implementations, as fixed point multiplications can take up many clock cycles compared to additions. Additions in the SPA are replaced by the Jacobi logarithm [137] which can be approximated by a very simple function, resulting in the max-log-SPA. Finally, the log-SPA implementation requires no normalization step [138]. A log-domain approach for non-binary turbo codes was investigated in [139]. In this contribution the log-SPA decoding scheme is derived for general non-binary LDPC codes based on LLRs. This algorithm is mathematically equivalent to the original decoding algorithm [140].

4.5 Simulation Model for Error Correction using LDPC codes

Performance of a communication system can be characterized by the BER expected from the channel. BER value is the probability of occurrence of bit-error at the receiver, and for reliable communication, the BER values should be very low. Forward Error Correction (FEC) schemes using error-control codes are widely used to reduce the BER values. Figure 4.5.1 shows a MATLAB model developed to study the BER performance of Gigabit Ethernet protocol using LDPC codes.

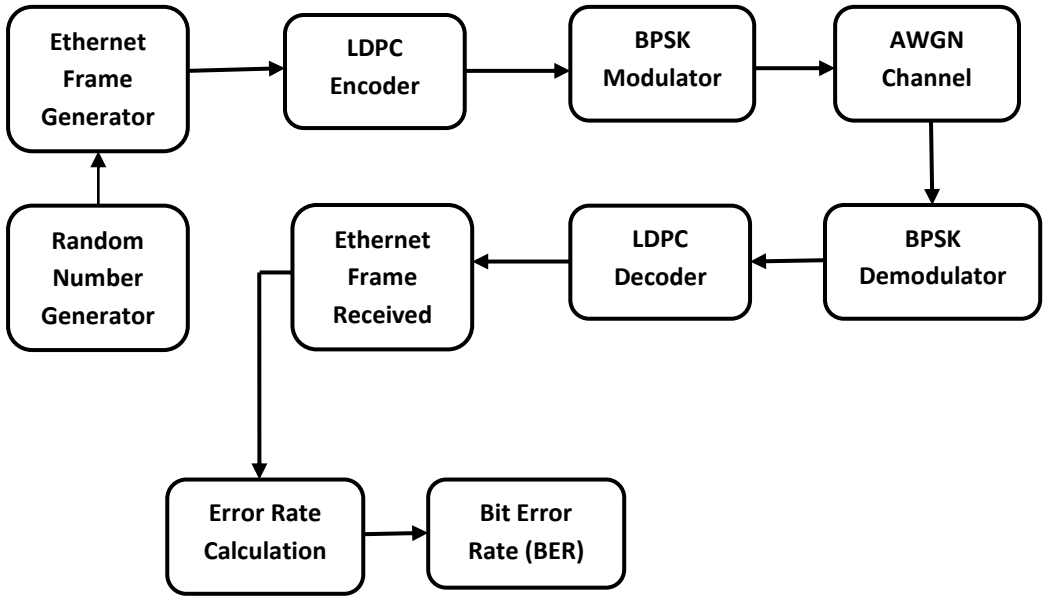


Figure 4.5.1 Matlab Simulation model for Error Correction Analysis

This model can be used for Error Correction Analysis of Gigabit Ethernet protocol. The Ethernet Frame is generated using Matlab code and is encoded using a generator matrix. The BPSK modulator maps the input binary signals, to an analog signal for transmission. The channel is the medium through which information is transmitted from the transmitter to the receiver and can be a copper, fibre optic or wireless channel. The channel is modelled as an AWGN channel. The LDPC decoder is implemented at the receiver. The decoding process uses standard functions in MATLAB for SPA-logdomain and SPA-Min-Sum algorithm, that loops through passing messages back and forth along the Tanner graph until maximum number of passes have

occurred. The BER performance is studied by measuring the BER of the received decoded message [141,142].

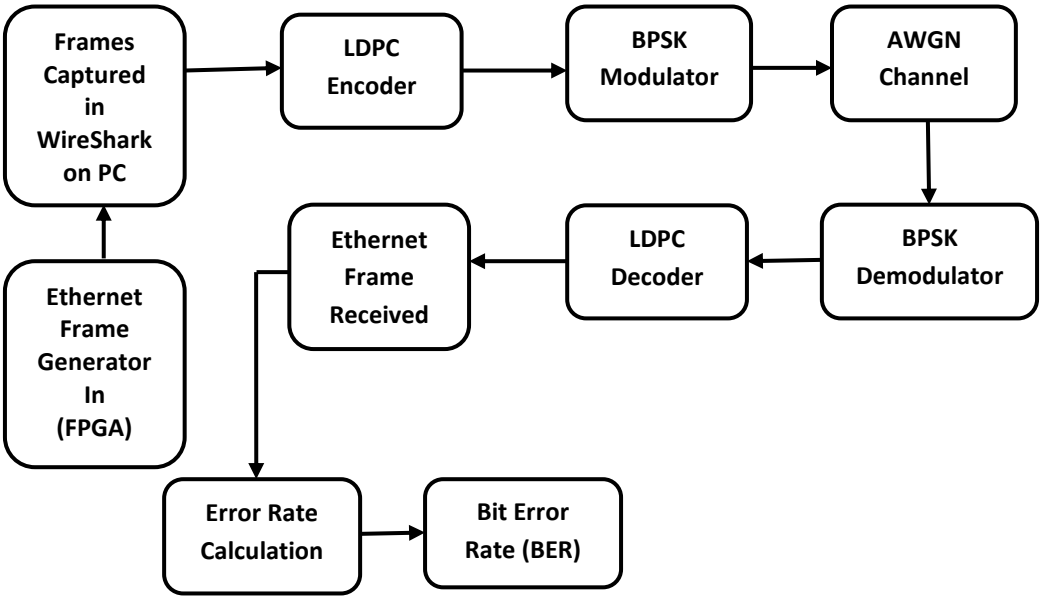


Figure 4.5.2 Matlab Simulation model for Error Correction interfaced with Gigabit Ethernet protocol design

Figure 4.5.2 illustrates the interfacing of the Matlab Simulation model with Gigabit Ethernet protocol design implemented on FPGA. Ethernet frame is generated using Altera's TSE IP core generator and these frames are captured using WireShark and given to the Matlab simulation model of error correction and the BER performance is plotted.