CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

VHDL Implementation of Reed-Solomon FEC architecture for high-speed optical communications

Master's thesis in Embedded Electronic System Design

HARINI SHANMUGAM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

VHDL Implementation of Reed-Solomon FEC architecture for high-speed optical communications

HARINI SHANMUGAM

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

HARINI SHANMUGAM

Supervisors: Christoffer Fougstedt and Per Larsson-Edefors
Examiner: Lena Peterson
Chalmers University of Technology
Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

HARINI SHANMUGAM
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In the quest to achieve high data rates, several 100 Gbps Ethernet standards for backplane, copper cables and fiber optic that include forward error correction based on Reed–Solomon (RS) codes have been recently approved. This thesis work presents the design and implementation of a high-throughput Reed-Solomon $RS(255, 239)$ decoder architecture suitable for those standards. Various error correction decoders have been formulated through algorithmic transformations of the inversionless Berlekamp Massey algorithm (IBMA). In this work, a Key Equation Solver (KES) based on the modified enhanced Parallel Inversionless Berlekamp Massey algorithm (ePIBMA) is used. Hardware implementation results are presented for the $RS(255, 239)$ codes over $GF(2^8)$ that reach 106.03 Gbps when implemented in a 65 nm CMOS process. Finally, post synthesis the timing, area and power estimates generated are also presented.

Keywords: ASIC, chien search, communications, decoder, error correction, ethernet, fiber optic, forward error correction, galois field, Gbps, hardware, high data rate, high speed, high throughput, IEEE, implementation, key equation solver, Reed-Solomon codes, syndrome calculator, thesis.

Acknowledgement

I would like to express my special thanks of appreciation to my supervisor Per Larsson-Edefors and co-supervisor Christoffer Fougstedt who gave me the opportunity to do research on this particular thesis topic. I am grateful for their patience, motivation, enthusiasm and time dedicated from start to finish. Not only was I able to widen by knowledge and technical skills during the course of this thesis to a great extent, they have also been instrumental in building my sense of confidence on the subject.
Lastly, my gratitude towards Lena Peterson for being a excellent guide and examiner.

Harini Shanmugam, Gothenburg, June 2020

Contents

List of Figures

List of Tables

List Of Abbreviations

ALC - Asynchronous Layered Coding
ASIC - Application Specific Integrated Circuit
BASE-T - Baseband Twisted Pair
CDP - Content Delivery Protocols
CMOS - Complementary Metal Oxide Semiconductor
CSEE - Chien Search and Error Evaluation
ECC - Error Correction Codes
EEE - Energy Efficient Ethernet
ePIBMA - enhanced Parallel Inversionless Berlekamp Masey Algorithm
FEC - Forward Error Correction
GbE - Gigabit Ethernet
GF - Galois Field
IEEE - Institute of Electrical and Electronics Engineers
ITU-T - International Telecommunication Union-Telecommunications section
KES - Key Equation Solver
LUT - Look Up Table
MAC - Media Access Control
MDS - Maximum Distance Separable
M2M - Machine to Machine
NACK - Negative Acknowledgement
NORM - NACK Oriented Reliable Multicast
PHY - Physical Layer
ROM - Read Only Memory
RS - Reed Solomon codes
RTP - Real Time Transport Protocol
SC - Syndrome Computer/Calculation
SNR - Signal to Noise Ratio
VHSIC - Very High Speed Integrated Circuit
VHDL - VHSIC Hardware Description Language

1
Introduction

The use of the forward error correction (FEC) codes is a classic solution to improve the reliability of unicast, multicast, and broadcast Content Delivery Protocols (CDP) and applications. FEC codes have been used with applications in public and private IP networks to provide protection against packet loss. They have been used with media delivery applications and for instance with real-time streaming media applications based on the Real-time Transport Protocol (RTP). More specifically, FEC schemes introduce error codes based on sparse parity-check matrices for object delivery protocols like Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) transport protocols. These codes are efficient in terms of processing but not optimal in terms of error recovery capabilities when dealing with "small" objects [1].

The Reed-Solomon FEC codes belong to the class of Maximum Distance Separable (MDS) cyclic error correction codes that are optimal in terms of error recovery capability. It means that a receiver can recover the K source symbols from any set of exactly K encoding symbols. These codes are also systematic codes, which means that the K source symbols are part of the encoding symbols. However, here the price to pay is a limit on the maximum source block size, on the maximum number of encoding symbols and a computational complexity higher than Low Density Parity Check codes [2] or Raptor codes [3] for instance. Since the real-time constraints of media delivery applications usually limit the maximum source block size, this is not considered to be a major issue in the context of FEC framework for many (but not necessarily all) use cases.

Reed-Solomon codes play an important role in modern multimedia communications and data storage systems. Demands for 100 Gigabit Ethernet (GbE) devices are increasing dramatically where data traffic converges, such as high-performance computing, servers, data centers and enterprise networks. In the future, bandwidth will be much more in demand than 100 GbE. For this reason, the IEEE 802.3ba task force approved IEEE std 802.3ba-2010 for the use of 40 Gb/s and 100 Gb/s Ethernet. Because the theory of Reed Solomon Code has been developed over forty years, related coding theory has been mature. Hence, it would be good time for us to make practical applications of high-speed low-complexity RS-based forward error correction (FEC) architecture implementations to meet the continuing demands for ever higher data rates (100 Gb/s and beyond) [4]. The Fig. 1.1 below shows the basic block diagram of a Reed-Solomon encoding and decoding flow over a noisy communication channel.

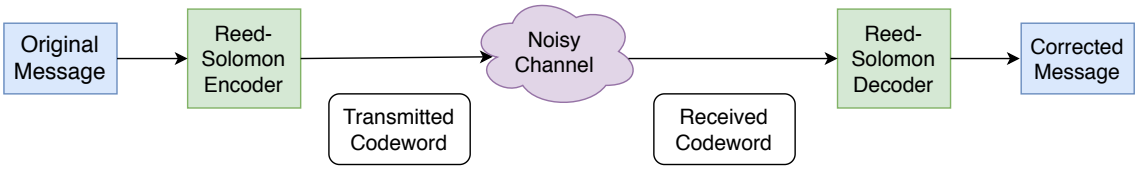

Figure 1.1: Reed-Solomon encoding/decoding over a noisy channel.

During this thesis project, a Reed-Solomon decoder architecture is designed and implemented using VHDL, i.e, VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. The developed system will be capable of reaching 100 Gbps throughput for a single data stream for the well known $RS(255, 239)$ code and synthesized using ASIC standard-cell libraries for evaluating the power consumed and area.

There are three different decoding algorithms to correct the errors introduced during transmission, namely, Berlekamp–Massey algorithm [8], extended Euclidean algorithm [9], and Welch–Berlekamp algorithm [10]. The Welch-Berlekamp algorithm is largely neglected by VLSI researcher community due to its algorithmic irregularity and high cost of on-the-fly polynomial evaluation. Initially, the extended Euclidean algorithm received popularity in VLSI community and industry practice, due to its natural regularity and short critical path delay. However, during this thesis project, the Enhanced Parallel Inversionless Berlekamp Massey algorithm is used as it is considered to be the one with the least hardware complexity [16].

1.1 Outline

The report is structured as follows: In Section II, an overview of the theoretical concepts used while designing are discussed. In Section III, the RS Decoder Architecture is discussed. In Section IV, the hardware implementation and verification of the three primary blocks of the RS decoder is explained in detail and in Section V the observations and outcome of simulations performed and synthesis results obtained are presented. Finally, in Section VI the work performed is concluded.

2

Context

In this chapter an overview of some of the theoretical concepts used while designing the Reed-Solomon decoder are presented.

2.1 The Advent of Fiber-Optic Communications

The development of telephone networks worldwide during the twentieth century led to many accomplishments in the field of electrical communication systems. Replacing wire pairs with coaxial cables increased the system capacity multifold. The first co-axial cable system deployed was a 3 MHz system capable of transmitting 300 voice channels or a single television channel. However, the bandwidth of such systems is limited by the frequency-dependent cable losses. This drawback led to the use of microwave communications systems where an electromagnetic carrier wave with frequencies in the range of 1–10 GHz was used to transmit the signal by using suitable modulation techniques.

Both coaxial and microwave systems have evolved considerably and are able to operate at bit rates of 100 Mbps. The most advanced coaxial system operated at a bit rate of 274 Mbps. A major drawback of such systems is their small repeater spacing (around 1 km), which makes the system relatively expensive to operate. A commonly used figure of merit for communication systems is the $bitrate - distance$ product, BL, where B is the bit rate and L is the repeater spacing.

It was soon realized that an increase of several orders of magnitude in the BL product could be made possible if an optical wave was used as the carrier. It was suggested in 1966 that optical fibers might be the best choice, as they are capable of guiding the light in a manner similar to the guiding of electrons in copper wires. The simultaneous availability of compact optical sources and a low-loss optical fibers in 1970s led to a worldwide effort for developing fiber-optic communication systems [5].

2.2 Ethernet IEEE 802.3 Standards

The IEEE 802.3 group is concerned with the maintenance and extension of the Ethernet data communication standards. The standards are in the series IEEE 802.3, table 2.1 [6], where each standard is identified using a different suffix letter, thus the different IEEE 802.3 standards can be uniquely identified. The different IEEE 802.3 standards define different aspects of Ethernet. Some standards may introduce new versions of Ethernet to keep pace with the growing requirements for

speed and performance, whereas other standards may define aspects like the data frames used.

Table 2.1: Ethernet IEEE 802.3 Standards, Supplements and Releases.

STANDARD SUPPLEMENT	YEAR	DESCRIPTION
802.3a	1985	10Base-2 (thin Ethernet)
802.3c	1986	10 Mb/s repeater specifications
802.3d	1987	FOIRL (fiber link)
802.3i	1990	10Base-T (twisted pair)
. . .	. . .	. . .
802.3bs	2017	200GbE (200 Gbit/s) over single-mode fiber and 400GbE (400 Gbit/s) over optical physical media
802.3bt	2018	Third generation Power over Ethernet with up to 100 W using all 4 pairs balanced twisted-pair cabling (4PPoE).
802.3bu	2016	Power over Data Lines (PoDL) for single twisted-pair Ethernet (100BASE-T1)
802.3bv	2017	Gigabit Ethernet over plastic optical fiber (POF)
802.3by	2016	Optical fiber, twinax and backplane 25 Gigabit Ethernet
802.3bz	2016	2.5GBASE-T and 5GBASE-T – 2.5 Gigabit and 5 Gigabit Ethernet over Cat-5/Cat-6 twisted pair
802.3cb	2018	2.5 Gbit/s and 5 Gbit/s Operation over Backplane
802.3cc	2017	25 Gbit/s over Single Mode Fiber
802.3cd	2018	Media Access Control Parameters for 50 Gbit/s and Physical Layers and Management Parameters for 50, 100, and 200 Gbit/s Operation
802.3ce	2017	Multilane Timestamping
802.3cf	2019	YANG Data Model Definitions

Ethernet, IEEE 802.3 defines the frame formats or frame structures that are developed within the media access control (MAC) layer of the protocol stack. Essentially the same frame structure is used for the different variants of Ethernet, although there are some changes to the frame structure to extend the performance of the system should this be needed. With the high speeds and variety of media used, this basic format sometimes needs to be adapted to meet the individual requirements of the transmission system, but this is still specified within the amendment/update for that given Ethernet variant.

The MAC layer mentioned above and the logical link control (LLC) sublayer together make up the data link layer, which corresponds to the second layer of the OSI model. While the LLC is responsible for providing flow control and multiplexing for the logical link, the MAC provides flow control and multiplexing of the transmission medium.

On June 17, 2010, the IEEE 802.3ba standard was approved. In March 2011

the IEEE 802.3bg standard was approved. On September 10, 2011, the P802.3bj 100 Gbit/s Backplane and Copper Cable task force was approved. The scope of this project is to specify additions to and appropriate modifications of IEEE Std 802.3 to add 100 Gbit/s 4-lane Physical Layer (PHY) specifications and management parameters for operation on backplanes and twinaxial copper cables, and specify optional Energy Efficient Ethernet (EEE) for 40 Gbit/s and 100 Gbit/s operation over backplanes and copper cables [7].

On May 10, 2013, the P802.3bm 40 Gbit/s and 100 Gbit/s Fiber Optic Task Force was approved. This project is to specify additions to and appropriate modifications of IEEE Std 802.3 to add 100 Gbit/s Physical Layer (PHY) specifications and management parameters, using a four-lane electrical interface for operation on multimode and single-mode fiber optic cables, and to specify optional Energy Efficient Ethernet (EEE) for 40 Gbit/s and 100 Gbit/s operation over fiber optic cables. In addition, this project is to also specify 40 Gbit/s Physical Layer (PHY) specifications and management parameters for operation on extended reach (>10 km) single-mode fiber optic cables [7].

2.3 Multimedia Communication System

The basic building blocks of Internet Protocol based Data/Image/Video communication system or Multimedia communication system (M2M) can be seen in Fig. 2.1. In the following sections, the function of the "Channel Decoder" block will be described in more detail.

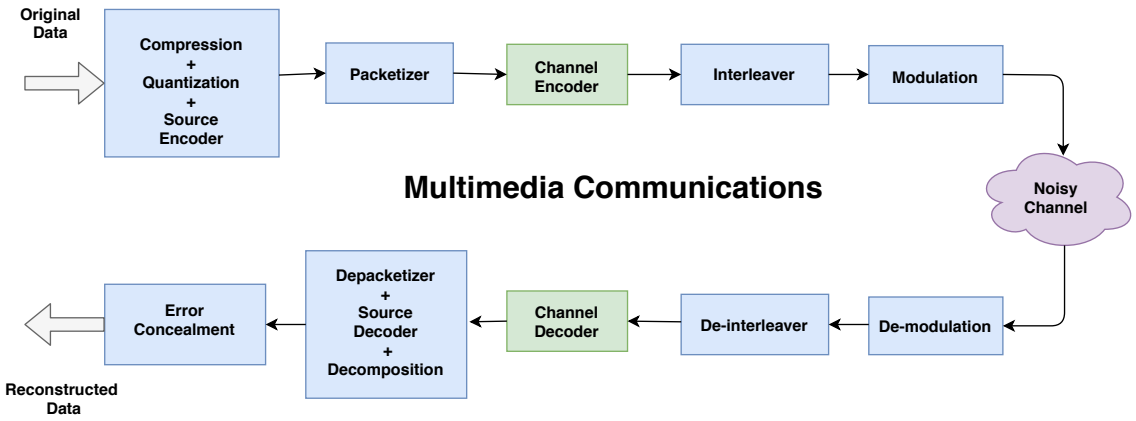

Figure 2.1: Block diagram of a end-to-end data communications over a noisy channel.

2.4 Error Control Code

Error control codes (ECC) are commonly used in telecommunication, computing and coding theory for controlling errors in data that are sent over unreliable or noisy communication channels [12]. The sender encodes the message with redundant or parity data in the form of an ECC. The redundancy allows the receiver to detect

a limited number of errors that may occur anywhere in the message and to correct these errors without retransmission. ECC gives the receiver the ability to correct errors without needing a reverse channel to request retransmission of data, but at the cost of a fixed, higher forward-channel bandwidth. The two main categories of ECC codes are block codes and convolutional codes.

In block codes [13], the decoder looks for errors and once detected, corrects them (according to the capability of the code). The technique has become an important signal-processing tool used in modern communication systems and in a wide variety of other digital applications such as high-density memory and recording media. Such coding provides system performance improvements at significantly lower cost than through the use of other methods that increase signal-to-noise ratio (SNR) such as increased power or antenna gain.

In convolutional coding [14], a continuous source of bits is used to generate continuous output stream. Then from each codeword, a column of the most significant bits is transmitted in blocks and then the next most significant bits and so on until the least significant bits are transmitted in a column. The reverse operation is performed by the controlling device at the receiver part with error checking and correction if required. The main effect of this approach is that if an error burst occurs, it is going to affect a single bit of each code word rather than affecting a string of bits in one or two code words.

Block codes depend only on the current message bit k while convolution coding is dependent on the current message bit as well as the previous sequence of source bits. Hence, it could be said that block codes were used in memory less systems but convolutional coding is dependent on memory. The convolutional coding is best suited for applications having low implementation cost with good performance.

There are many types of block codes, but among the classical ones the most notable is Reed-Solomon coding because of its widespread use in compact discs, DVDs, and hard disk drives. Other examples of classical block codes include Golay, Multidimensional parity, BCH (Bose–Chaudhuri–Hocquenghem) and Hamming codes [13].

2.5 Reed-Solomon Codes

A Reed-Solomon code is a block code and specified as RS(N, K) as shown in Fig. 2.2. The variable N is the size of the codeword. K is the number of the data symbols

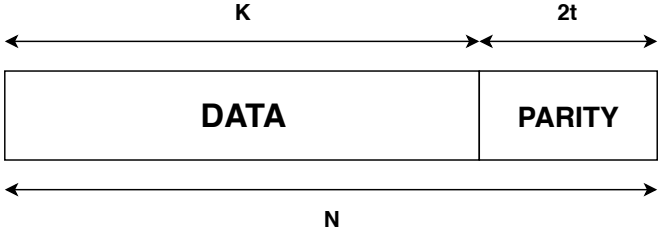

Figure 2.2: Structure of RS Code.

and $2t$ is the number of parity symbols, where each symbol contains s number of bits. The relationship between the symbol size s and the size of codeword N is given by (2.1). This means that if there are s bits in one symbol, then the number of distinct symbols in one codeword, excluding the one with all zeros, is given by:

$$N = 2^{s-1} \tag{2.1}$$

This code allows correcting up to t symbol errors where t is given by,

$$t = \frac{N - K}{2} \tag{2.2}$$

2.5.1 Galois Field (GF)

Reed-Solomon codes are based on a specialist area of mathematics known as Galois fields or finite fields [15]. A finite field has the property that arithmetic operations (+,-,x,/ etc.) on field elements always have a result in the field. A Reed-Solomon encoder or decoder needs to carry out these arithmetic operations.

2.5.2 RS decoder

The Reed-Solomon decoder tries to correct errors by calculating the syndromes for each codeword. Based upon the syndromes, the decoder is able to determine the number of errors in the received block. If there are errors present, the decoder tries to find the locations of the errors by creating an error locator polynomial. The roots of this polynomial are found using the Chien search algorithm and using Forney's algorithm, the symbol error values are found and corrected [11]. General process of decoding is,

1. Syndrome Calculation (SC unit)
2. Determine error-location polynomial (KES unit)
3. Solving the error locator polynomial - Chien search (CSEE unit)
4. Calculating the error magnitude
5. Error correction

Consider an $RS(N, K)$ code defined in Galois Field, i.e, GF(2^m), where $N = 2^m - 1$. In such a case, $2t$ redundant symbols are added to the original K-symbol codeword $C(x)$. This codeword is created as $C(x) = M(x) \cdot G(x)$, where $M(x)$ is the message polynomial and $G(x)$ is the generator polynomial of the code. After the codeword is transmitted through a noisy channel, the RS decoder receives $R(x) = C(x) + E(X)$, where $E(x)$ is the polynomial that represents the noise.

Syndrome Computation

The RS decoding process begins with the syndrome calculator unit. This unit computes the $2t$ syndromes S_i that are the coefficients of the syndrome polynomial $S(x)$. This is achieved by evaluating the received polynomial in the $2t$ roots of $G(x)$, specifically $S_i = R(\alpha^{i+1})$ for $i = 0, .., 2t - 1$, where α is the primitive element of GF(2^m).

Error Locator Polynomial

The KES unit obtains the error locator polynomial $\wedge(x)$ and the error magnitude $\Omega(x)$ polynomials by solving the key equation $\wedge(x) \cdot S(x) = \Omega(x) \bmod x^{N-K}$. The KES block is the most complex one and it is usually implemented with either the Modified Euclidean (ME) algorithm or the Berlekamp-Massey (BM) algorithm [16].

Error Evaluator Polynomial

Next, the Chien search is performed to find the error locations, which is accomplished by evaluating $\wedge(x)$ in all the possible positions (*i.e.* $\wedge\left(\alpha^{-n}\right)$, for $n = 1, .., N-1$).

Error Magnitude Polynomial

Finally, an error evaluation method (*e.g.* Forney's formula) is used to calculate the error magnitude ($e.g. E_n = \Omega(\alpha^{-n}) / \wedge'\left(\alpha^{-n}\right)$) when the Chien search finds an error location, which is whenever $\wedge(\alpha^{-n}) = 0$.

Error Correction

If the total number of errors in R(x) does not exceed the error correcting capability t, all the errors in $R(x)$ are corrected by subtracting the error magnitudes from the received symbols. If all $2t$ syndromes S_i are zero, then no error has occured [4], [11].

3

RS Decoder Architecture

The hard-decision RS decoder architecture consists commonly of three main computational blocks, which need to be implemented. The first block is the syndrome computer (more details are given in section 4.1). This component obtains a set of syndromes that is a function of the error pattern in the received frame. Upon computation of the $2t$ (t denotes the error correcting capability of the decoder) syndromes by the syndrome computer (SC) unit, the values are used in the second block of the decoder called the key equation solver (more details are given in section 4.2), to compute the error-locator polynomial.

The error-locator polynomial will be generated in the key equation solver (KES) unit using the enhanced parallel inversionless Berlekamp-Massey algorithm (ePIBMA) that effectively takes advantage of the generalized Horiguchi-Koetter formula. Here the ePIBMA algorithm was choosen as it requires only $2t + 1$ systolic cells, in contrast to the $3t$ or more systolic cells required for regular architectures based on inversionless Berlekamp-Massey algorithm (IBMA) or the Euclidean algorithm [16].

The last block, chien search and error evaluation (CSEE) (more details are given in section 4.3) determines the roots of the error-locator polynomial. Finally, the error magnitudes are found, typically using Forney's algorithm (see Lemma 4 from [16]). The output of the CSEE block is subtracted from the received sequence to obtain the corrected frame [11].

The four decoding stages are commonly pipelined as shown in Fig. 3.1 and in these units the full polynomial-base GF(2^m) is required and the corresponding GF multipliers and adders need to be implemented.

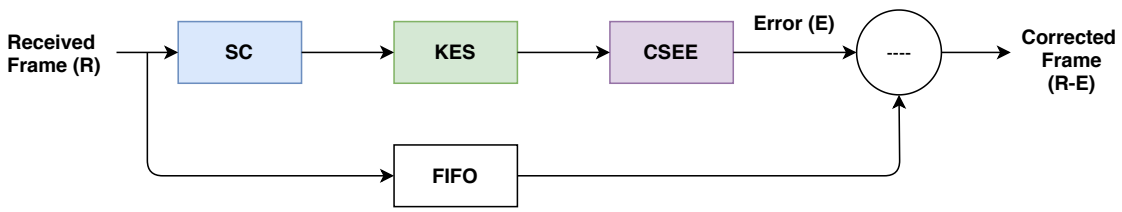

Figure 3.1: Basic units of a conventional $RS(N, K)$ decoder.

3.1 Conventional RS Decoders

Conventionally an architecture consisting of ch parallel SC and CSEE blocks that compute each one P symbols at a time and one KES stage that is shared among ch channels is seen in literature [17]. If this architecture is replicated q times, $q \cdot ch$ channels are decoded at the same throughput per channel, see Fig. 3.2 [11]. Thus the throughput of the complete decoder is given as,

$$\text{Throughput} - \text{of} - \text{decoder} = \frac{q \cdot ch \cdot N \cdot m \cdot f_{clk}}{[N/P]} \text{bps} \tag{3.1}$$

And the throughput per channel of the decoder is given as,

$$\text{Throughput} - \text{per} - \text{channel} = \frac{N \cdot m \cdot f_{clk}}{[N/P]} \text{bps} \tag{3.2}$$

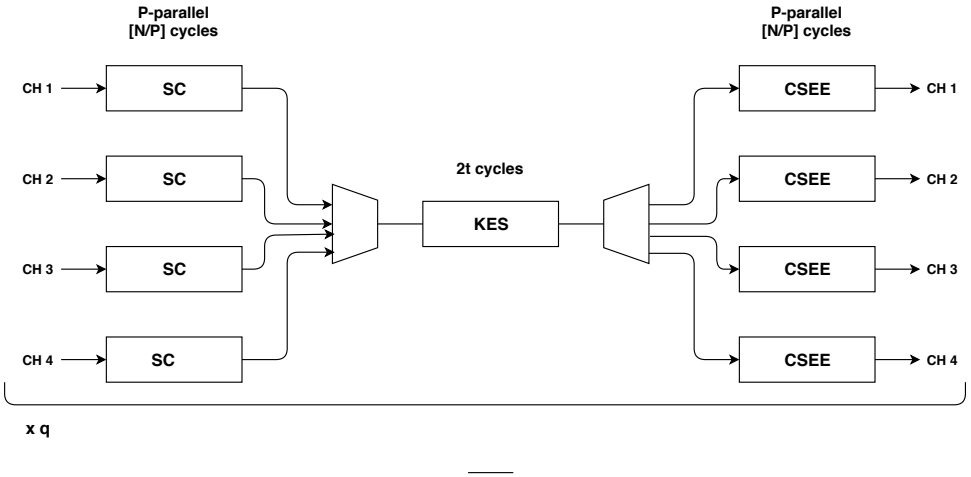

Figure 3.2: Conventional parallel RS architecture.

3.2 High-Speed RS Decoders

Considering a single channel high-speed low-latency RS decoder, the SC and CSEE blocks both take N cycles whereas the systolic KES block takes only $2t$ cycles, which is typically much smaller than N. A common practice is to fold the KES circuit (by $[N/2t]$ times, if systolic) to reduce the circuit complexity while the latencies of the three pipeline stages are equalized in order to achieve faster data rates, Fig. 3.3. In general, h systolic KES processors given as $h = [2t/[N/P]]$ are needed in parallel to keep up with the throughput of the SC and CSEE blocks. Moreover, it should be noted that the closer N/P is to $2t/h$ the higher the hardware utilization efficiency is [11].

In case of a $RS(255, 239)$, the maximum achievable efficiency for $h = 1, 2, 3, 4...$ is reached when P value is set to $P = 16, 32, 64, ...$ Assuming that the decoder can

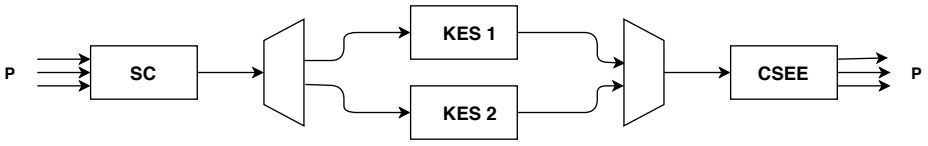

Figure 3.3: High-speed RS architecture.

work when the f_{clk} rate is set to 500 MHz in the target CMOS technology, a throughput rate greater than 100 Gbps can be achieved when two systolic KES blocks are disposed in parallel. The throughput of the decoder is given as,

$$\text{Throughput} - \text{of} - \text{decoder} = \frac{N \cdot m \cdot f_{clk}}{[N/P]}\text{bps} \tag{3.3}$$

The computational timing chart for the high-speed decoder that processes a single channel with two KES blocks disposed in parallel is given in Fig. 3.4 below. In case of $RS(255, 239)$, each computational block processes 32 input symbols per clock cycle and thus the $N = 255$ symbol codeword is evaluated in $N_c = [N/P] = 8$ clock cycles and $2t$ equates to 16 clocks.

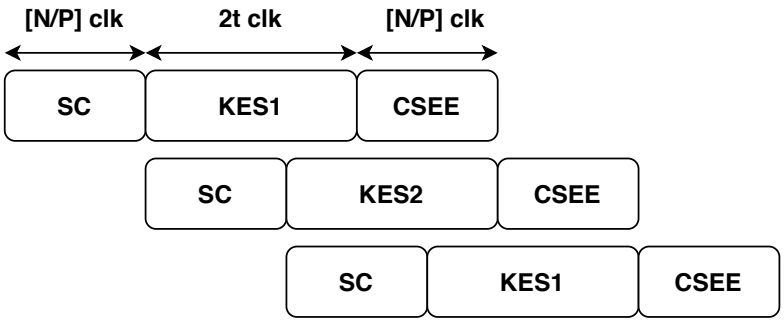

Figure 3.4: High-speed RS Decoder timing chart with h=2 KES blocks.

4

Implementation

In this chapter the hardware implementation and the verification of the three main basic blocks of the $RS(255, 239)$ decoder will be discussed in detail.

The Reed-Solomon $RS(255, 239)$ code was chosen for implementation as it is a well-known code for high-speed optical communications [17] and it is recommended by the ITU-T for optical fiber submarine cable systems [19].

4.1 Syndrome Computation

The syndrome computer (SC) unit processes the $2t = 16$ syndromes using $2t$ computational blocks disposed in parallel. The architecture of the syndrome calculator is derived from the Horner's rule and is similar to the syndrome computational unit implemented in [11].

Before the start of the computation, the received polynomial codeword (R) is padded with zeros such that its length is $N_c \cdot P$ symbols. Next, the computation is performed iteratively during the $N_c = 8$ clock cycles using equations

$$syn_i^{flag+1} = syn_i^{flag} \cdot \alpha^{P \cdot i} + \sum_{p=1}^{P} R_{(P \cdot (N_c - 1) - P \cdot (flag) + p)} \cdot \alpha^{(p-1) \cdot i} \tag{4.1}$$

$$SyndromeOut_i = syn_i^{N_c} \tag{4.2}$$

where R is the coefficient of the received polynomial, $flag \in (0, 1, 2, .., N_c - 1)$ and $i \in (1, 2, .., 2t)$.

4.1.1 Function of $RS(255, 239)$ syndrome computer

1. The received polynomial, $N = 255$ codeword is padded with zeros when the reset signal is set to standard logic '1', such that the codeword size is extended to 256 coefficients, which are 8 bits each.
2. Next, at every clock cycle beginning from $flag = 0$ to $flag = 7$, $P = 32$ coefficients are stored in the array R which are then sent to the $2t = 16$ computational blocks disposed in parallel, see Fig. 4.1.

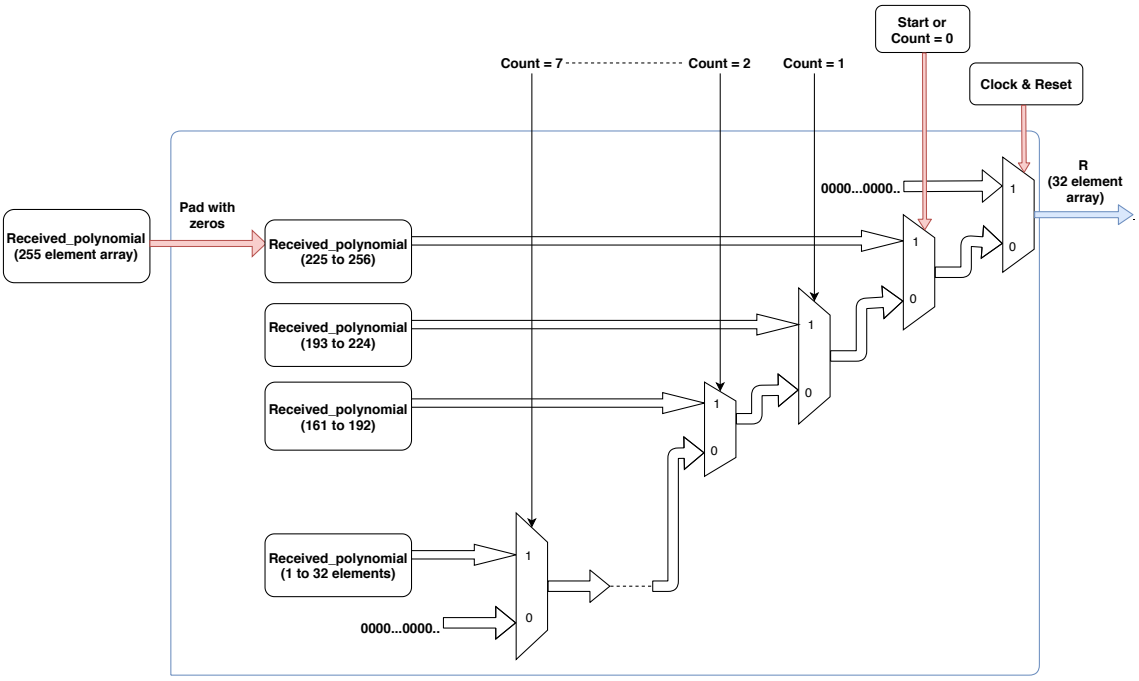

Figure 4.1: Design structure for processing the P=32 symbols per clock cycle.

3. At each computational block the incoming P coefficients are multiplied with their respective α values as given in (4.8). The $2t$ products are then summed together using the GF_{add} summation unit.
4. The result from the summation unit is then multiplied with α^{P*i} and added back to the summation unit result during the next clock cycle. Note that i is the i^{th} computational block unit, see Fig. 4.2.

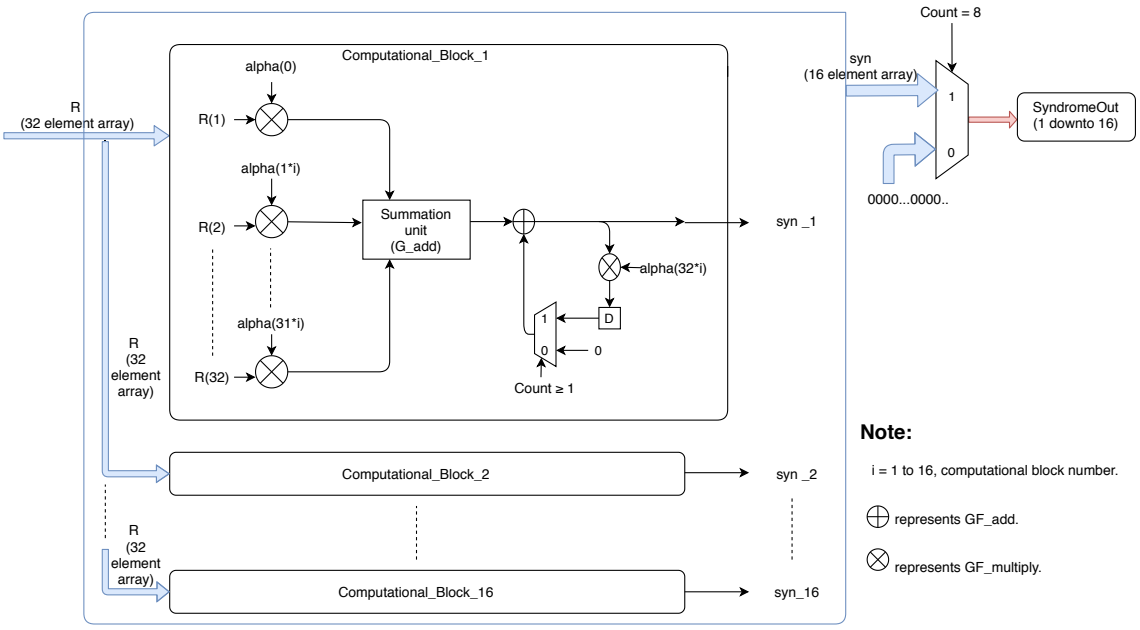

Figure 4.2: Design structure of $2t$ parallel computational blocks.

5. The above steps are repeated at each clock cycle and at the last iteration $flag = 7$ the final $2t$ syndromes are obtained as per (4.2) and the syndromes are obtained at the end of the 9^{th} clock cycle.

4.1.2 Verification of the syndrome computer

In order to verify the result of the SC unit developed using VHDL in ModelSim, it was essential to first develop a reference system in Matlab.

Considering the $RS(255, 239)$ code over $GF(2^8)$, the codeword $C(x)$ is first created. Here the coefficients of $C(x)$ are inserted as elements into a matrix of size $[1, 239]$. $C(x)$ is then passed through a reference Matlab encoder code. Next, errors of varying magnitudes are introduced into the encoded message at this stage while testing the SC unit.

The received message of size $[1, 255]$, is then passed through the reference SC unit developed in Matlab, which is based on Horner's rule given by the following equation.

$$R(\alpha) = r_0 + \alpha(r_1 + \alpha(r_2 + \alpha(r_3 + ... + \alpha(r_{n-1} + \alpha \cdot r_n)....))) \tag{4.3}$$

where R is the received polynomial codeword; $r_0, r_1, r_2, ..., r_{n-1}$ are the coefficients of the received polynomial codeword and α is the primitive polynomial. Finally, the $2t$ syndromes are calculated using equation:

$$S_i = R(\alpha^{i+1}) \tag{4.4}$$

where $i \in (1, 2, ..., 2t)$. A block diagram of the above described process can be seen in Fig. 4.3. Note, the evaluated $2t = 16$ syndromes will then have to be converted into binary vector format i.e as $t = 8$ bit symbols before they are compared with the result obtained while simulating the SC design described using VHDL. Refer to section 5.1.1 to read more on the logic simulations.

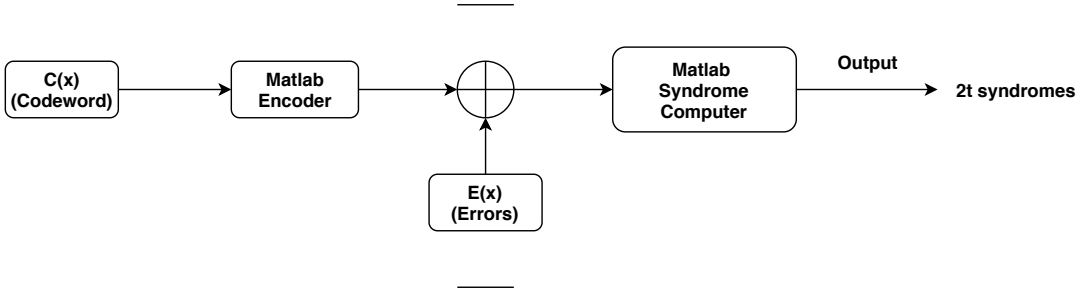

Figure 4.3: Work flow of syndrome computation in Matlab.

4.2 Key Equation Solver

The KES unit is based on the modified Enhanced Parallel Inversionless Berlekamp-Massey algorithm (ePIBMA) [18]. This algorithm is being used as it has the shortest critical path and utilizes the lowest number of resources. The $2t$ syndromes computed earlier are passed as inputs to this algorithm and its outputs are the error-locator $\wedge(x)$ polynomial, the error-evaluator $B(x)$ polynomial, γ value, z_{cnt} value, which is used to compute the magnitude of the error and $L_\wedge$, the length of the error-locator polynomial that is used to check the validity of the correction.

In order to limit the degree of the $B(x)$ polynomial to $t-1$, the author of [16] introduced an auxiliary criterion and utilizes a separate loop logic z_{cnt} [11] [18], to accumulate the unknown $\alpha^{-(t+e-2)}$ (e being the number of errors), which will be used in the error evaluation. The enhanced algorithm used to design the KES unit can be found in [11] [16] [18]. Fig. 4.4 shows the systolic ePIBMA block diagram.

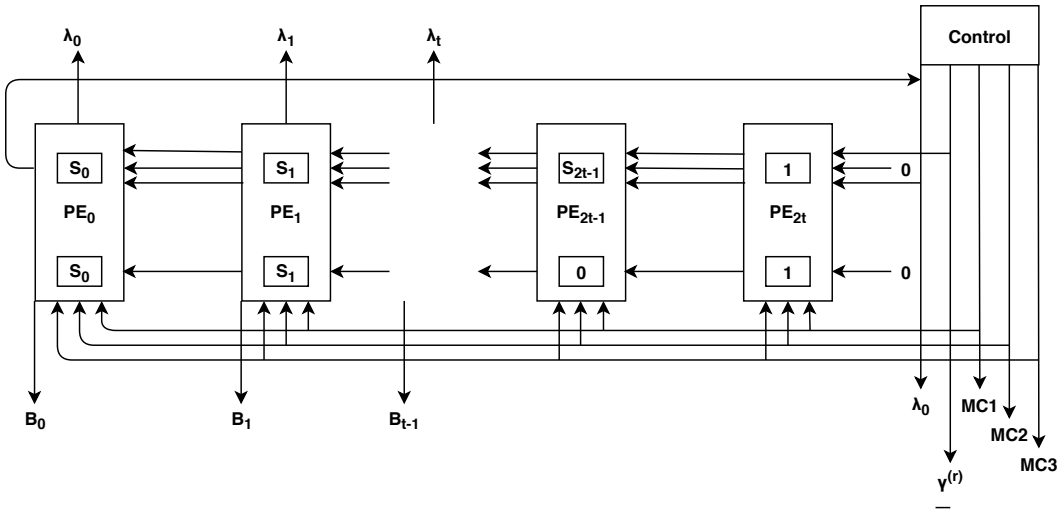

Figure 4.4: The systolic ePIBMA block diagram [16].

The ePIBMA architecture contains an array of $2t+1$ homogeneous systolic processor elements (PE), that stores the $\hat{\Omega}(x)$ and $\hat{\Theta}(x)$ array values. The control signal MC1 represents the boolean operation $\Omega^{(r)} \neq 0$ and $L_\wedge^r \leq L_B^r$. In each iteration, the element $\hat{\Omega}_i$ is set to zero if its index i corresponds to the value denoted by the control signal MC2. Finally, MC3 represents the boolean operation $L_B < t-1$ evaluated in the algorithm.

4.2.1 Function of $RS(255, 239)$ KES unit

1. Initialization of the following array and variable elements during the 1^{st} iteration:
   (a) $\hat{\Omega}(x) = S_0 + S_1(x) + ... + S_{14}(x) + S_{15}(x) + x^{16}$
   (b) $\hat{\Theta}(x) = S_0 + S_1(x) + ... + S_{14}(x) + 0 + x^{16}$
   (c) $\gamma = 1$
   (d) $L_\wedge = L = 0$

(e) $z_{cnt} = 0$

2. From the 2^{nd} to 17^{th} iteration the $\hat{\Omega}(x)$ array values need to be updated as described in the ePIBMA algorithm. Also, the $\hat{\Theta}(x)$ array values, $L_{\wedge}$, L, γ and z_{cnt} values need to be updated when the conditions described by the control signals MC1, MC2 and MC3 are met.
3. Finally, during the 18^{th} iteration the values obtained at the end of the previous iteration are displayed and passed to the CSEE unit. The list of values passed to the CSEE unit are as follows:
   (a) $\wedge = [\Omega_0, \Omega_1, ..., \Omega_8]$
   (b) $B = [\Theta_0, \Theta_1, ..., \Theta_7]$
   (c) γ, $L_{\wedge}$, z_{cnt}.

4.2.2 Verification of the Key Equation Solver

In order to verify the result of the KES unit developed using VHDL in ModelSim, similar to the SC unit, it was essential to first develop a reference system in Matlab.

The ePIBMA algorithm presented in [16] was first programmed in Matlab following the same design steps as described above. The result obtained was then compared to the result obtained while simulating the KES unit described using VHDL in ModelSim.

4.3 Chien Search and Error Evaluation

In order to maintain the throughput reached by the previous stages, the CSEE unit has to process $P = 32$ symbols at a time. Therefore, the total number of clock cycles to process the $N = 255$ codeword is $N_c = [N/P] = 8$. The CSEE unit designed in VHDL using ModelSim evaluates the error-locator $\wedge(x)$ polynomial for all its possible roots and calculates the error-magnitude $Y(x)$ polynomial in a total of 9 clock cycles. The CSEE architecture designed is given in Fig. 4.5 [11].

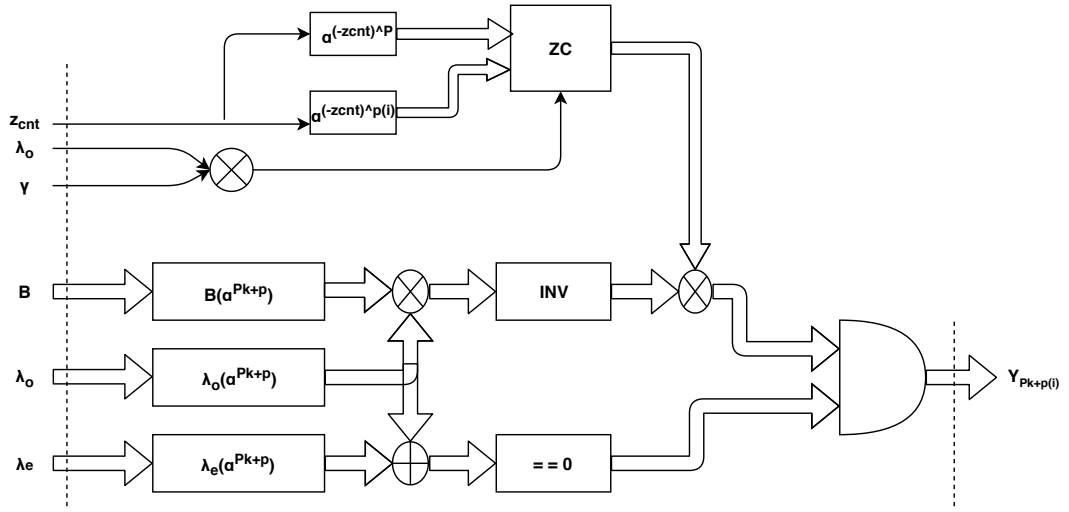

Figure 4.5: The architecture of the CSEE unit [11].

The evaluation of the $\wedge(x)$ polynomial is performed as $\wedge(x) = \wedge_o(x) + \wedge_e(x)$, where $\wedge_o(x)$ and $\wedge_e(x)$ are the polynomials containing the odd and even coefficients of $\wedge(x)$. Next the B_x, $\wedge_o(x)$ and $\wedge_e(x)$ polynomials are evaluated for $P = 32$ different values $p \in (0, 1, ..., P-1)$ of x at a time, using the below equations.

$$B(\alpha^{Pk+p}) = \sum_{j=0}^{7} B_j \cdot \alpha^{Pk+p} \tag{4.5}$$

$$\wedge_o(\alpha^{Pk+p}) = \sum_{j=0}^{3} \wedge_{o(2j+1)} \cdot \alpha^{Pk+p} \tag{4.6}$$

$$\wedge_e(\alpha^{Pk+p}) = \sum_{j=0}^{4} \wedge_{e2j} \cdot \alpha^{Pk+p} \tag{4.7}$$

Here, $k \in (0, 1, ..., N_c - 1)$ thus all the possible roots are evaluated in $N_c = 8$ clock cycles. Since, we use the ePIBMA, the error magnitude is calculated with the below equation.

$$Y_i = \frac{\gamma \cdot \lambda_0 \cdot \alpha^{(-z_{cnt})^{p(i)}}}{B(X_i^{-1}) \cdot \wedge'(X_i^{-1})} \tag{4.8}$$

Here, $i \in (1, 2, ..., 255)$, λ_0 is the coefficient 0 of $\wedge(x)$. The term $\wedge^{'}(x)$ is evaluated as $\wedge^{'}(x) = x^{-1} \cdot \wedge_o(x)$.

4.3.1 Computation of P Consecutive Powers of z_{cnt} using ROM

The computations can be performed with lower latency using ROMs addressed by z_{cnt}, as depicted in Fig. 4.6. Since there are only $2t = 16$ possible values for z_{cnt}, the size of the $P - 1$ LUTs required to initialize the $\alpha^{(-z_{cnt})^{p(i)}}$ computations is $2t$ words each.

Thus, the contents of the i^{th} LUT will be $\alpha^{(-z_{cnt})^{p(i)}}$ where $z_{cnt} \in (1, 2, ..., 16)$ and $p(i) \in (0, 1, ...31)$. Each clock cycle the evaluated error locations are shifted P positions and the values of $\alpha^{(-z_{cnt})^{p(i)}}$ are updated according to the i^{th} iteration.

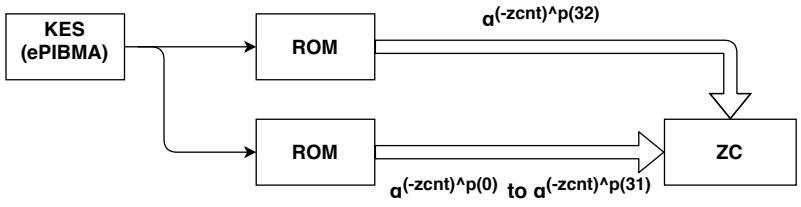

Figure 4.6: Generation of the required powers of z using ROMs.

5

Results

This chapter will explain more in detail the outcome of simulations performed in ModelSim and synthesis results obtained from Cadence RTL Compiler.

5.1 Verifying the system logic function

To test the $RS(255, 239)$ decoder system designed, the necessary $N = 255$ coefficient received polynomial test sequence was first developed using Matlab. Next, a testbench was developed in ModelSim which provides the received polynomial test sequence as input to the entire system. Before placing the entire system comprising of the SC, KES and CSEE units together, each of the units were individually verified and tested. This section will explain the outcome of these logic simulations.

5.1.1 Syndrome Computation Unit

The primary inputs to this unit were the clock, the reset, an enable signal named start and the $N = 255$ coefficient received polynomial test sequence supplied by the testbench. The α values required for the evaluation of the computational blocks were generated using Matlab. Note, the hardware structure of the computational blocks is explained in section 4.1.

To test the correctness of the entire system, various 8 bit errors at different positions were introduced into the received polynomial test sequence at this stage of testing. It was made sure that the total number of errors introduced did not exceed the error correcting capability t of the $RS(255, 239)$ decoder.

The evaluated syndromes obtained as output while simulating were then verified with the result obtained from the reference SC model developed in Matlab.

5.1.2 Key Equation Solver

The primary inputs to this unit were the clock, the reset and the verified syndromes passed from the SC unit. The 9 coefficient error-locator polynomials, the 8 coefficient error-evaluator polynomials, along with the γ value and the z_{cnt} value obtained as simulation outputs from the KES unit was verified with the reference KES model results developed in Matlab. The verified outputs were then passed down to the CSEE unit.

5.1.3 Chien Search and Error Evaluation

The primary inputs to this unit were the clock, the reset, the error-locator polynomial, the error-evaluator polynomial, the γ value and the z_{cnt} value. While testing the system for its correctness, the output observed during simulations was verified with the reference CSEE model results developed in Matlab. The CSEE unit was observed to evaluate the error magnitude at the respective error locations accurately. The obtained error magnitude values were then added (addition over Galois Field) to the received polynomial to obtain the corrected RS codeword.

5.1.4 Full System

The developed individual units were then connected together to form the entire system and was observed to function correctly, provided the total number of errors introduced into the test sequence did not exceed the error correcting capability. When errors greater than the error capacity were introduced, the sequence of coefficients received from the communication network were uncorrected by the decoder, i.e, the exact sequence (received polynomial) was observed at the output of the $RS(255, 239)$ decoder.

5.2 Synthesis and Timing Analysis

In order to be certain that the system designed was synthesizable using the 65 nm CMOS process technology, the code was first elaborated and observed that zero error messages were displayed. During the elaboration phase, the RTL Compiler makes use of a virtual gate library to which no specific process technology is associated. The VHDL descriptions were then synthesized by mapping them to a 65 nm process technology library of standard cells supplied by ST Microelectronics.

5.2.1 Timing and Area Report of Full System

Generally, the optimization goal during synthesis is to create the smallest possible implementation of the design that satisfies a certain timing constraint. But in order to get an indication of the intrinsic implementation timing, the synthesis process was started with no timing constraints set and using low computational effort.

After the initial exploratory synthesis, new timing goals were set and using medium computational effort the synthesis process was carried out. The obtained numbers on area and timing slack for the various timing goals set are given in Table 5.1.

Table 5.1: Timing and Area Report of Full System developed.

Timing goal	**Effort**	**Total Area**[μm^2]	**Timing Slack**[ps]
Unconstrained	Low effort	1 098 452	7888, Final arrival time
1000 ps	Medium effort	1 081 431	−1265, Negative slack
2240 ps	Medium effort	963 661	−14, Negative slack
2251 ps	Medium effort	944 355	−3
2253 ps	Medium effort	951 209	−13
2254 ps	Medium effort	957 957	0
2270 ps	Medium effort	927 842	0
2390 ps	Medium effort	863 476	0
3000 ps	Medium effort	754 516	0

The area estimate obtained for unconstrained timing goal was observed to be higher as no area optimizations is being done by the synthesis tool and significantly less number of buffers (0.7% of total area), higher number of sequential, inverter and logic instances were mapped during synthesis, compared to the other timing goals set.

Here, various timing goals were chosen to observe the highest throughput that can be achieved by the developed system i.e. the lowest timing constraint met before observing negative slack, as negative slack indicates failure to achieve the desired timing constraint. From the above Table 5.1 it is clear that 2254ps is the lowest timing constraint met or the worst-case path delay value.

5.2.2 Throughput Calculation

For a $RS(N, K)$ code, where K is the number of information symbols encoded and sent through the noisy communication channel, the throughput of a decoder is calculated to be:

$$\text{Throughput} = \frac{\text{Ksymbols}}{\text{TimeConstraint}} \tag{5.1}$$

Thus, for each of the timing constraints met by the designed and implemented $RS(255, 239)$ decoder, considering only the $K = 239$ original information symbols and discarding the $2t = 16$ redundant symbols, the throughput is calculated to be:

$$\text{TimingGoal} : 2254\,\text{ps} => \text{Throughput} = \frac{239}{2254 \cdot 10^{-12}} = 106.03\,\text{Gbps} \tag{5.2}$$

$$\text{TimingGoal} : 2270\,\text{ps} => \text{Throughput} = \frac{239}{2270 \cdot 10^{-12}} = 105.28\,\text{Gbps} \tag{5.3}$$

$$\text{TimingGoal} : 2390\,\text{ps} => \text{Throughput} = \frac{239}{2390 \cdot 10^{-12}} = 100\,\text{Gbps} \tag{5.4}$$

Here, 106.03 Gbps is observed to be the highest data throughput achieved by the designed and verified decoder, as negative slack which indicates the failure to met the targeted speed is observed for timing constraints below 2254 ps, Table 5.1.

5.2.3 Area Report of individual units

The estimates of area utilized by the individual SC, KES and CSEE units when timing goals corresponding to 100 Gbps or higher were set during synthesis are shown in Table 5.2.

Table 5.2: Estimated area utilized by individual units for timing goals corresponding to 100 Gbps and above.

	2254[ps]	**2270**[ps]	**2390**[ps]
Type	**Area**[μm^2]	**Area**[μm^2]	**Area**[μm^2]
SC unit	110 641	110 667	110 632
KES unit	24 560.6	24 560.6	24 577.8
CSEE unit	749 986	740 366	678 619.2

Here, it should be noted that 2 KES units were disposed in parallel as part of the architecture, in order to achieve the higher throughput. Along with the 3 primary blocks, additional area is utilized by the subtractor block, which is needed to subtract the output of the CSEE block from the received sequence to obtain the corrected result.

From Tables 5.1 and 5.2, it is evident that the sum of the area estimates of the individual units of the decoder equals the total area estimate obtained for a specific timing goal. In addition, the CSEE unit stands out, as it comprises of multiple or to be exact $P-1$ LUTs of size $2t$ words each that stores precomputed information, needed for the evaluation of the error locations and the error magnitudes.

5.2.4 Critical path of decoder

The critical path for different timing goals remained through the Chien Search and Error Evaluation block. Hence, area optimizations on the CSEE unit was difficult.

Comparing the gates report of the CSEE unit generated for 2390ps and 2254ps timing goals, it was observed that a higher number of inverters, buffers and logic gates were utilized to fulfill the latter's timing goal. Moreover, logic gates with extra input terminals and similar/higher drive strengths, which occupy higher area per gate were employed along the critical path, resulting in an increase in the area occupied by the gates. Here, the addition of higher of inverters/buffers helps to decrease the total propagation delay in order to meet the timing goal, but tends to increase the area.

Comparing the gates reports obtained for the SC and KES units for the different timing goals, the area values remained constant as they were not timing critical in comparison to the CSEE unit.

5.3 Power Analysis

As power is a very important design metric, in addition to static timing analysis, probabilistic power analysis was also performed. The 65 nm process technology

library of standard cells implemented using Low-Power Standard-VT transistors, which are characterized at nominal process corners, at 1.2V supply voltage and a temperature of 25C, was used.

5.3.1 Probabilistic Power Analysis

The total switching power, the major power dissipation mechanism, is the sum of the switching power in all the nodes, given by the below equation:

$$\text{SwitchingPower}, \mathrm{P_{SW}} = \sum_{i} f \cdot Vdd^2 \cdot Ai \cdot Ci, \tag{5.5}$$

where Ai represents the switching activity and Ci represents the capacitance of wire segment. Now in order to get an estimate on the power dissipation of the system, some default signal switching probabilities (e.g., Ai=0.01 and 0.25) were considered and their respective toggle rates ($togg$) were calculated and set on the primary data inputs of the system developed. The relation between Ai and $togg$ is shown in below equations.

$$\text{togg} = \frac{f}{1\,\text{GHz}} \cdot Ai \cdot 2 \tag{5.6}$$

$$f = \frac{1}{\text{TimeConstraint}} \tag{5.7}$$

The total power estimates (Dynamic + Leakage power) obtained when various timing goals were set and the toggle rate being varied is shown in Table 5.3.

Table 5.3: Total Power Report of Full System.

		Ai**=0.01**	Ai**=0.25**
Timing goal	**Effort**	**Power**[mW]	**Power**[mW]
2254 ps	medium effort	382.72	399.57
2270 ps	medium effort	379.05	395.86
2390 ps	medium effort	347.34	363.46

From the total power estimates obtained for the full system, it is evident that the total power dissipated increased for a higher Ai or togg value, which is in accordance with the power equation (Ai being proportional to P_{SW}). Furthermore, the total power (including the leakage power) dissipated was observed to be higher when stricter timing constraints were employed, irrespective of the togg value, which can be accounted to the fact that larger circuitry is required to meet the desired timing goal.

The leakage power estimates obtained when various timing goals were set is shown in Table 5.4.

Table 5.4: Leakage Power Report of Full System.

		Ai=**0.25**
Timing goal	**Effort**	**Power**[µW]
2254 ps	medium effort	89.622
2270 ps	medium effort	84.745
2390 ps	medium effort	74.424

A comparison of the total power dissipated by the individual units of the decoder for different timing goals can be seen in Table 5.5.

Table 5.5: Power Report of individual units for timing goals corresponding to 100 Gbps and above.

		2254[ps]	**2270**[ps]	**2390**[ps]
Ai	**Unit**	**Power**[mW]	**Power**[mW]	**Power**[mW]
0.01	SC	121.27	121.23	116.90
0.01	KES	26.75	26.72	18.28
0.01	CSEE	163.34	162.85	277.45

Comparing the relative unit power dissipated by the SC and CSEE units, the total power value of the SC unit (whose area estimate value is much smaller relatively) remains quite comparable to the CSEE unit as it has a highly parallelized dynamic structure, whereas the CSEE unit comprises mostly of multiple static LUTs. Also, similar to the area estimates obtained, the sum of the total power dissipated by the individual units of the decoder equate more or less to the total power obtained for a specific timing goal.

6
Conclusion

In this thesis work, the design and implementation of a very high throughput, single channel Reed-Solomon $RS(255, 239)$ decoder that can reach 100 Gbps is presented. The implemented high speed functional decoder architecture consists of a highly parallelized Syndrome Computation and Chien Search and Error Evaluation blocks. It also uses two systolic Key Equation Solver units connected in parallel, implemented using the ePIBMA algorithm, which is devised through the algorithmic transformations of the Inversionless Berlekamp Massey algorithm (IBMA).

The function of the $RS(255, 239)$ decoder and its verification has been explained in detail. With respect to hardware implementation, the synthesis was performed by mapping the hardware descriptions to standard cells in the 65 nm process technology supplied by ST Microelectronics. Here, the library of standard cells are implemented using Low Power standard-Vt transistors, at 1.2 V supply voltage and a temperature of 25 °C. Post synthesis, the timing and power analysis have been carried out and the estimated values generated by the tool have also been presented.

In conclusion, with regards to the system designed using VHDL, the design has been hard coded for the $RS(255, 239)$ decoder, the outcome of simulations performed in ModelSim, the synthesis results obtained from Cadence RTL Compiler and the maximum throughput achieved by the implemented decoder and its limitations have been discussed. Finally, the proposed architectures have proved to be more time efficient than previously published high-throughput $RS(255, 239)$ decoders.

Bibliography

[1] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono. Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME. RFC 6865, DOI 10.17487/RFC6865, February 2013.

[2] B. Vasi, I. B. Djordjevic and R. K. Kostuk. Low-density parity check codes and iterative decoding for long-haul optical communication systems. *Journal of Lightwave Technology*, pp. 438-446, 2003.

[3] U. Demir and O. Aktas. Raptor versus Reed Solomon forward error correction codes. *International Symposium on Computer Networks*, pp. 264-269, 2006.

[4] J. I. Park, J. Yeon, S. J. Yang and H. Lee. An ultra high-speed time-multiplexing Reed-Solomon-based FEC Architecture for Optical Communications. *Soc Design Conference (ISOCC), 2012 International*, pp. 451-454, Nov. 2012.

[5] G. Agrawal. Fiber-optic communication systems: Third Edition. New York: Wiley-Interscience, pp.2-3, 2002.

[6] IEEE 802.3. In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/IEEE-802.3, June 2019.

[7] 100 Gigabit Ethernet. In *Wikipedia*.
Retrieved from https://en.wikipedia.org/wiki/100-Gigabit-Ethernet, March 2019.

[8] E. R. Berlekamp. Algebraic Coding Theory. New York, NY, USA: McGraw-Hill, 1968.

[9] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving key equation for decoding Goppa codes. Inf. Control, vol. 27, no. 1, pp. 87–99, Jan. 1975.

[10] L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes. U.S. Patent 4 633 470, Dec. 1986.

[11] G. Perrone, J. Valls, V. Torres and F. G. Herrero. High-Throughput One-Channel RS(255,239) Decoder. *21st Euromicro Conference on Digital System Design*, pp. 110-114, Aug. 2018.

[12] N. Glover and T. Dudley. Practical Error Correction Design For Engineers. Broomfield, CO: Cirrus Logic - Colorado, 1991.

[13] B. Sklar and F. J. Harris.The ABCs of linear block codes. *IEEE Signal Processing Magazine*, vol. 21, no. 4, pp. 14-35, July 2004.

[14] R. A. Baby. Convolution coding and applications: A performance analysis under AWGN channel. *International Conference on Communication Networks (ICCN)*, pp 84-88, Nov. 2015.

[15] N. Petra, D. D. Caro and A. G. M. Strollo. A Novel Architecture for Galois Fields Multipliers Based on Mastrovito Scheme. *IEEE Transactions on Computers*, vol. 56, no. 11, pp 1470-1483, Nov. 2007.

[16] Y. Wu. New Scalable Decoder Architectures for Reed–Solomon Codes. *IEEE Transactions on Communications*, vol. 63, no. 8, pp 2741-2761, Aug. 2015.

[17] H. Lee. High-speed VLSI architecture for parallel Reed-Solomon decoder. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 2, pp. 288-294, April 2003.

[18] G. Perrone, J. Valls, V. Torres and F. G. Herrero. Reed-Solomon Decoder Based on a Modified ePIBMA for Low-Latency 100 Gbps Communication Systems. *Circuits, Systems, and Signal Processing*, pp. 1793-1810, Apr. 2019.

[19] Telecommunication Standardization Section, International Telecommunication Union, ITU-T Recommendation G.975. Forward error correction for submarine systems, 1996.