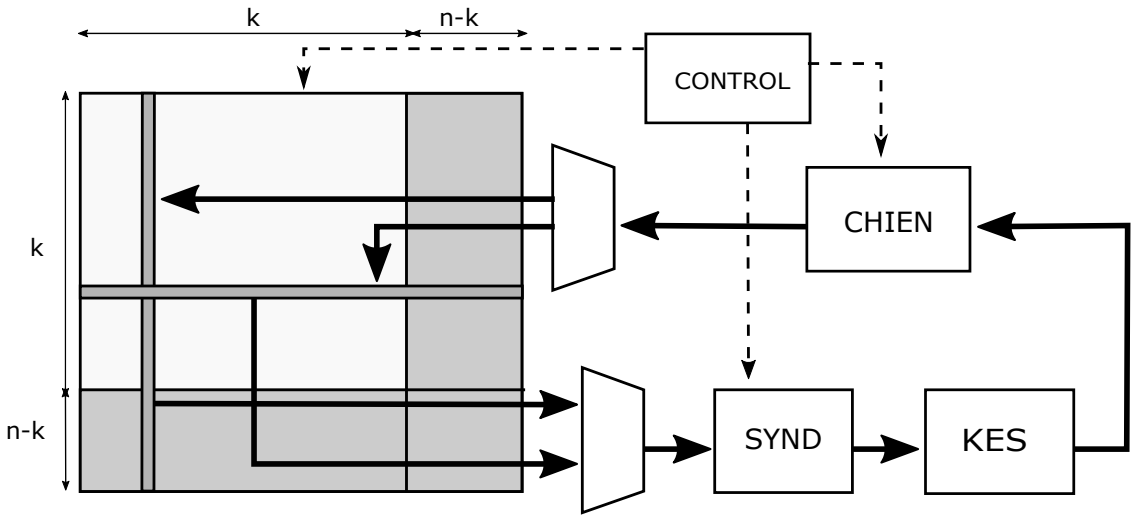


Reconfigurable-Rate Product Decoders for Rate-Adaptable Optical Networks

Master's thesis in Embedded Electronic System Design

VIKRAM JAIN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Reconfigurable-Rate Product Decoders for Rate-Adaptable Optical Networks

VIKRAM JAIN

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Reconfigurable-Rate Product Decoders for Rate-Adaptable Optical Networks
VIKRAM JAIN

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

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Block diagram of the implemented Reconfigurable-Rate product decoder

Gothenburg, Sweden 2018

Reconfigurable-Rate Product Decoders for Rate-Adaptable Optical Networks
VIKRAM JAIN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Optical communication systems use forward error correction (FEC) to reduce the bit-error rate (BER) of the received information bits. The legacy optical links were designed to operate at fixed data rate and never changed parameters during the course of their operation. However, recently rate-adaptable optical systems which can vary parameters over time have gained considerable attention in the research community. These systems can vary their parameters based on the traffic requirements or operator decision. To cater to such systems, FEC schemes which can vary code rate and coding gain are required. For example, if the transmission channel noise is low, lower coding gain is required which means that the code rate can be increased. In this work, we introduce a multi-rate product decoder that can operate in different modes governed by the code rate and the decoding iterations. The implemented multi-rate product decoder provides an estimated net coding gain range of 9.96–10.46 dB at a post-FEC BER of 10^{-15}. The decoder is synthesized in a 28 nm FD-SOI process technology and provides high throughputs in excess of 300 Gbps, reaching 1.6 Tbps for one mode of operation. It also exhibits very low decoding latencies of below 100 ns for all modes of operation. With an efficient clock gating strategy, the power dissipation incurred is below 1 W which corresponds to an energy dissipation per information bit of 1.5 pJ/bit.

Keywords: Forward error correction, Rate-adaptive optical communication systems, Product decoders, multi-rate FEC, coding gain, High-throughput systems.

Acknowledgements

I would like to acknowledge my supervisor Per Larsson-Edefors for providing me the opportunity to work on this challenging task and guiding me throughout the work. We had some amazing discussions and his optimism towards the project propelled my work. I would also like to thank my co-supervisor Christoffer Fougstedt for helping with the technical aspects of the project. This work was financially supported by the Knut and Alice Wallenberg Foundation and Vinnova. I would also like to thank Elma Hurtic and Henri Lillmaa for their contributions on SiBM-based decoders that I have used in this work.
A sincere gratitude to Swedish Institute for funding my Masters studies and supporting my stay in Sweden. Finally I would like to thank my family and friends for being supportive and understanding.

VIKRAM JAIN, Gothenburg, June 2018

List of Abbreviations

ARQ	Automatic Repeat Request
ASIC	Application-Specific Integrated Circuits
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit-Error Rate
CMOS	Complementary Metal-Oxide-Semiconductor
ECC	Error-Control Coding
EON	Elastic Optical Networks
FD-SOI	Fully-Depleted Silicon-On-Insulator
FEC	Forward Error Correction
FFM	Finite-Field Multiplier
FPGA	Field-Programmable Gate Array
FRPD	Fixed-Rate Product Decoder
GF	Galois Field
HD	Hard Decision
KES	Key Equation Solver
LDPC	Low-Density Parity-Check
LSB	Least Significant Bit
MRPD	Multi-Rate Product Decoder
MRMTPD	Multi-Rate Multi-t Product Decoder
MSB	Most Significant Bit
NCG	Net Coding Gain
OH	Overhead
OSNR	Optical Signal-to-Noise Ratio
OTN	Optical Transport Network
SD	Soft Decision
SiBM	Simplified Inverse-free Berlekamp Massey

SNR	Signal-to-Noise Ratio
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

Contents

1
Introduction

Optical communication, in the last few decades, has seen a dramatic increase in the throughput with the development of improved optical devices and technologies. Throughput of 10 Gbps and beyond has become prevalent in today's fiber optical communication systems [1]. However, at such data rates, these systems are marred by several optical impairments and require techniques like forward error correction (FEC) to compensate for the degradation of optical signal-to-noise ratio (OSNR) [2]. When used in optical communication systems, the FEC chips are required to provide high throughput, low post-FEC bit-error-rate (BER) below 10^{-15} and high net coding gain[1] [3]. Introduced in wavelength-division multiplexing optical systems to compensate for amplified spontaneous noise caused by optical amplifiers [4], the first FEC schemes were termed as the first-generation and provided net coding gain in the range of 6 dB [3]. As the throughput scaled to 10 Gbps, other optical impairments such as nonlinear effects, chromatic dispersion and polarization mode dispersion started gaining dominance over other fiber impairments [5]. Second-generation FEC schemes were able to combat these types of impairments and provided a net coding gain of around 8 dB [6]. Today, the throughput and transmission distance for optical systems have become much larger and with that new requirements are being administered. Optical transparency or elimination of optical signal regeneration using opto-electronic devices is becoming a primary constraint. With the lack of solely optical devices for error correction, FEC chips are providing transparent end-to-end systems with the third-generation FEC schemes having coding gain of 10 dB and beyond [7].

Traditionally, wireline and wireless communication systems were designed to operate at a specific information bit rate and never changed over the course of their lifetimes unless replaced by another device [8]. Due to their fixed-rate design, several of their parameters, such as the code used in FEC, the modulation scheme or constellation and the power or strength of the transmitted signal, remained fixed. A detailed description of such communication systems and their parameters is introduced in chapter 2. In wireless communication systems, however, this has changed over the years and the fixed systems are being replaced with adaptive technologies capable of adapting transmission parameters based on the current state of the channel. The adaptivity is introduced to increase the reliability of transmission by lowering the bit rate. This trend has not caught up at the same pace for the optical communication systems owing to a more static transmission medium. However,

[1]The net coding gain is defined as the improvement in signal-to-noise ratio over an uncoded transmission for a certain bit-error rate taking into consideration the redundant bits added for decoding operation.

there has been a gradual shift towards dynamic transmission system in optical networks with the concept of elastic optical networks (EON) garnering a lot of interest in the research community.

The EON paradigm has become the engine driving new flexible and adaptive optical networks which can address the challenge of transmission of data with varying bandwidth demands and reliability [9]. These EON networks are implemented using flexible transceivers which can adapt their parameters based on the channel conditions. Some important parameters that are being considered for flexibility include: modulation formats (BPSK, QPSK, 16-QAM), data rate, and FEC code rate (ratio of number of information bits to total block length). In this work, we focus on the last parameter, i.e., the FEC code rate which can be made flexible by varying the overhead of the code (OH), the ratio of number of parity bits to number of information bits. Ideally, variation in overhead translates into an improvement in net coding gain at the cost of lowering the throughput [10].

1.1 Problem statement and Delimitations

FEC chips are used in optical communication systems to reduce the post-FEC BER below 10^{-15} which provides protection against several noise sources in optical fibers. FEC chips also reduce the required signal transmission power as the errors can be corrected at the receiver and the signal does not require high transmission power to overcome the channel noises. The future optical networks are predicted to become flexible with the ability to vary their parameters based on the channel conditions. Thus, FEC chips used in these flexible optical networks are also required to support flexibility in the form of variable code rate or OH. These next generation FEC chips should also be able to support the increasing throughput requirements of the optical networks. At the same time, the FEC chips should be energy-efficient with low energy dissipation per information bit such that they do not become the bottleneck in terms of energy dissipation in the optical communication system.

1.2 Related Works

We present some of the works that have been published in the topic of flexible-rate FECs and which are relevant to our understanding. Zou et al. [11] presented a rate-adaptive FEC scheme based on low-density parity check (LDPC) codes implemented on a field-programmable gate array (FPGA). The scheme is claimed to achieve a net coding gain of 11.83 to 12.25 dB at a post-FEC BER of 10^{-15} using shortening of LDPC codes to obtain an overhead range from 25 to 42.9 %. However, the throughput of the design is too low for optical communication and the power dissipation is above 1.2 W. Sugihara et al. [12] developed a rate-adaptive FEC scheme by concatenating an LDPC code and a Bose-Chaudhuri-Hocquenghem (BCH) code. The design entails six code overheads in the range of 25.5 to 149.5 % obtained by using row-splitting for LDPC code and shortening for BCH code. The net coding gains reported for these code rates are from 12.0 to 13.5 dB with a throughput range from 50 to 100 Gbps. However, these results are obtained in simulation and no hardware

implementation is reported. Gho et al. [13] proposed a rate-adaptive modulation and FEC scheme with serially concatenated Reed-Solomon codes and hard-decision decoding using shortening and puncturing to attain variable code rates. The design also uses an inner repetition code with soft combining to provide further code rate variation. Simulation results suggest a throughput of maximum 200 Gbps with highest net coding gain of 7.6 dB. Rasmussen et al. [14] demonstrated that rate-adaptive coding can provide power reduction of up to 75 % by reducing code rate during periods of low load. The result sets an important precedence in using FEC chips for reduction in power consumption.

1.3 Approach

The works described in section 1.2 are a class of rate-adaptive FEC chips which have utilized soft-decision decoding which results in a more complex design with target throughputs that are lower than design considerations for fiber optic communication systems. Considering that our multi-rate FEC design is proposed to provide high throughputs, we have chosen hard-decision based decoders [15]. The multi-rate decoder implementation proposed in this work consists of twelve modes of operation defined by the selection of overhead and decoding iterations. Overheads from 21 to 40 % using shortening of component codes are considered. The selection of overhead is based on the requirement of the system to provide throughputs of around 400 Gbps. It can also be noted that overheads above 60 % have been shown to provide diminishing returns in terms of coding gains [16]. The number of decoding iterations considered range from three to five. We do not consider extending the number of Iterations beyond five since it provides diminishing return in terms of net coding gain and also lowers the information throughput [7].

To familiarize the reader with the concept of error-control codes, specifically FEC codes, in chapter 2 we will first introduce the theory behind channel codes and the methods used for encoding and decoding of these codes. In chapter 3, we will deal with the implementation strategy employed for the Reconfigurable-Rate product decoder. In chapter 4, we introduce the evaluation method used in this work. Followed by the preliminary results obtained from the implementation and performance analysis in chapter 5. Finally, a conclusion will be drawn at the end of the report.

2

Introduction to Coding Theory

In this chapter, an introduction to the theory and terminology used in channel coding is provided. We begin with the introduction to the principle of digital communication, error-control, forward error-correction, finite fields, followed by a deeper understanding of Bose-Chaudhuri-Hocquenghem (BCH) codes and their encoding and decoding process and finally an introduction to product-code memory and product decoding.

2.1 Error-Control Coding

Digital communication systems are ubiquitous in a person's day-to-day activities and are used in technologies like cell phones, digital television via satellite or cable, internet through wired (Ethernet, optical fiber, etc.) and wireless (WiFi, WiMax, etc.) media, data storage including optical drives and flash drives. In general, digital communication systems consist of a source transmitting data over a channel to a destination, as shown in Fig. 2.1. The transmission can occur from one location to another (space domain) or it can occur by storing the data at one time and retrieving it at some time later (time domain) [17, 18].

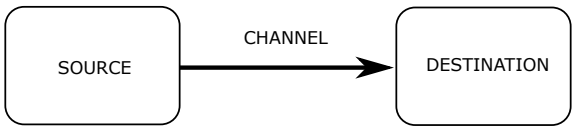


Figure 2.1: Point-to-point digital communication system.

An ideal communication system is characterized by reliability of transmission, i.e., the probability of obtaining the exact data, transmitted by the source, at the destination is 1. In reality, the transmission medium or channel is not a perfect medium but it introduces noise into the transmission data. If the strength of the noise is enough to alter the value of the original data stream, the destination will receive erroneous data. Shannon formalized this communication problem and derived a framework shown in Fig. 2.2 [19]. A source encoder converts the incoming information bits either into a different form or into a more efficient stream by removing redundant bits, i.e., compression. The channel encoder then adds redundancy to the data stream. The resulting data stream is now considered to be coded. When the channel decoder receives the coded data symbols it uses the redundant bits to detect and correct some of the errors. This addition of redundancy to the information symbols is called error-control coding or forward error-correction (FEC) [20].

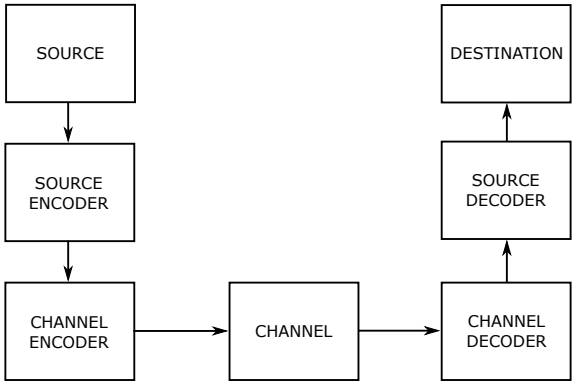


Figure 2.2: Block diagram of digital communication system formalized by Shannon.

FEC is widely used in digital communication systems and derives its name from the facts that coding of data stream is done prior to the transmission and that any bit errors occurring due to channel noise are corrected at the receiving end of the system. The strictly forward nature of FEC provides higher channel-utilization efficiency, compared to protocols based on re-transmission, such as automatic repeat request (ARQ), limited only by the code rate of the FEC. On the other hand, one of the shortcomings of FEC is that if the bit errors are higher than the error-correction capability of the decoder then un-corrected errors are propagated in the decoded data [21].

Selection of FEC code is dependent on several parameters such as net coding gain (NCG), type of channel error, error-correction capability, code rate, code length, complexity, and throughput. The NCG is an important parameter which provides an estimate of transmitted signal power reduction when using coding. NCG is defined as the reduction in the ratio of energy per information bit (E_b) to the noise power density (N_0) — known as E_b/N_0 — when compared to an uncoded system. E_b/N_0 is estimated at a particular bit-error rate (BER), where BER is the ratio of number of bit errors to the number of transmitted bits in a certain time. Type of channel errors also plays an important role when selecting the FEC scheme. In general, there are broadly three error types: random errors, burst errors and byte errors. Random errors, as the name suggests, occur in an unpredictable manner across the data stream. Burst errors occur over a set of information symbols. Finally, byte errors are a subset of burst error which occur over a byte of data symbols. These errors can occur independently or a combination of two or three errors can also happen. Each error type has an equivalent FEC scheme for effective detection and correction of errors, and one scheme for resolving a type of error may not be effective for another type of error [22].

Error-correction capability, code rate, complexity and throughput are parameters that define the implementation of the design and have to be considered for trade-off to obtain the best possible design. For example, increasing the error-correction capability will decrease the code rate and provide better coding gain but at the same time increases the complexity of the decoder. Increase in complexity increases the area of the decoder which in turn increases the power consumption.

Therefore, the selection of FEC scheme requires finding a good trade-off of the above parameters to attain high coding gain at high throughput and low complexity.

2.2 Modelling Channel Coding

A simple model of a transmission channel is shown in Fig. 2.3. A binary message u, comprising of k bits and denoted by $u = u_1, u_2, ..., u_k$ is sent to the channel encoder.

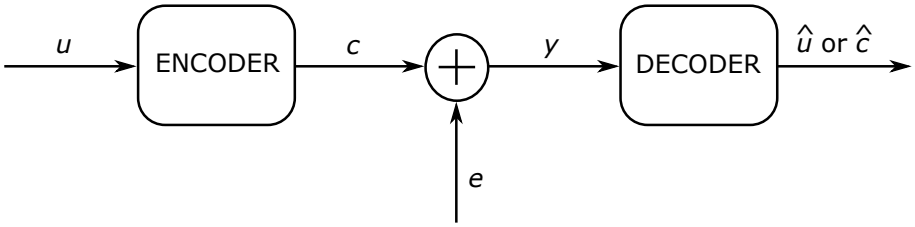


Figure 2.3: Model of channel coding showing the encoder converting information message u to codeword c. In the transmission channel, channel noise e is added to transmitted codeword. At the decoder the received information data, y, is decoded and converted back to the original codeword $\hat{c}$ or the message $\hat{u}$.

Error-correcting codes can be divided into two sub-classes: block codes and convolutional codes. Block coding is performed on incoming information blocks independently and is a memoryless operation, i.e., the codewords are independent of each other. On the other hand, convolutional coding is a memory-based operation in which the output of the encoding depends not only on the current information bits but also on the previous information or codeword bits [21].

In block codes, using some rule, the channel encoder converts the received message into a binary codeword, c of length n bits and denoted by $\mathbf{c} = \{\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_n}\}$ such that $n > k$. The number of redundant bits added to the message is, thus, $n-k$: called parity or check bits. The resulting code rate of such FEC is then denoted by R, where $R = k/n$ [21].

The codeword is then transmitted over the channel and at the receiver a binary message of length n is received which is denoted by y, $\mathbf{y} = \{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_n}\}$, and is called the received vector. This received vector is then decoded by the channel decoder and translated either to the transmitted codeword c to produce the estimated codeword $\hat{c}$ or to the transmitted message u to produce the estimated message $\hat{u}$. If the channel is noiseless then the received codeword y is equal to the transmitted codeword c. In a noisy channel, noise is added to the transmitted codeword to produce the received codeword as given by $y = c + e$, where e is the channel error vector and is given by $\mathbf{e} = \{\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_n}\}$ [21].

Decoding of the received codeword can be divided into two classes: hard-decision (HD) and soft-decision (SD) decoding. HD decoding is a straightforward technique in which the decoder decides whether each bit of the transmitted codeword was a '0' or '1' in the received vector y. On the other hand, SD decoding entails complex algorithms which exploit reliability to decide each bit of the received vector from a multi-valued set of quantization levels. SD decoding can provide higher coding gains than HD decoding but is complex to implement. In this work, our aim

is to achieve high throughput and low power consumption, so henceforth, we will consider only HD decoding [21].

2.3 Finite Fields

Finite fields or Galois fields are an algebraic concept that is commonly used in error-control codes. A field is a set of finite elements on which addition, subtraction, multiplication and division can be applied such that the resulting element of the above operation is an element of the same set, assuming that addition and multiplication are associative, commutative and distributive [20].

A Galois field, $GF(q)$, is a type of algebraic field in which the value of q must be a prime number or a power of prime number. When applied to binary codes, the Galois field considered is also a binary field, $GF(2^m)$, where the elements of the field are binary elements of size m. The primitive element in the Galois field is denoted by α and each nonzero element β of the field can be defined as an integer power of α such that, $\beta = \alpha^i$, where $0 \leq i \leq 2^m - 2$ [20]. The primitive element α is a root of an irreducible polynomial in the Galois field, called the primitive polynomial and represented by $p(x)$, such that $p(\alpha) = 0$.

Two important operations that will be performed on the elements of the GF during decoding are addition and multiplication. Addition and multiplication of elements in $GF(2)$ are modulo-2 operations, which can be further simplified to XOR operations (addition) and AND operations (multiplication). An example of modulo-2 operation applied to addition and multiplication is shown in Table 2.1.

Table 2.1: Modulo-2 addition and multiplication.

$\oplus$	**0**	**1**
0	0	1
1	1	0

$\otimes$	**0**	**1**
0	0	0
1	0	1

Elements of a Galois field are represented in three possible forms; power, polynomial and vector. A GF table consists of the elements of $GF(2^m)$ in all the three representations. The GF table begins with the element α^0 corresponding to 1 in polynomial representation and binary '1' in vector representation. The first m consecutive elements of the table are then populated by shifting the bit '1' to the left until the MSB of the vector is '1' at the m_{th} element (α^{m-1}). The $(m+1)_{th}$ element α^m is calculated by using the primitive polynomial and its property that the primitive element is a root of the primitive polynomial, $p(\alpha) = 0$. The element α^{m+1} is calculated by multiplying α with its predecessor α^m. This process continues until all the required elements of the GF table are populated.

A simple example of generating the elements of a GF table is presented in the text to follow. Let us design a GF table for a $GF(2^3)$ field. Here, $m = 3$ and the primitive polynomial defined for this field is given by $p(\alpha) = 1 + \alpha + \alpha^3$. The first element of the GF table is $\alpha^0 = 1$ or '001'. The second element, obtained by left shift of the binary '1', is '010' or α^1. Similarly, the third element is α^2 or '100'.

To obtain the fourth element or α^3, we will use the primitive polynomial as shown below,

$$\begin{aligned} p(\alpha) &= 1 + \alpha + \alpha^3 = 0 \\ \implies \alpha^3 &= 1 + \alpha \end{aligned}$$

Now, to obtain the fifth element we multiply α with α^3 as shown below,

$$\begin{aligned} \alpha^4 &= \alpha \cdot \alpha^3 \\ &= \alpha \cdot (1 + \alpha) \\ &= \alpha + \alpha^2 \end{aligned}$$

This multiplication of α with its previous element continues till the last element of the table is obtained.

$$\begin{aligned} \alpha^5 &= \alpha \cdot \alpha^4 \\ &= \alpha \cdot (\alpha + \alpha^2) \\ &= \alpha^2 + \alpha^3 \\ &= 1 + \alpha + \alpha^2 \end{aligned}$$

$$\begin{aligned} \alpha^6 &= \alpha \cdot \alpha^5 \\ &= \alpha \cdot (1 + \alpha + \alpha^2) \\ &= \alpha + \alpha^2 + \alpha^3 \\ &= \alpha + \alpha^2 + 1 + \alpha \\ &= 1 + \alpha^2 \end{aligned}$$

In terms of vector representation, bitwise XOR and AND operations are performed to obtain the α^m element and beyond. For example,

$$\begin{aligned} \alpha^3 &= 1 + \alpha \\ Vector &: 001 \otimes 010 = 011 \end{aligned}$$

$$\begin{aligned} \alpha^4 &= \alpha + \alpha^2 \\ Vector &: 010 \otimes 100 = 110 \end{aligned}$$

The GF table obtained for $GF(2^3)$ is shown in Table. 2.2.

Table 2.2: GF table for $GF(2^3)$ elements.

Power	Polynomial	Vector
α^0	1	001
α^1	α	010
α^2	α^2	100
α^3	$1 + \alpha$	011
α^4	$\alpha + \alpha^2$	110
α^5	$1 + \alpha + \alpha^2$	111
α^6	$1 + \alpha^2$	101

2.4 BCH Codes

BCH codes [23] are a class of random error-control cyclic codes (cyclic shift of a codeword gives another word belonging to the same set of codewords) that are used in many error-control coding systems due to their efficiency in correcting random errors and also due to their simpler decoding process using an algebraic method known as syndrome decoding. A BCH code can be expressed by the set of parameters BCH(n, k, t), where n is the block length, k is the number of information bits, and t is the number of errors that can be corrected. Another important parameter is the minimum distance, d, which is the distance between two codewords or the number of positions where two codewords differ. Primitive narrow-sense binary BCH codes are defined using a primitive element α, where α is an element of the Galois field, $GF(2^m)$, with m being a positive integer. For these codes, their parameters are related as,

$$\begin{aligned} n &= 2^m - 1 \\ n - k &= m \cdot t \\ d_{min} &\geq 2 \cdot t + 1 \end{aligned}$$

where n is the block length, k is the number of information bits, t is the number of errors that can be corrected and d is the minimum distance.

These codes are also called t error-correcting BCH codes. Other important parameters of interest include code rate, defined as the ratio of number of information bits to the number of bits in the code block, $R = \frac{k}{n}$, and the overhead defined as $\text{OH} = \frac{n}{k} - 1$.

2.5 Encoding of BCH Codes

Encoding of a linear block code such as BCH codes can be performed by using a generator matrix, G, of the code. The incoming message is multiplied, modulo-2, with the generator matrix to form the code.

$$c = u \cdot G \tag{2.1}$$

where u is the message of length k and c is the resulting code of length n.

2.6 Algebraic Decoding of BCH Codes

As shown in Fig. 2.4, BCH decoding involves three major phases; calculate syndrome, solve key equation and locate error. Syndromes reflect the presence of errors in the received data bits and are used to solve key equation using an error-locator polynomial. This polynomial is then fed to the Chien search, which computes the roots of the polynomial by substituting the primitive elements α^i of the Galois field, where i is an integer from 0 to n, into the polynomial. The roots of this polynomial represent the error locations and in the next step the bit at these locations are flipped to correct the error [21].

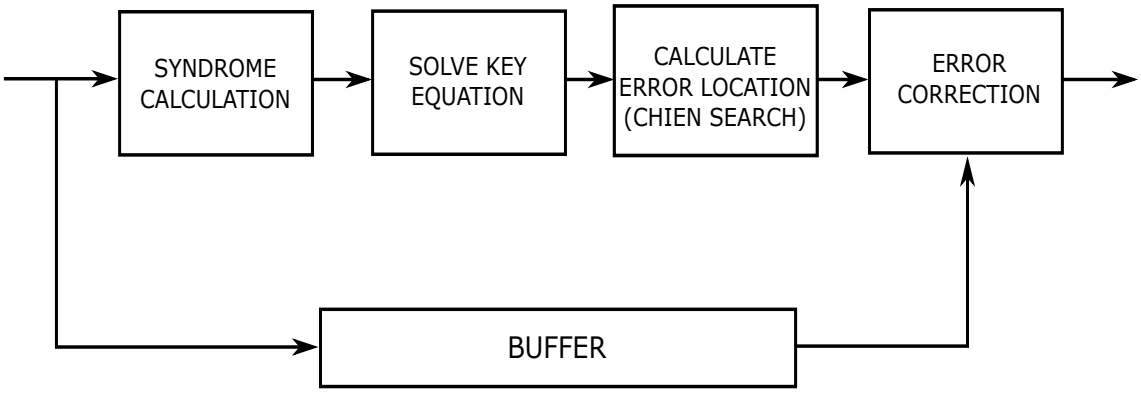


Figure 2.4: Algebraic decoding of BCH codes.

2.6.1 Syndrome Calculation

Syndromes indicate the presence of one or more errors in the received codeword. A value of '1' in the syndrome indicates an error, but the location of the error cannot be computed from the syndrome. For finding the location, the syndromes are fed into the key-equation solver block. Syndromes for linear block codes are calculated using the following equation,

$$S = y \cdot H^T \tag{2.2}$$

where $\mathbf{S} = \{\mathbf{S_1}, \mathbf{S_2}, ..., \mathbf{S_{2t}}\}$ is the syndrome vector, y is the received vector and H is the parity check matrix of the code with H^T representing the transpose. For a t-error correcting code, a total of $2t$ syndromes are required to be calculated.

The parity check matrix, H, for a t error-correcting code is as shown below,

$$H = \begin{bmatrix} \alpha^{n-1} & \alpha^{n-2} & \dots & \alpha^2 & \alpha & 1 \\ \alpha^{2(n-1)} & \alpha^{2(n-2)} & \dots & \alpha^4 & \alpha^2 & 1 \\ \alpha^{3(n-1)} & \alpha^{3(n-2)} & \dots & \alpha^6 & \alpha^3 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \alpha^{2t(n-1)} & \alpha^{2t(n-2)} & \dots & \alpha^{4t} & \alpha^{2t} & 1 \end{bmatrix}$$

Each element of the syndrome, S_i is calculated using the following equation,

$$S_i = y(\alpha^i) = y_0 + y_1 \cdot \alpha^i + y_2 \cdot \alpha^{2i} ... + y_{n-1} \cdot \alpha^{(n-1)i} \qquad (1 \leq i \leq 2t)$$

We defined in section 2.2, the received codeword as a modulo-2 addition of codeword and error vector, $y(x) = c(x) + e(x)$. Substituting this in the above syndrome calculation we get,

$$S_i = c(\alpha^i) + e(\alpha^i)$$

By definition of parity check matrix, codeword multiplied with H is always equal to zero and thus the above equation becomes,

$$S_i = e(\alpha^i)$$

Assuming that v ($v \leq t$) number of errors have occurred in the received codeword at the positions given by $j_1, j_2, ..., j_v$, the error polynomial can then be represented as:

$$e(x) = e_{j_1} \cdot x^{j_1} + e_{j_2} \cdot x^{j_2} + ... + e_{j_v} \cdot x^{j_v}$$

Thus,

$$\begin{aligned} S_i &= e_{j_1} \cdot \alpha^{i(j_1)} + e_{j_2} \cdot \alpha^{i(j_2)} + ... + e_{j_v} \cdot \alpha^{i(j_v)} \\ &= e_{j_1} \cdot (\alpha^{j_1})^i + e_{j_2} \cdot (\alpha^{j_2})^i + ... + e_{j_v} \cdot (\alpha^{j_v})^i \end{aligned}$$

The $2t$ syndromes are, thus, calculated as,

$$\begin{aligned} S_1 &= e_{j_1} \cdot (\alpha^{j_1}) + e_{j_2} \cdot (\alpha^{j_2}) + ... + e_{j_v} \cdot (\alpha^{j_v}) \\ S_2 &= e_{j_1} \cdot (\alpha^{j_1})^2 + e_{j_2} \cdot (\alpha^{j_2})^2 + ... + e_{j_v} \cdot (\alpha^{j_v})^2 \\ S_3 &= e_{j_1} \cdot (\alpha^{j_1})^3 + e_{j_2} \cdot (\alpha^{j_2})^3 + ... + e_{j_v} \cdot (\alpha^{j_v})^3 \\ &\vdots \\ S_{2t} &= e_{j_1} \cdot (\alpha^{j_1})^{2t} + e_{j_2} \cdot (\alpha^{j_2})^{2t} + ... + e_{j_v} \cdot (\alpha^{j_v})^{2t} \end{aligned} \qquad (2.3)$$

For binary BCH codes, the elements of the error vector are essentially 1. The set of equations shown in eq. 2.3 can be simplified further with only $\alpha^{j_1}, \alpha^{j_2}, ..., \alpha^{j_v}$ as the unknowns.

2.6.2 Key-equation Solver

The $2t$ syndromes form a system of linear equations or key-equations shown in eq. 2.3. A solution to the equations will provide the error location. However, solving such large set of equations, especially when t is large, becomes a tedious process. Instead, the equations are solved indirectly by using a polynomial whose coefficients are derived from the syndromes. This polynomial is called an error-locator polynomial and is denoted by $\lambda(x)$ whose degree depends on the value of t. The roots of the error-locator polynomial are the reciprocals of $\alpha^{j_1}, \alpha^{j_2}, ..., \alpha^{j_v}$.

$$\lambda(x) = \lambda_t x^t + ... + \lambda_2 x^2 + \lambda_1 x + \lambda_0 \tag{2.4}$$

Several algorithms can be used to find a solution to the key-equations, by forming the error-locator polynomial, such as Euclidean, Berlekamp-Massey and its optimized forms. A more simpler algorithm used in high-throughput systems is the Peterson algorithm [24].

Peterson Algorithm

The Peterson algorithm is a non-iterative approach for computing the error-locator polynomial [24]. Syndromes and the coefficients of the error-locator polynomial are related to each other by Newton's identity and a resultant system of linear equations are derived from this property [25]. The system of equations can be succinctly expressed in a matrix form as,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ S_2 & S_1 & 1 & 0 & \dots & 0 & 0 \\ S_4 & S_3 & S_2 & S_1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & S_{2t-5} & \dots & S_{2t-v} & S_{2t-v-1} \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_v \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \\ -S_5 \\ \vdots \\ -S_{2t-1} \end{bmatrix}$$

$$\implies A \cdot \lambda = S \tag{2.5}$$

Initially, the value of v is set to t and the matrix A is constructed. If t errors have occurred then the determinant of A will be non-zero and in that case, eq. 2.5 is solved. If the determinant is zero, then fewer than t errors have occurred and the last two rows and two rightmost columns of the matrix A are deleted. The same procedure is repeated till the determinant is non-zero and then eq. 2.5 is solved to derive the coefficients, as shown in Fig. 2.5.

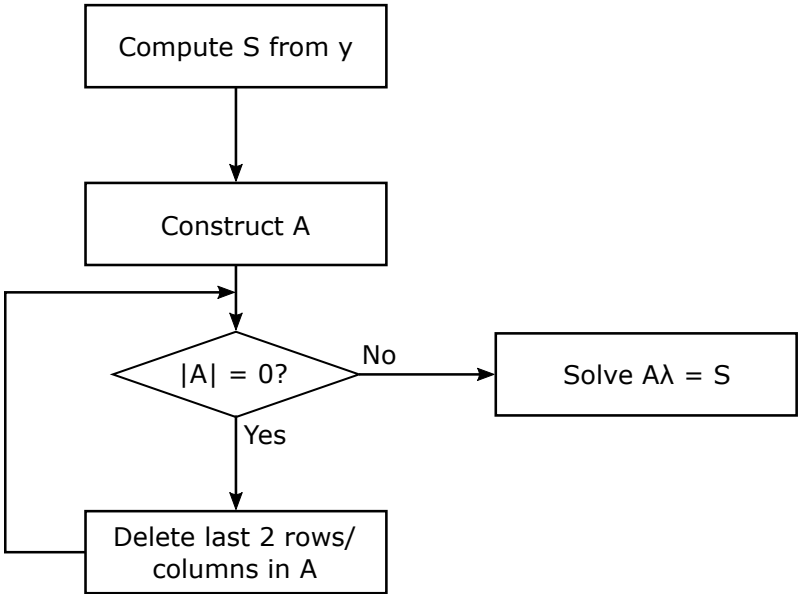


Figure 2.5: Flowchart for Peterson's algorithm used to generate error-locator polynomial using syndromes.

For smaller values of t, the solution is straightforward as shown in the results below [25].

Single error-correction

$$\lambda_0 = 1$$
$$\lambda_1 = S_1$$

Double error-correction

$$\lambda_0 = 1$$
$$\lambda_1 = S_1$$
$$\lambda_2 = \frac{S_3 + S_1^3}{S_1}$$

Triple error-correction

$$\lambda_0 = 1$$
$$\lambda_1 = S_1$$
$$\lambda_2 = \frac{S_1^2 \cdot S_3 + S_5}{S_3 + S_1^3}$$
$$\lambda_3 = S_3 + S_1^3 + S_1 \cdot \lambda_2$$

Four error-correction

$$\begin{aligned}
\lambda_0 &= 1 \\
\lambda_1 &= S_1 \\
\lambda_2 &= \frac{S_1(S_7 + S_1^7) + S_3(S_1^5 + S_5)}{S_3(S_1^3 + S_3) + S_1(S_1^5 + S_5)} \\
\lambda_3 &= S_3 + S_1^3 + S_1 \cdot \lambda_2 \\
\lambda_4 &= \frac{S_1^2 \cdot S_3 + S_5 + (S_1^3 + S_3) \cdot \lambda_2}{S_1}
\end{aligned}$$

Inverse-free Peterson algorithm

The Peterson algorithm shown in the previous section requires an inverse (divide) operation which makes the key-equation solver complex to implement in hardware. A reformulation of the equations obtained from Peterson's algorithm can be done by multiplying the equations of the coefficients with their denominator on both sides [26]. This eliminates the division operation and reduces the complexity significantly.

Triple error-correction

$$\begin{aligned}
\lambda_0 &= S_1^3 + S_3 \\
\lambda_1 &= S_1 \cdot \lambda_0 \\
\lambda_2 &= S_1^2 \cdot S_3 + S_5 \\
\lambda_3 &= \lambda_0^2 + S_1 \cdot \lambda_2
\end{aligned} \tag{2.6}$$

Four error-correction

$$\begin{aligned}
\lambda_0 &= S_3(S_1^3 + S_3) + S_1(S_1^5 + S_5) \\
\lambda_1 &= S_1 \cdot \lambda_0 \\
\lambda_2 &= S_1(S_7 + S_1^7) + S_3(S_1^5 + S_5) \\
\lambda_3 &= S_1^4 \cdot S_5 + S_1^2 \cdot S_7 + S_3(S_1^6 + S_3^2) \\
\lambda_4 &= S_1^3(S_1^7 + S_7) + S_3(S_1^7 + S_1 \cdot S_3^2 + S_7) + S_5(S_1^5 + S_1^2 \cdot S_3 + S_5)
\end{aligned} \tag{2.7}$$

2.6.3 Chien Search

The roots of the error-locator polynomial are calculated using a commonly used algorithm called Chien search [27]. As stated earlier, Chien search computes the roots of the polynomial by substituting the primitive elements α^i of the Galois field, where i is an integer from 0 to $n-1$, into the error-locator polynomial $\lambda(x)$. The element α^i is a root of the polynomial if $\lambda(\alpha^i) = 0$, where i represents the position of the error.

To demonstrate the working of Chien search algorithm, let us assume that BCH$(15, 7)$ with $t = 2$ is used. Let us also assume that the process of key-equation

solver has generated the error-locator polynomial based on the received codeword as $\lambda(x) = 1 + \alpha^{13}x + x^2$. The algorithm starts by substituting x with the first element of Galois field, $GF(2^4)$, $\alpha^0 = 1$ into the polynomial.

$$\begin{aligned}\lambda(1) &= 1 + \alpha^{13} \cdot 1 + 1^2 \\ &= 1 + \alpha^{13} + 1 \\ &= \alpha^{13}\end{aligned}$$

Since $\lambda(1) \neq 0$, the first element is not the root of the equation. This process is repeated by substituting $\alpha, \alpha^2, ..., \alpha^{14}$ into the polynomial and the roots are obtained at α^4 and α^{11}.

$$\begin{aligned}\lambda(\alpha^4) &= 1 + \alpha^{13} \cdot \alpha^4 + \alpha^8 \\ &= 1 + \alpha^{17} + \alpha^8 \\ &= 1 + \alpha^{15} \cdot \alpha^2 + \alpha^8 \\ &= 1 + \alpha^2 + 1 + \alpha^2 \\ &= 0\end{aligned}$$

Note: Addition in GF is a modulo-2 or XOR operation, so addition of similar element is 0. The values of α^i is derived from the GF table.

$$\begin{aligned}\lambda(\alpha^{11}) &= 1 + \alpha^{13} \cdot \alpha^{11} + \alpha^{22} \\ &= 1 + \alpha^{24} + \alpha^{22} \\ &= 1 + \alpha^{15} \cdot \alpha^9 + \alpha^{15} \cdot \alpha^7 \\ &= 1 + \alpha^9 + \alpha^7 \\ &= 1 + \alpha + \alpha^3 + 1 + \alpha + \alpha^3 \\ &= 0\end{aligned}$$

Thus, there are errors at the positions 4 and 11 of the received codeword. A simple bit flip at these positions is then used to correct the error.

2.7 Varying Code Properties

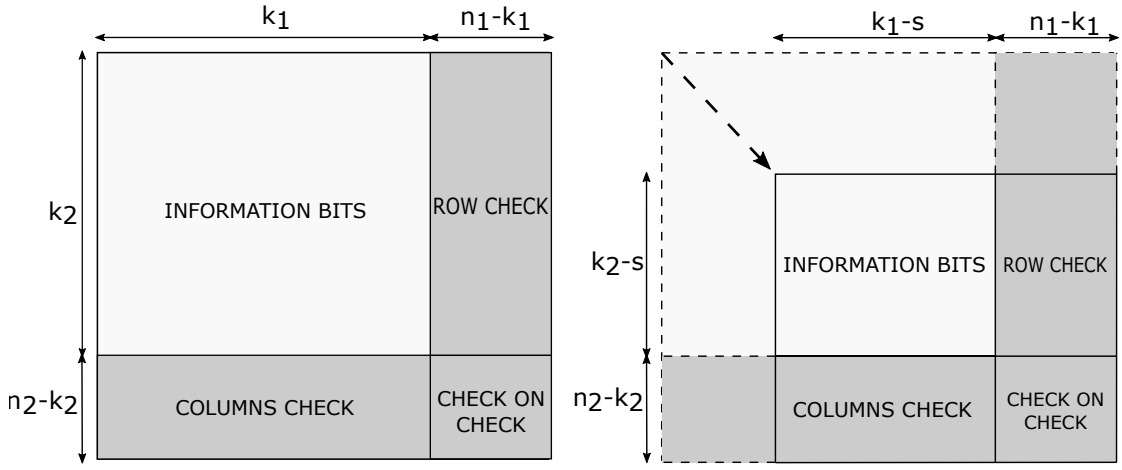
The overhead (or code rate) of component codes can be varied by using the following methods [22]:

- Shortening : Process of removing some information bits from the codeword during the encoding process. The removed bits are then never transmitted over the channel. The information bits can be removed either at the MSB or the LSB, keeping the number of parity bits intact. Shortening results in an increased overhead from the original code—the mother code. It is well known that by increasing the code overhead, coding gain can be increased at the cost of throughput [28]. The shortened BCH codes are denoted by the parameters $n_s = n - s$ and $k_s = k - s$, where s is the number of bits shortened.

- Lengthening : Process of introducing additional information into the codeword, keeping the number of parity check bits unchanged. The resulting codeword is represented with parameters $n_l = n + l$ and $k_l = k + s$. Lengthening lowers the overhead, decreasing the coding gain but increasing the information throughput.
- Extending : Extending the code means additional bits are introduced to the parity check bits. The resulting code are represented with $n_e = n + e$ and $k_e = k$. Extending increases the overhead and improves the coding gain.
- Puncturing : Process of removing parity bits and substituting erasures. Puncturing decreases the overhead and the resulting code is represented with $n_p = n - p$ and $ke = k$.
- Augmenting : Process of adding codewords to the set of codewords C. The resulting minimum distance of the code becomes lower than the original.
- Expurgating : Process of removing codewords from the set C. Minimum distance of the resulting code increases from the original.

2.8 Product-code Memory

A product-code memory can be formed by combining smaller component codes [29–31]. The resulting concatenated codes can provide higher error-correction capability than the component codes. A product-code memory is constructed by encoding information bits row-wise, using a row component code, followed by column-wise encoding, using column component codes, as shown in Fig. 2.6a. The twofold encoding over both data and parity gives a resulting minimum distance which is the product of minimum distance of individual component codes and is given by $d_1 \cdot d_2$.



(a) Original product-code memory **(b)** Shortened product-code memory

Figure 2.6: Product-code memory is a $n_1 \times n_2$ memory as shown in (a). In (b), the resulting memory with shortening applied is shown.

Product decoding can be implemented using a product-code memory which is iteratively decoded by row and column component decoders. Incoming data bits

are loaded into the memory as row and column codes. The code from the rows are then forwarded to the decoder where errors are corrected and the corrected code is written back. This is followed by a similar process for the column codes. This process is then repeated for a number of iterations to remove the errors [32].

An advantage of product-code memory over using long FEC codes is that by employing low-complexity component codes in the product-code memory, the complexity of the product decoder is reduced since smaller component decoders are required. In contrast, the long FEC codes require much more complex component decoders with higher n, k and t. Component code selection plays a vital role not only on the error-correction performance, but also on the speed and encoding/decoding complexity of the product decoder.

To form a product-code memory, let there be two BCH component codes, BCH(n_1, k_1, t_1) and BCH(n_2, k_2, t_2). A product-code memory is formed by the concatenation of the two component codes. The product-code memory formed is an $n_1 \times n_2$ matrix, with information bits forming a $k_1 \times k_2$ matrix inside it, as shown in Fig. 2.6a. The code rate of the resulting product-code memory is a product of the code rate of individual component codes, $R = R_1 \cdot R_2 = \frac{k_1}{n_1} \cdot \frac{k_2}{n_2}$, the overhead is $\mathrm{OH} = \mathrm{OH}_1 \cdot \mathrm{OH}_2 = \frac{n_1 \cdot n_2}{k_1 \cdot k_2} - 1$, and the error-correction capability is $t_1 \cdot t_2$. A similar process is applied when using shortened BCH codes to construct the product-code memory. The shortened product-code memory becomes a $(n_1 - s) \times (n_2 - s)$ matrix with the enclosed information being reduced to a $(k_1 - s) \times (k_2 - s)$ matrix, as shown in Fig. 2.6b. Since in shortening, the shortened bits are never transmitted the component decoders corresponding to the shortened bits are removed, resulting in reduced hardware.

2.9 Coding Gain

Coding gain is one important metric used to evaluate a FEC coding scheme. Coding gain is defined as the reduction in E_b/N_0 from uncoded to coded system to obtain a particular error rate, where E_b is the average energy of transmission per bit and N_0 is the noise power density. In a nutshell, coding gain is the reduction in signal power (lower E_b/N_0) required to achieve the same BER as would have been achieved without the coding, since the errors can be corrected. The net coding gain of a coded system is measured at the decoder output and is the coding gain achieved taking into consideration the redundant bits added for decoding operation. Throughout this work, net coding gain will be used to evaluate the FEC performance. Fig. 2.7 represents a typical example of deriving net coding gain from the BER vs E_b/N_0 plot.

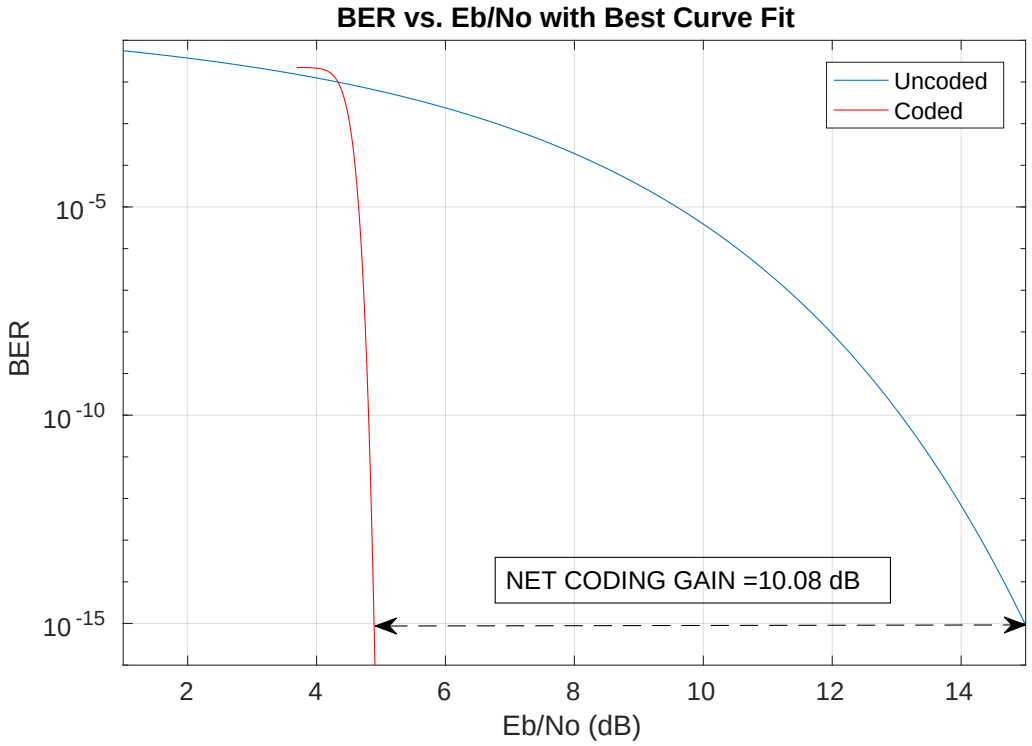


Figure 2.7: BER vs E_b/N_0 plot showing an example of deriving net coding gain.

3

Implementation of the Multi-rate Product Decoders

In this chapter, we describe the architecture of the product decoders implemented during this work starting with the baseline design on which the multi-rate scheme is applied. Followed by the architecture of the multi-rate product decoder. We look at individual modules and the added design over the baseline design. Finally, an extension design with variable error-correction capability is presented.

3.1 Baseline Design

The architecture and the implementation of the baseline fixed-rate product decoder is borrowed from a recently published staircase decoder [15]. Instead of using a staircase memory, a product-code memory is used. The basic block diagram of the fixed-rate product decoder is shown in Fig. 3.1. SYND represents the syndrome calculation unit, KES represents the key-equation solving unit, while CHIEN handles Chien search, as described in section 2.6. Central to the product decoder is the product-code memory which is an array that stores the incoming codewords as described in section 2.8. In this design, the product-code memory is symmetric, i.e., $n = n_1 = n_2$ and $k = k_1 = k_2$, however, asymmetric codes (codes with different code lengths) can also be used but that would require using different Galois field (GF) component decoders.

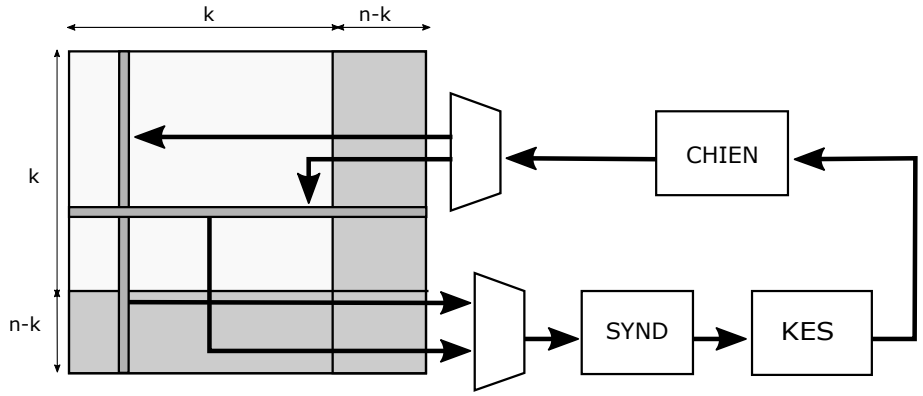


Figure 3.1: Block diagram of baseline product decoder. The baseline decoder consists of a product-code memory and n component decoders with SYND, KES and CHIEN unit.

A total of n BCH component decoders are used for decoding the codes, providing a highly parallel decoding operation. As stated in section 2.8, a state machine is used for iterative decoding, in which all the row codewords are decoded by the component decoders and the errors are corrected followed by decoding of all the column codewords and the errors are corrected in the product-code memory. This process is repeated for a number of iterations, at the end of which the codewords in the product-code memory are outputted.

3.1.1 Clock Gating in Baseline Design

The baseline design uses extensive clock gating to provide an energy-efficient design. The component decoders used are pipelined with a pipeline stage after the SYND unit and another pipeline stage before the CHIEN unit. The syndrome unit indicates presence of errors if the syndrome calculated is non-zero. For any component decoder, if the syndrome calculated is zero then the codeword is believed to be error-free. In that case, the component decoder pipeline is disabled. This prevents the unwanted decoding computation when no errors are present. If all the syndromes of the n component decoders are zero then the codewords in the product-code memory are believed to be correct and the product-code memory is clock gated.

3.1.2 Syndrome Calculation

Syndrome calculation is done as described in section 2.6.1 by multiplying the incoming codeword, y, with the transpose of parity check matrix H. Fig. 3.2 outlines a block diagram of syndrome calculation as implemented in hardware. The parity check matrix is generated in MATLAB using an in-built function, `cyclgen(n,ply)`, where n represents the code length and `ply` is the generator polynomial. The generator polynomial is also generated using a function in MATLAB, `bchgenpoly(n,k)`, with k being the number of information bits.

Elements of H and syndrome belong to the $GF(2^m)$ but the symbols used for the input data are binary and, thus, multiplication and addition can be simplified to AND and XOR operations respectively. As seen in the figure, the elements of y are ANDed with the elements of each row in H and the results are element-wise XORed to produce the syndromes. This process is repeated for all the columns of H to obtain $2t$ syndromes each of size m. Only odd syndromes are calculated due to a property of the Galois field [25].

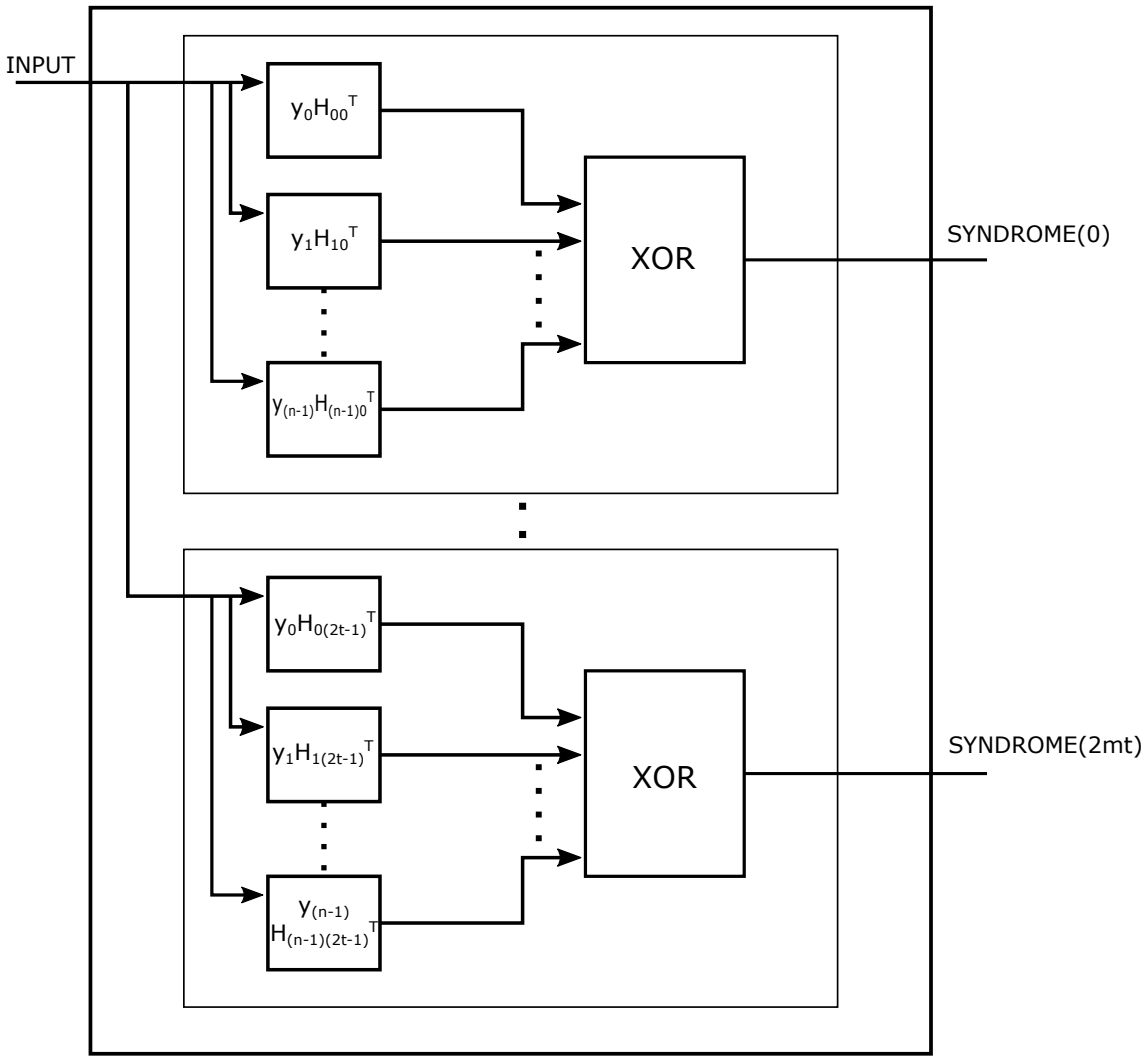


Figure 3.2: Block diagram of syndrome calculation in baseline design showing the vector-matrix multiplication of incoming codeword and transpose of parity check matrix.

3.1.3 Key-equation Solver

The syndromes obtained in the syndrome calculation module are forwarded to the key-equation solver to generate the error-locator polynomial as explained in section 2.6.2. Since the elements of the syndromes are defined in $GF(2^m)$, multiplication is done in polynomial basis using the Mastrovito algorithm and are called finite-field multipliers (FFM) [33]. The odd syndromes are used to generate the coefficients of the error-locator polynomial as per eq. 2.6. The error-locator polynomial is then forwarded to the Chien search for finding its roots.

3.1.4 Chien Search

Chien search is used to solve the error-locator polynomial by finding the roots of the polynomial as described in section 2.6.3. A root of the polynomial represents an error at the position represented by the power of the GF element substituted into the polynomial. Fig. 3.3 shows a block diagram for computing the error vector using Chien search. The input to the Chien search unit are the coefficients generated in the KES unit. Each coefficient is multiplied with a power of α derived from the error-locator polynomial. This process is done for each bit of the n bit codeword. For example, the error-locator polynomial for $t = 3$ code is represented by the equation,

$$\lambda(x) = \lambda_3 x^3 + \lambda_2 x^2 + \lambda_1 x + \lambda_0$$

For the first bit, x is replaced with α. The coefficient λ_1 is multiplied with α, λ_2 is multiplied with α^2 and λ_3 is multiplied with α^3. For the second bit, x is replaced with α^2 and the coefficient λ_1 is multiplied with α^2, λ_2 is multiplied with α^4 and λ_3 is multiplied with α^6 and so on till the n^{th} bit. All the multiplications are done in polynomial basis using FFMs. The resulting products are XORed and then a bitwise OR operation is applied to generate the error vector bits. The output error vector is an all zero vector with '1's at the position where error has occurred. This vector is then used to correct errors in the product-code memory.

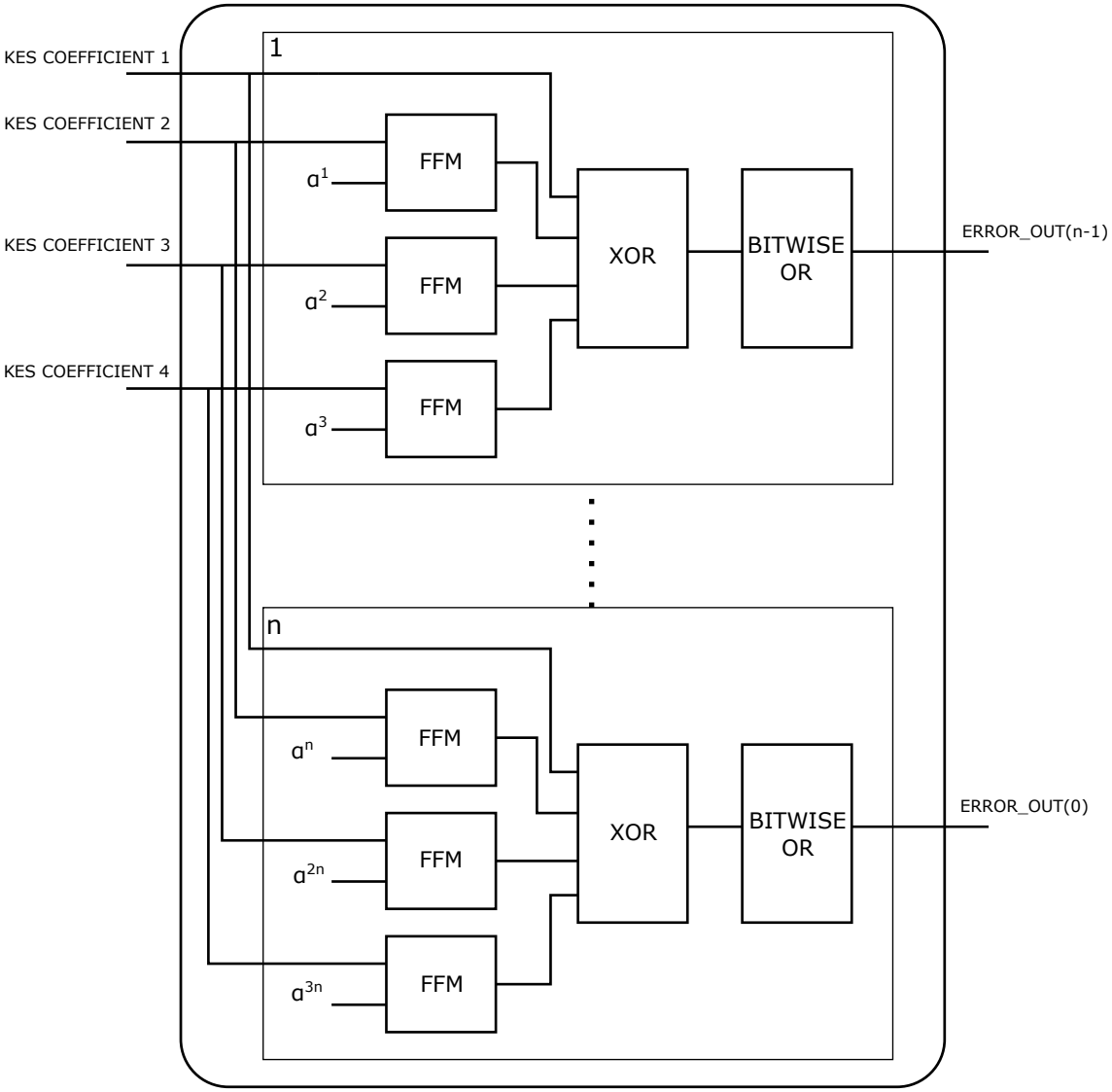


Figure 3.3: Block diagram of Chien search in baseline design showing the substitution of primitive element α^i into the error-locator polynomial at each bit position.

3.2 Multi-Rate Product Decoder Architecture

Multi-rate FEC can be implemented in two ways: rate-adaptive systems and multi-rate systems. Rate-adaptive systems are designed such that the decoder can calculate the post-FEC BER at runtime. If the post-FEC BER value is higher than a threshold then information is relayed back to the transmitter over a feedback line. When transmitter receives this information, it reduces the code rate to increase the coding gain. The presence of the feedback mechanism introduces a latency into the system which is detrimental to high speed operation in optical links. In this thesis, we implement a multi-rate system which operates in different modes of operation defined by varying code rate and the decoding iterations. Since multi-rate systems are non-adaptive in nature, code rates and decoding iterations have to be varied manually at the transmitter. The configuration data for the code rate or decoding

iteration can be carried by a special field in the frame of optical transport network (OTN) such that any change in code rate or iteration can be relayed to the decoder at the receiver. When the decoder receives a configuration change, a simple state machine can be used to dynamically switch the code rate.

In this work, the code rate is varied by using shortening of the component codes which varies the overhead. Decoding iterations of the product decoder is implemented as a variable in the product decoding state machine. Modes can also be based on varying the block length of the codes, but chip area increases rapidly with larger codes and also the amount of unused memory portions for smaller block length modes increases. Also, with block length the parameter m of the Galois field $GF(2^m)$ changes which changes the applicable finite-field arithmetic. The implication of this is that separate component decoders for higher block length are required in addition to the existing component decoders, increasing the area and complexity of the design.

The baseline component decoders are fully parallel which means that introducing reconfigurability does not affect system throughput as there are no feedback loops in the baseline design. We introduce reconfigurability to the baseline design presented in section 3.1 and the resulting multi-rate product decoder (MRPD) architecture is shown in Fig. 3.4. The codes selected in this work are BCH$(255, 231)$ for both row and column to fulfill the requirement of high throughput and high coding gain. A detailed description of the choice is provided in section 3.4. Thus, the product memory is an $n \times n$ array with information bits forming an inner array of dimensions $k \times k$. The control logic handles the different modes of decoder operation, including different overheads/rates and different number of iterations. The control logic is applied only to the product-code memory, SYND and the CHIEN unit. The KES unit, on the other hand, is left unaltered because the KES unit does not change with varying code-rates but only when there is a change in error-correction capability t.

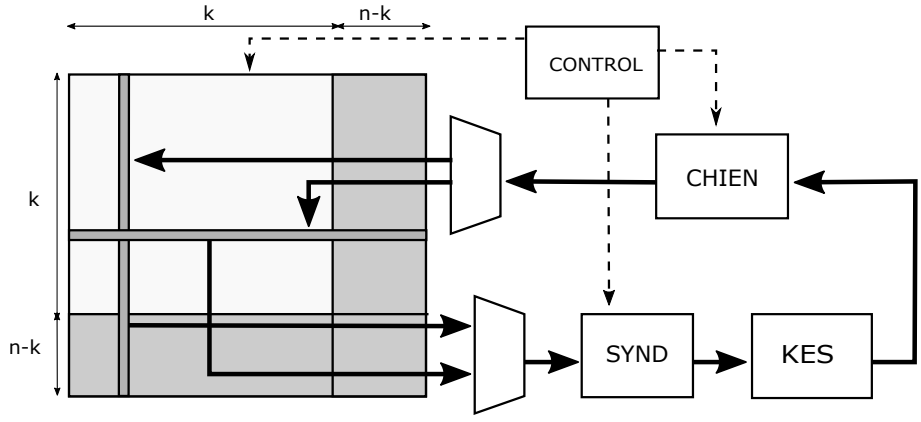


Figure 3.4: Block diagram of multi-rate product decoder. Control unit is added to the baseline design to handle the modes of operation.

3.2.1 Product-code Memory

The product-code memory forms the backbone of the product decoder. Incoming information bits are stored into the product-code memory by storing the code bits

into columns of the memory matrix. When shortening is applied, by replacing the information bits with zeros, the corresponding memory matrix is reduced as shown in Fig. 2.6b. In the baseline design, when shortening is applied by removing information bits, the corresponding part of the product memory is also removed. The resulting product-code memory is an $n-s \times n-s$ array with a $k-s \times k-s$ information array (s is the number of shortened bits). However, in the multi-rate design, the shortened bits cannot be removed but rather have to be disabled. The shortened bits are flushed and gated such that any switching activity at these positions is prevented. Gating also helps in providing a power-efficient design since the switching activity is prevented at the shortened part of the memory.

A bit mask of size n is used for gating the memory, such that at the shortened bit positions the bits of the mask are set to 1 or 0 (the shortened bits are based on the mode selected), as shown in Fig. 3.5. The incoming data bits are then ANDed with the bit mask and stored into the product-code memory as rows and columns. This ensures that the shortened bits are set to 0 which prevents switching activity at these positions. Also, the presence of gating of incoming signal to be stored into the product-code memory flushes the shortened part of memory when switching between different modes, especially when moving from longer to shorter codes.

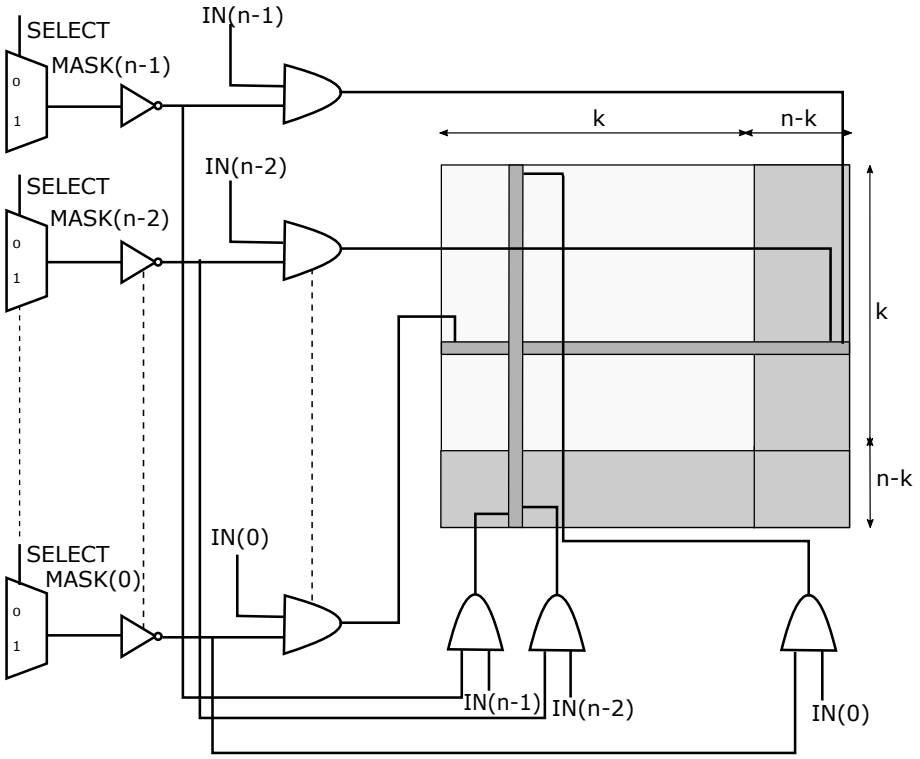


Figure 3.5: Product-code memory for MRPD design with the gating mechanism to handle shortening.

3.2.2 Syndrome Calculation

Syndromes, as described in section 2.6.1, are calculated using $s = y \cdot H^T$, where s is the syndrome, y is the received codeword and H is the parity check matrix. As

explained in section 3.1.2, the equation can be simplified to form an XOR tree such that each syndrome is a set of XOR operations of the codeword bits at positions where the parity check matrix is 1. When shortening is applied to the design, a number of lower XOR operations (equal to the number of shortened bits) can be removed from the hardware.

In the multi-rate design, the full XOR tree is retained and when shortening is applied, the codeword is shifted to the most significant bits (MSBs). This is achieved by using a set of multiplexers in the SYND unit (Fig. 3.4), as shown in Fig. 3.6. The incoming codeword bit signals are multiplexed such that the MSB of the shortened code $(n - s - 1)$ is always at the MSB of the mother code and the lower significant bits (equal to shortened bits s) are set to 0 to prohibit signal switching. The variable s represents the number of shortened bits and $s = 0, s1, s2$ and $s3$ for the four modes. The resulting set of bits are passed to the XOR tree, which generates a set of $2t$ syndromes of size m bits.

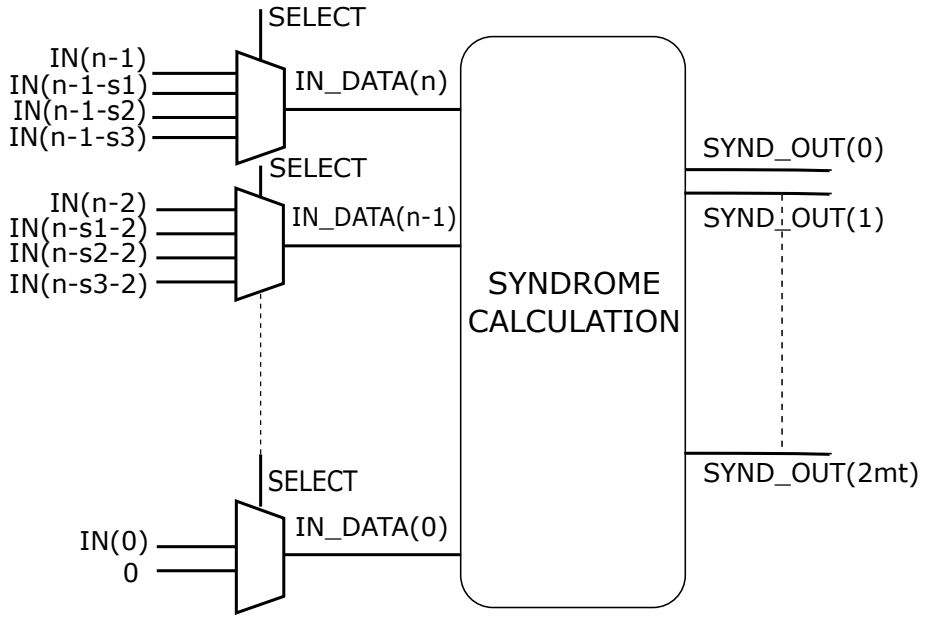


Figure 3.6: Syndrome calculation unit for MRPD design. A set of multiplexers are used to shift the incoming message and passed to the baseline syndrome calculation.

3.2.3 Chien Search

The error-locator polynomial generated in the KES unit (Fig. 3.4) is forwarded to the CHIEN unit where α is substituted into the polynomial, as described in section 2.6.3. section 3.1.4 describes the implementation of Chien search in the baseline design. Finite-field multipliers (FFM) are used to multiply the coefficients with incremental degree of α, the resulting values are then XORed together to form the value of the polynomial. A value of zero for the polynomial represents an error at that position. The CHIEN unit is designed to be fully parallel to achieve high throughput. Due to its parallel nature, $n \cdot t$ FFMs are used for every component decoder. To support all modes of operation, all $n \cdot t$ FFMs are available in the hardware making it necessary

to utilize gating to prevent unnecessary computation and power dissipation.

At the shortened bits (s) positions for which the computation is not required, the coefficients are ANDed with enable signals, as shown in Fig. 3.7. The enable signals are set to 0 or 1 depending on the selected mode of operation. For example, when the SELECT is 00 (base OH), none of the coefficients are gated. When SELECT is 01, GATED_1 is set to zeros, i.e., the coefficients for the $s1$ bits at the upper bit positions are disabled. When SELECT is 10, GATED_1 and GATED_2 are set to zeros, i.e., the coefficients for the $s1$ and $s2$ bits at the upper bit positions are disabled. Finally, when SELECT is 11, GATED_1, GATED_2 and GATED_3 are set to zeros, i.e., the coefficients for the $s1$, $s2$ and $s3$ bits at the upper bit positions are disabled. The resulting coefficients are then sent to the baseline Chien search described in section 3.1.4. In this way, the inputs to the FFMs are set to 0 and any unnecessary computation is prevented. Finally, the resulting error signal is shifted back to the LSB using the set of multiplexers shown in Fig. 3.7.

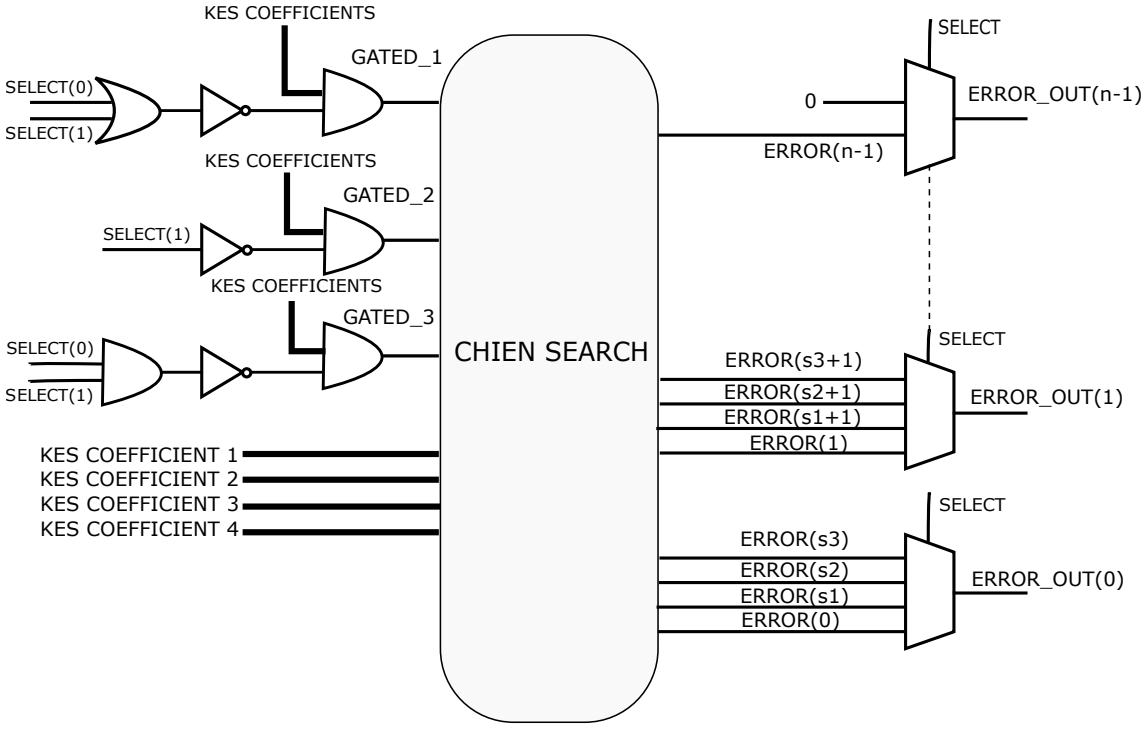


Figure 3.7: Chien search for MRPD design with gating to handle shortening at the input and multiplexers at the output to shift the error signal.

3.3 Multi-rate, Multi-t Product Decoder

The MRPD design described in section 3.2 is based on fixed error-correction capability with $t = 3$. By increasing t a further improvement of coding gain can be achieved, but with the increase in t comes an increase in area and power dissipation. An extension design, multi-rate, multi-t product decoder (MRMTPD), was designed with varying code rate, decoding iteration and a variable t between three and four. The design of MRMTPD is similar to the MRPD with two major difference; the

selection of OHs for the design when operated with $t = 4$ and an ability to vary t. The OHs that were selected with $t = 4$ were: the base OH of 30 %, OH 33 %, OH 41 % and OH 49 %. The selection of these OHs are governed by the lower limit of the base OH and higher limit of achieving high throughputs. When operated with $t = 3$, the OHs remained the same as selected in the MRPD design. Due to the difference in the OHs between the $t = 3$ and $t = 4$ operations, additional hardware was included into the product-code memory, SYND unit and the CHIEN unit. Also, the KES unit was replaced with a design with reconfigurable t.

3.3.1 Product-code Memory

Similar to section 3.2.1, the product-code memory of the MRMTPD includes a gating mechanism operated based on the selection of OH. However, due to difference in actual values of OHs, additional multiplexers were added to generate masks in the case of $t = 4$, as shown in the Fig. 3.8. A set of multiplexers, with the flag ECC as the select signal, were then used to select between the $t = 3$ or $t = 4$ mask. For example, when operating with $t = 3$ and mode 2 of 25 % OH ($n = 227, k = 203$), 28 bits at the upper bit positions have to be masked. On the other hand, when operating with $t = 4$ and mode 2 of 33 % OH ($n = 239, k = 207$), only 16 bits of upper bit positions have to be masked. Due to the difference in the OHs, the same set of multiplexers cannot be used for both the ts and hence two set of multiplexers are used whose output is then selected based on the value of t.

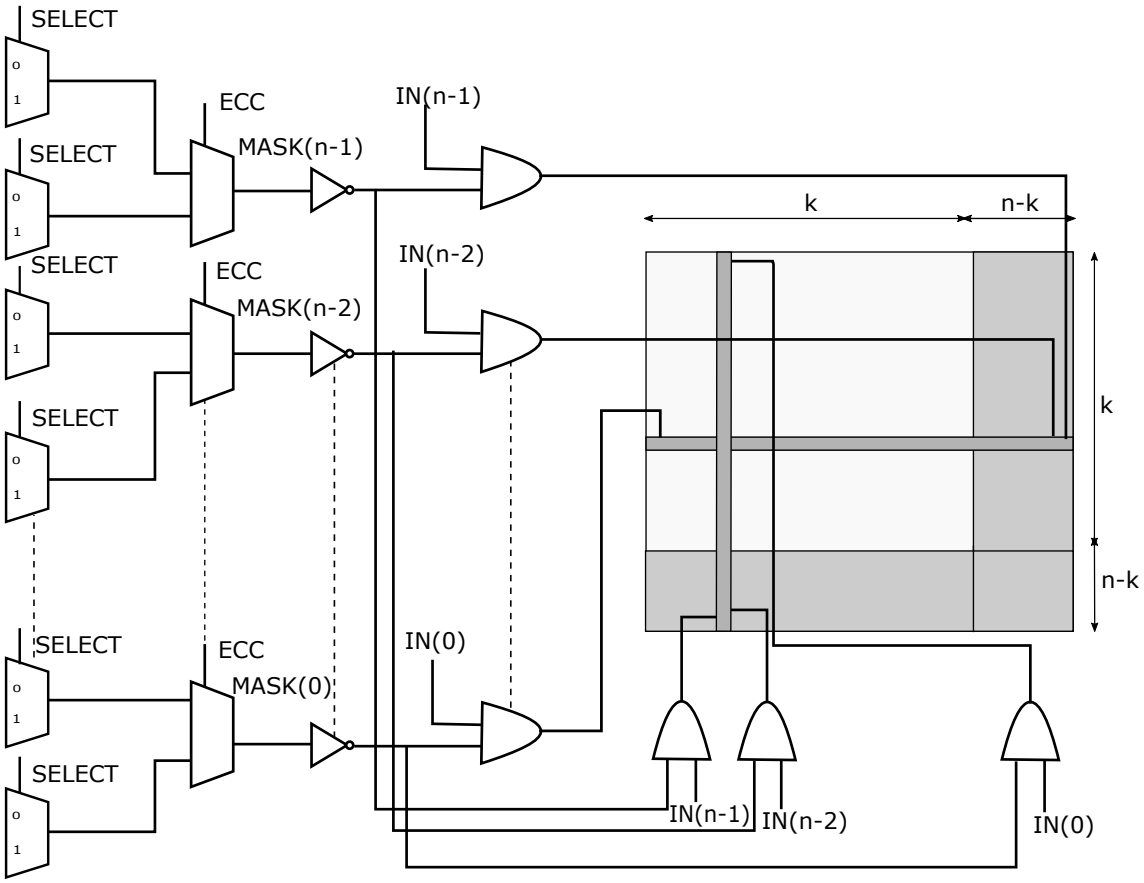


Figure 3.8: Product-code memory for MRMTPD with additional multiplexers to handle $t = 4$ mode of operation.

3.3.2 Syndrome Calculation

Fig. 3.9 shows the hardware implementation of the syndrome calculation unit in the MRMTPD design. Two sets of multiplexers, each for the OHs for different t, are used to shift the incoming data to the MSB. The output from the two sets are sent to another multiplexer which selects the shifted data based on the t selected and sends the output to the syndrome calculation unit described in section 3.1.2. When operating in $t = 4$ mode, the number of syndromes calculated are four (only odd syndromes), while in $t = 3$ mode, three syndromes are calculated. So, the baseline syndrome calculation unit is required to calculate four syndromes to fulfill the requirement. But, when operating in $t = 3$ mode the fourth syndrome is not required and, thus, should be gated. A multiplexer is used to set the inputs to fourth syndrome calculation to zero to prevent switching activity and reducing the power dissipation.

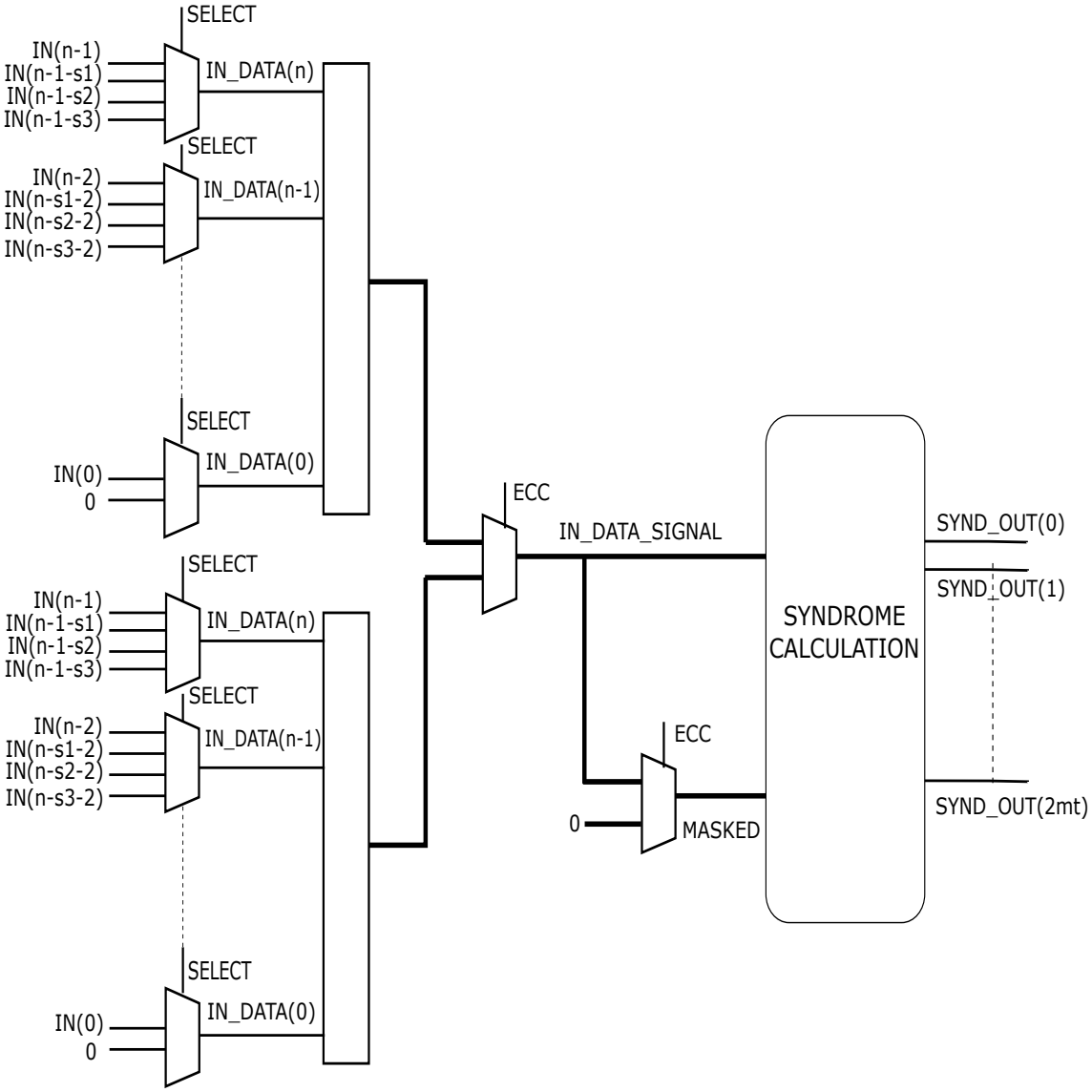


Figure 3.9: Syndrome calculation unit for MRMTPD with additional multiplexers to handle $t = 4$ mode of operation.

3.3.3 Key-equation Solver

As the MRMTPD design is a multi-t design, the KES unit of the product decoder should be made reconfigurable. Fig. 3.10 shows a block diagram of the implementation of the KES unit in the MRMTPD design. The coefficient generation blocks shown are similar to the baseline design described in section 3.1.3 and consists of coefficient generation for $t = 3$ and $t = 4$. Implementation of the coefficient generation for $t = 3$ was presented in section 3.1.3. A similar strategy is applied for $t = 4$ with the implementation of eq. 2.7.

A multiplexer with ECC selection signal, shown in Fig. 3.10, is used to shut down the calculations in the $t = 4$ coefficient generation block by setting the input to this block to zero when the design operates with $t = 3$. However, when operating in $t = 4$ mode, the $t = 3$ coefficient generation block is not disabled because a

lot of the terms generated in $t = 3$ coefficient generation block are re-used in the $t = 4$ coefficient generation. Finally, at the output a set of multiplexers based on the operating ECC are used to select the relevant coefficient values and forwarded to the CHIEN unit.

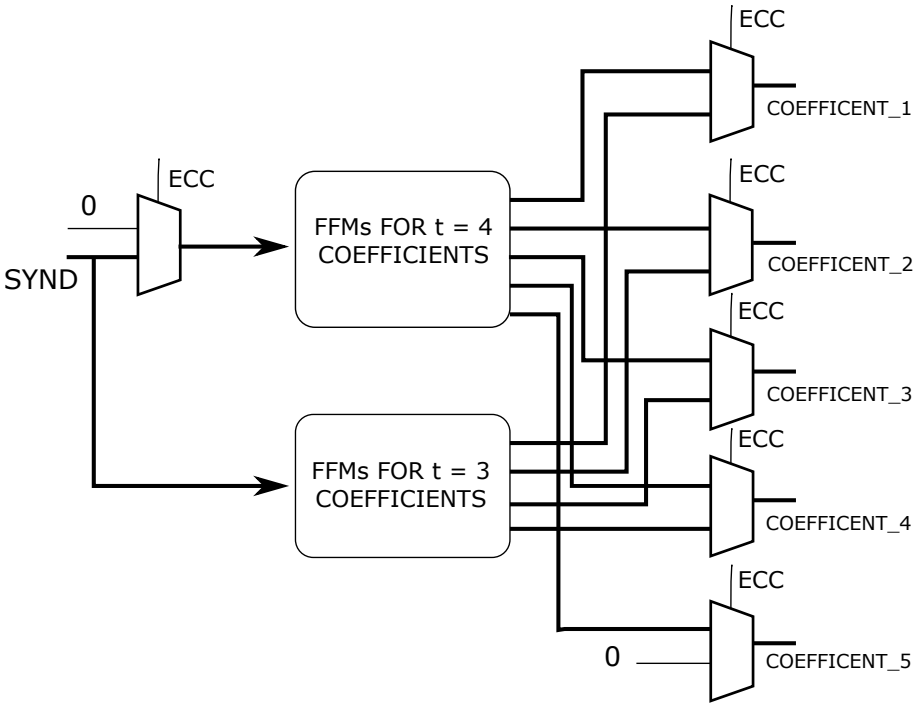


Figure 3.10: Reconfigurable KES unit for the MRMTPD design with additional multiplexers to handle $t = 4$ mode of operation.

3.3.4 Chien Search

The CHIEN unit of the MRMTPD is similar to the CHIEN unit of MRPD design described in section 3.2.3, with the addition of hardware to handle different set of OHs of $t = 3$ and $t = 4$ modes. Fig. 3.11 shows a block diagram of the Chien search as implemented in hardware. KES coefficients coming from the KES unit are gated at the positions of shortened bits. Due to the difference in OHs for $t = 3$ and $t = 4$, the number of shortened bits are also different which means that the same set of gated signals cannot be used. As seen in Fig. 3.11, a set of additional multiplexers (MUXES) is used to select the gated coefficient based on the t mode. For example, when operating with $t = 3$ and mode 2 of 25 % OH, the number of shortened bits is 28. But with $t = 4$ and mode 2 of 33 %, the number of shortened bits is 16. The bits from 17 to 28 have to be multiplexed such that when operating with $t = 4$ GATED_1 at these bits are replaced with GATED_2. This is applied to all the modes and the output of the gated coefficients is sent to the baseline CHIEN SEARCH module. At the output, shifted signals for both the $t = 3$ and $t = 4$ case are generated and based on the ECC mode the final error signal is outputted.

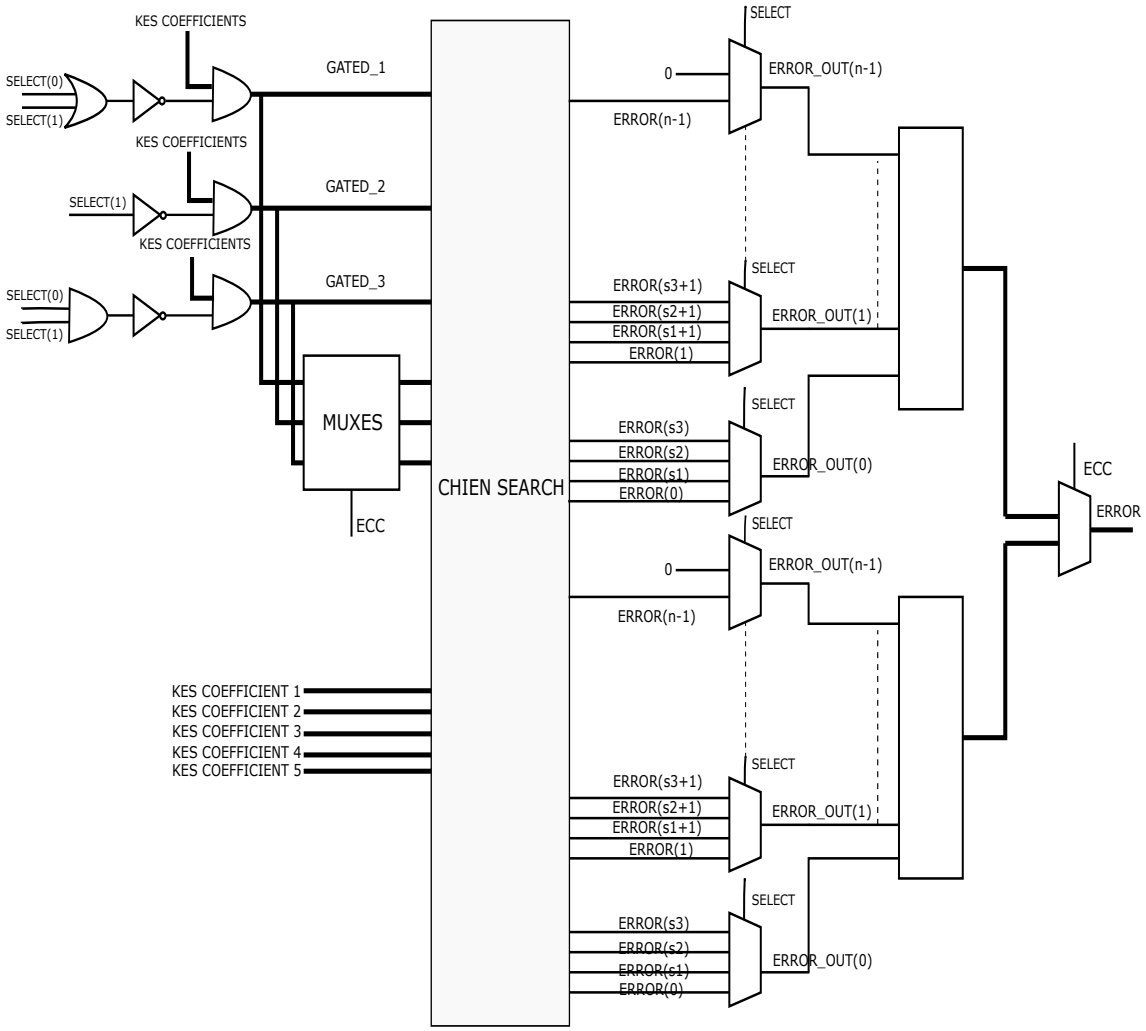


Figure 3.11: Chien search for MRMTPD with additional multiplexers to handle $t = 4$ mode of operation.

3.4 Design Consideration

To fulfill the combined requirement of high coding gain and operation at high throughput, a product-code with BCH$(255, 231)$ component codes is selected for $t = 3$ and BCH$(255, 223)$ component codes are selected for $t = 4$ code. The BCH(255) codes have higher number of information bits than the lower BCH(127) codes and even with high decoding latency the information throughput will be high. On the other hand, BCH(255) codes will require less hardware than BCH(511) codes making them ideal for our requirement. We have confined the design to $t = 3$ and $t = 4$, as these have been shown to ensure coding gains above 10 dB [34]. Product decoders with $t = 2$ have a higher error floor which leads to a lower coding gain at the required post-FEC BER of 10^{-15}. The multi-rate scheme applied is based on the selection of code overheads (OH) and number of decoding iterations to achieve a useful range of code rates. The modes of operation used for the $t = 3$ design are: base OH of 21 %, OH 25 %, OH 33 % and OH 40 %. Table 3.1 provides a summary

of code parameters for $t = 3$ used in this work. For $t = 4$, the modes of operations are: base OH of 30 %, OH 33 %, OH 41 % and OH 49 %. Table 3.2 represents the code parameters for $t = 3$.

The number of decoder modes is extended further by varying the number of iterations from three to five. The design trade-off is such that the more iterations, the higher the coding gain and the lower the throughput. For example, more iterations can be combined with a lower-rate code to yield higher coding gain. The decoding iterations are limited to five because it has previously been reported that more than five iterations yields diminishing coding gain return [34].

Table 3.1: Product and component code parameters for $t = 3$

	n	**k**
Product-code overhead = 21 %	255	231
Product-code overhead = 25 %	227	203
Product-code overhead = 33 %	180	156
Product-code overhead = 40 %	155	131

Table 3.2: Product and component code parameters for $t = 4$

	n	**k**
Product-code overhead = 30 %	255	223
Product-code overhead = 33 %	239	207
Product-code overhead = 41 %	202	170
Product-code overhead = 49 %	177	145

4

Evaluation Method

Considering that chip area and power dissipation are issues for reconfigurable architectures, in which extra circuits are needed to support the change between modes, we have implemented and evaluated the multi-rate product decoder in the framework of an application-specific integrated circuit (ASIC). Also, in order to meet the demand to have high throughput, ASIC platforms are selected for this design. A simple flow graph showing the process of evaluation is shown in Fig. 4.1.

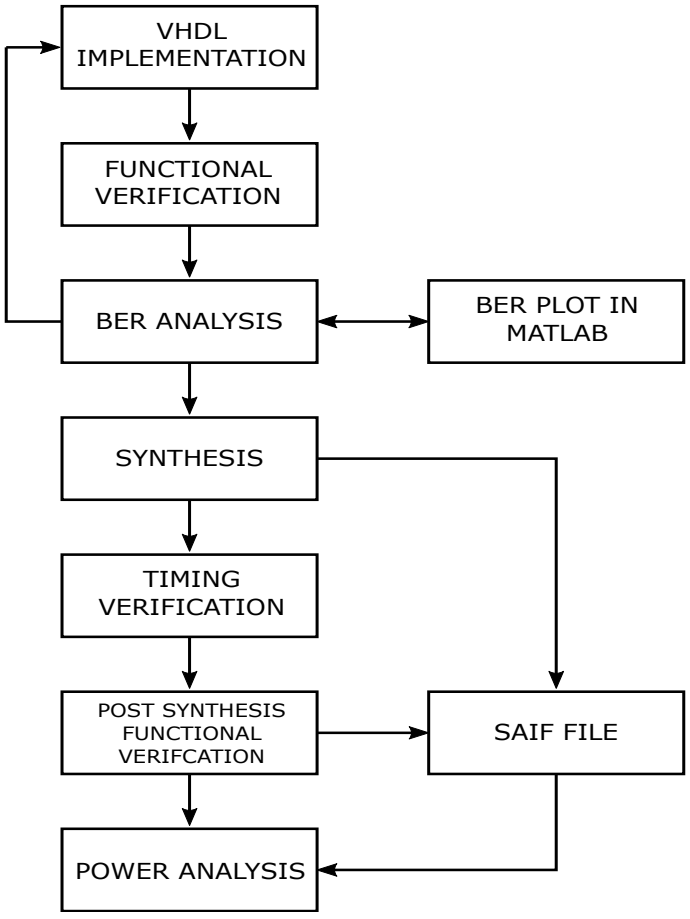


Figure 4.1: Flow graph for evaluation of designs.

The design of the product decoder is undertaken using VHDL. The design is simulated for functional verification using a VHDL testbench (Testbench 1) in Cadence Incisive. The testbench is used to generate uniformly-distributed data which are encoded using a product encoder. To simulate a binary-symmetric channel

in VHDL, a random number generator is used to generate errors with a probability corresponding to the input signal-to-noise ratio (SNR). The errors are then added to the encoded data which are passed through the decoder to verify its operation. If the decoded data is not equal to the encoded data, then an error counter is incremented and the error count is printed on the console using assert statement. A timestamp is printed on the console by the testbench at the end of each block decoding. Using the difference in timestamps between two consecutive block decoded, we calculate the block decoding latency. The throughput is calculated by dividing the number of information bits (k^2) with the block decoding latency.

A second testbench (Testbench 2) which generates random error with probability given by input SNR and adds the error to a datastream of zeros is used for BER analysis. A bash script is used to sweep the input SNR which is passed to the testbench. The erroneous data is then passed through the decoder and the output is compared with a zero vector. The number of ones present in the decoded data provides the error count. The post-FEC BER is calculated by dividing the error count by the number of bits in all the decoded blocks, where number of bits in all the decoded blocks is calculated by multiplying the number of decoded blocks with n^2. The results of the post-FEC BER are extrapolated in MATLAB using the function `berfit` to obtain the BER plot. For optical communication, a common target post-FEC BER is 10^{-15} and, thus, this threshold is used to define the net coding gain (NCG) in this work.

The net coding gain estimated using the VHDL implementation is used for the selection of the overheads. A MATLAB code for product decoder was written but the built-in functions for decoding BCH codes were unable to support shortening. Thus, it was decided to use VHDL simulations to select the values of OHs. The target of selection of the overhead was to generate a wide range of OHs such that the highest possible OH can support the requirement of high throughput. The intermediate OHs were selected such that a minimum of 0.1 dB of coding gain is available between any two OHs selected.

After the BER analysis, the VHDL code is synthesized. The design is synthesized in Cadence Genus to a low-leakage library of a 28-nm 0.9-V fully-depleted silicon-on-insulator (FD-SOI) process technology, using the slow-slow corner and a temperature of 125 °C with the effort set to high. Based on an architectural analysis, the target clock rate is set to 610 MHz. The clock gating parameter of the synthesis environment is set to true. When the clock gating is enabled in the synthesis tool, the tool handles the insertion of clock gating into the design. On a successful synthesis, the Verilog netlist and the SDF constraint file are generated. The timing report is used to verify that the design meets the timing constraint set during the synthesis. The area report from Cadence Genus provides the total area of the design as well as area of individual modules. Post-synthesis functional verification is performed on the Verilog netlist using Testbench 2.

Testbench 2 is used to generate a switching activity file called SAIF file. When generating the SAIF file, the input SNR to the testbench is set to the value derived from the BER plot at the threshold of 10^{-15} for a particular mode of the product decoder. In Cadence Incisive the Verilog netlist, the SDF constraint file and the testbench are imported and an SAIF file is generated. In the Genus environment,

the technology libraries are set to the typical-typical corner at a temperature of 25 °C. Clock gating is enabled, the clock tree buffers are set and the max clock tree fanout is set to 20. The SAIF file is then read into the Genus environment and using the report power functionality of the tool the power report is generated. In addition, clock-tree power is also estimated in Genus by providing the chip area derived during synthesis. The total power is then derived by adding the average power obtained from the power report and the clock tree power report. This process is then repeated for all the twelve modes of operation of the MRPD design, the MRMTPD design and for the baseline design. The energy dissipation per information bit is derived by taking the ratio of the energy dissipated during 1 second to the information throughput (Gbps).

The complete process of functional verification, BER and power analysis is performed individually for all the modes of operation of the MRPD design. For comparisons to prior art, we also analyze the performance of a fixed-rate product decoder and a product decoder based on the conventional iterative approach of SiBM [35,36]. The SiBM product decoder was synthesized and simulated with error-correction capability of $t = 3$ and the fixed-rate product decoder was synthesized and simulated with both $t = 3$ and $t = 4$. Also, an extension design with multi-rate, multi-t product decoder (MRMTPD), is implemented and simulated with both $t = 3$ and $t = 4$.

5

Results

In this section we describe the results obtained from evaluations performed on the VHDL decoder implementation. The section begins with the results obtained for the evaluation of the baseline design as described in section 3.1, followed by evaluation results of the multi-rate product decoder (MRPD) (section 3.2) and the multi-rate multi-t product decoder (MRMTPD) (section 3.3).

5.1 Reference Designs

The algorithm used in the key-equation solver (KES) plays an important role in terms of throughput and power dissipation of the decoder. In general, Berlekamp-Massey (BM) and its different optimizations like simplified inverse-free BM (SiBM) are iterative in nature and require at least t clock cycles to compute the error-locator polynomial. This iterative operation has a detrimental effect on the decoder as it lowers the throughput and increases the energy dissipation due to increased number of cycles. Thus, approaches such as Peterson and direct-solution or inverse-free Peterson algorithms which compute the error-locator polynomial in a single cycle are important alternatives to the iterative approach.

As discussed in chapter 4, evaluation of two designs were carried out. First was the implementation of a fixed-rate product decoder (FRPD) based on the BCH(255,231) component codes and using direct-solution Peterson KES [37]. The FRPD design was evaluated with error-correction capability $t = 3$, called FRPD1, and $t = 4$, called FRPD2. Second was the implementation of a product decoder with SiBM based KES with $t = 3$. Table 5.1 provides the results obtained when the designs were evaluated with decoding iteration of four and their corresponding base OHs. These designs are used as baselines for the design considerations when implementing the multi-rate designs.

In our implementations, the product decoder with SiBM achieved throughputs that were 40 % lower than those of the FRPD1. In terms of power dissipation also, the SiBM approach had 10 % more power dissipation than the FRPD1 with the energy dissipation per information bit being 77 % higher for SiBM. However, with increasing t this observation may not hold true as the direct-solution approach scales faster in terms of area and power dissipation than does the SiBM approach.

Table 5.1: Evaluation results of FRPD1, FRPD2 and SiBM designs at number of iterations = 4 and base OH

	FRPD1	**FRPD2**	**SiBM**
Cell area (mm^2)	6.69	9.11	7.54
Code rate, R	0.82	0.76	0.82
Throughput (Gbps)	1252	1167	775.16
Block decoding latency (ns)	42.61	42.61	68.838
Power @ BER 10^{-15} (mW)	788.49	1305.56	866,62
Energy @ BER 10^{-15} (pJ/info. bit)	0.63	1.11	1.12
Estimated net coding gain (dB)	10.055	10.5	10.08

5.2 Multi-rate Product Decoder

We evaluated our MRPD design for BER analysis considering three cases: number of iterations equal to three, four and five. In each case, all the four modes based on OH were evaluated. Figs 5.1-5.3 show the post-FEC BER as a function of E_b/N_0 for three, four and five iterations respectively. Assuming three, four and five iterations, Fig. 5.4 shows the output BER as a function of E_b/N_0 for the extreme decoder modes, i.e., 21 % and 40 % OH. Coding gain range estimated are 0.31 dB, 0.33 dB and 0.38 dB for three, four and five iterations respectively. A wider range of 0.5 dB can be achieved if the decoder is operated with three iterations at the base OH and with five iterations at 40 % OH. In addition, the net coding gains (at a post-FEC BER of 10^{-15}) for different iterations and the decoder modes are shown in Table 5.2, Table 5.3 and Table 5.4.

The range of coding gain depends on the block length of the component codes. In this design, the minimum coding gain is limited by the overhead of the block length of the mother code, i.e., BCH$(255, 231)$, which is 21 %. Conversely, the upper limit of coding gain, i.e., the highest OHs is limited by the constraint of achieving high throughputs. The coding range could be extended further by utilizing higher OH, but this has the drawback of reducing the throughput. Another alternative to increase coding gain range is to utilize longer components codes, e.g., BCH$(511, 484)$, whose product decoder has a baseline OH of 11 %; but this leads to a more complex decoder with higher energy dissipation. Another design aspect that can be considered for increasing the coding gains is to increase the error-correction capability t. However, increasing t increases the coding gains of individual modes, but the range would remain the same.

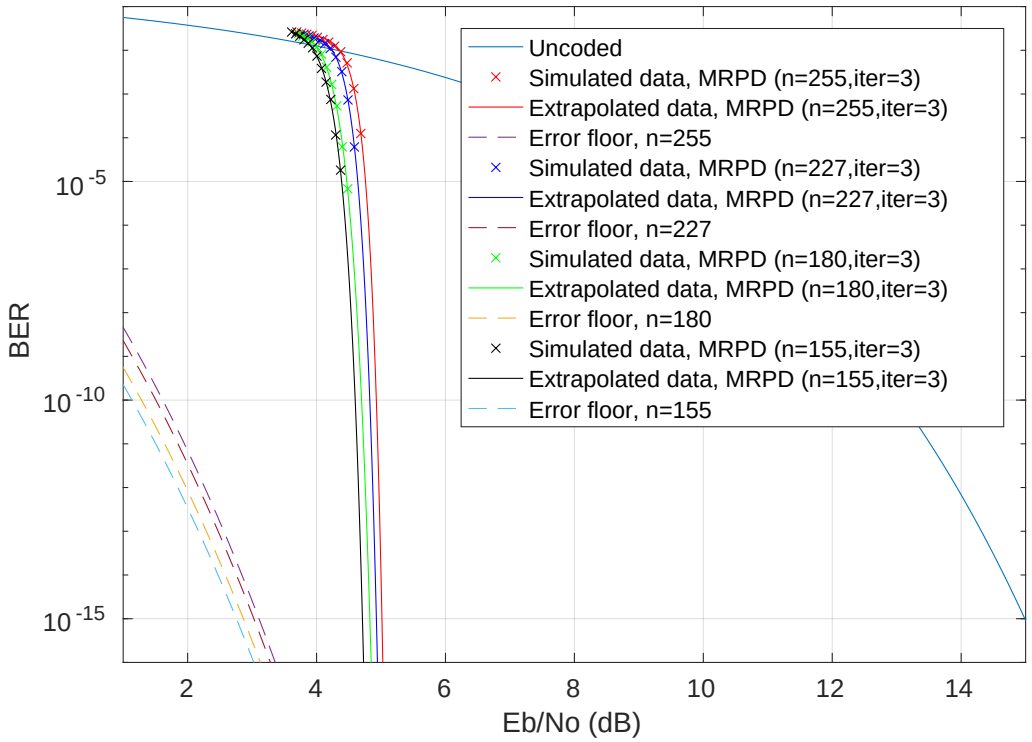


Figure 5.1: BER as a function of E_b/N_0 for iteration = 3.

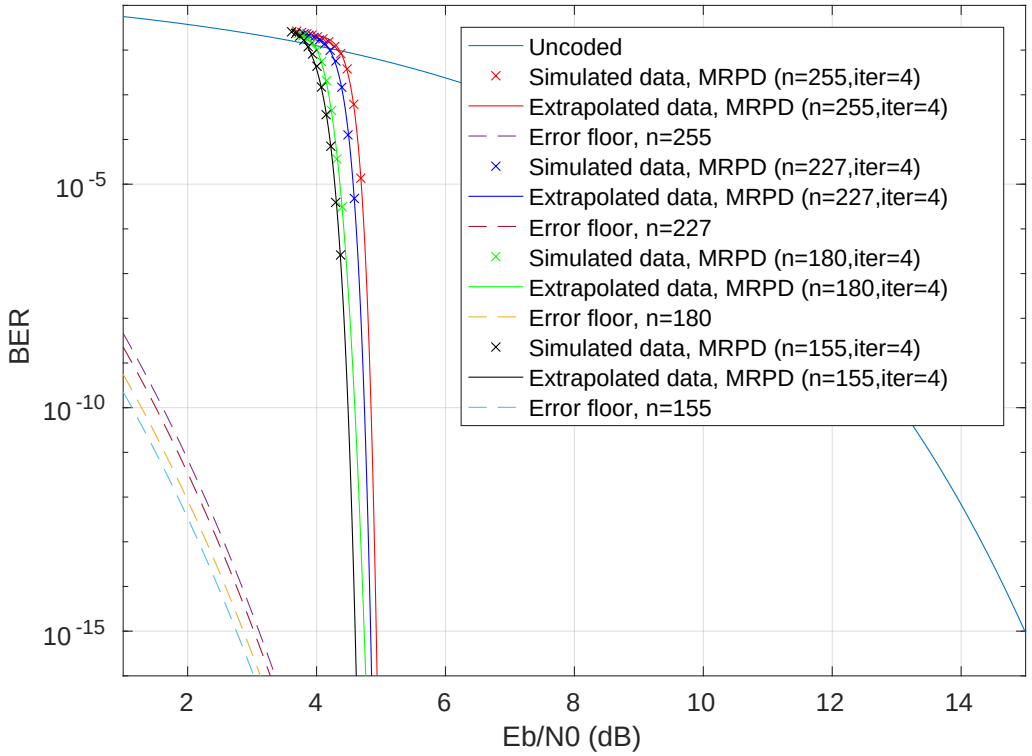


Figure 5.2: BER as a function of E_b/N_0 for iteration = 4.

Tables 5.2-5.4 presents the results of the multi-rate product decoder implementation operated at iteration equal to three, four and five respectively. There are

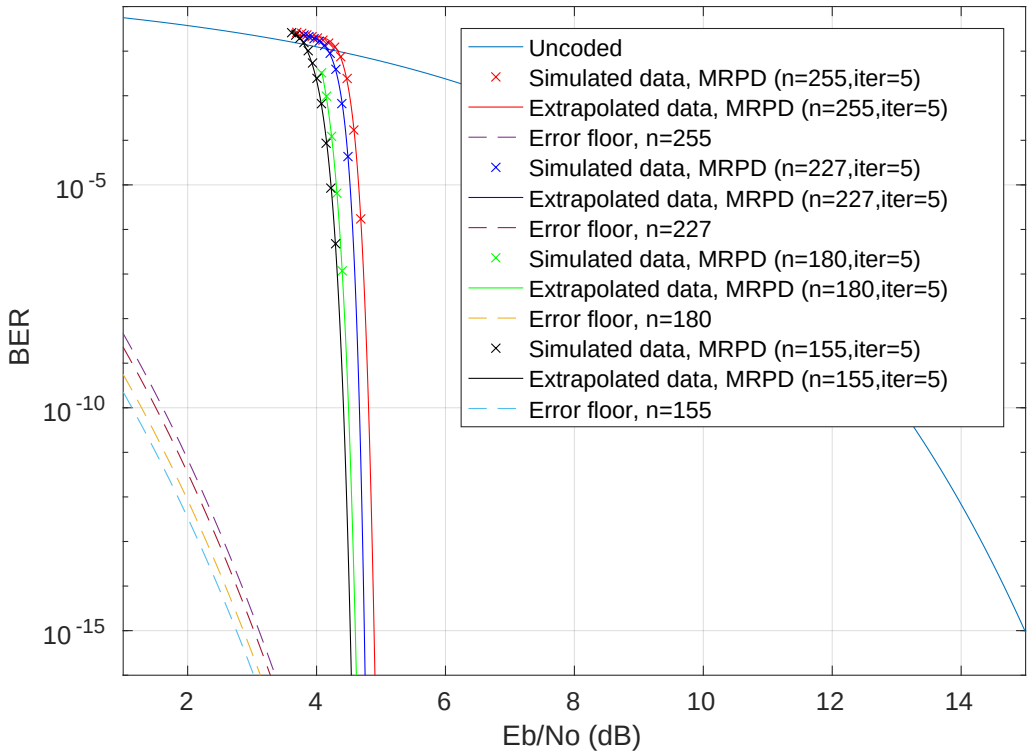


Figure 5.3: BER as a function of E_b/N_0 for iteration = 5.

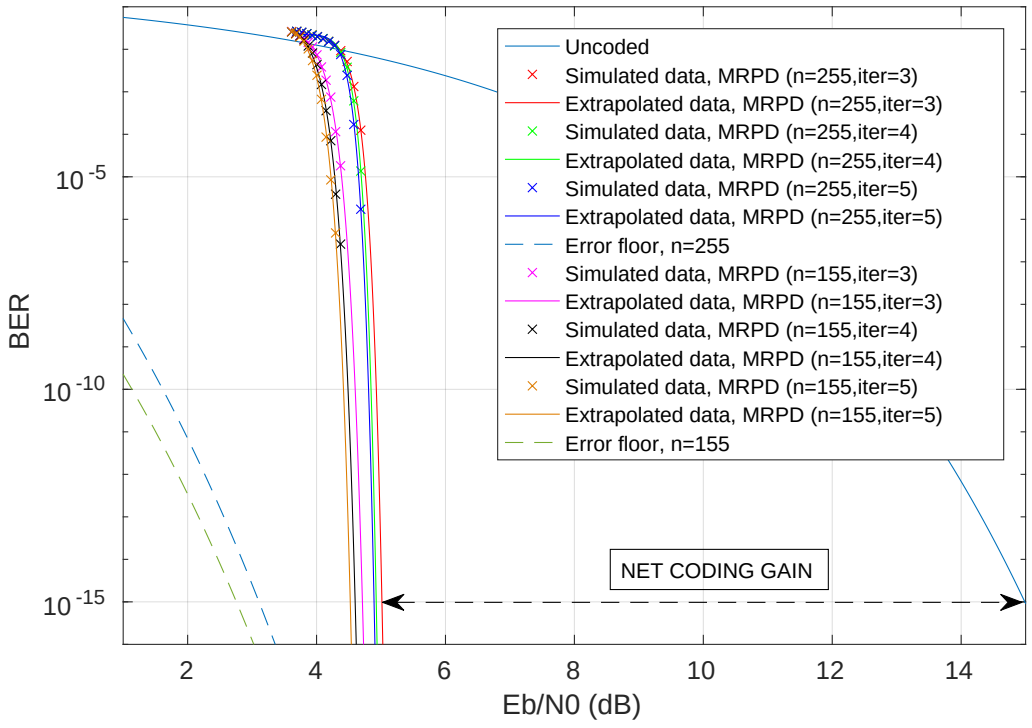


Figure 5.4: BER as a function of E_b/N_0.

implications of a varying code overhead at the circuit level: When implementing a decoder having a mother code on top of which a varying overhead is introduced,

the support for flexible code overheads increases the circuit area [16]. The area of MRPD design increases to 8.78 mm^2 when multi-rate hardware is added to the FRPD1 design which had an area of 6.69 mm^2. So, the area cost for introducing this flexibility is 31 % over the FRPD1 design with the same configuration. Note that this increase in area is for having all the four modes of operation in one design. If fixed designs are used then the number of chips required will be equal to the number of modes of operation and each design has its own area cost. Variation in number of iteration provides a more resource-efficient alternative to obtain variable coding gains at fixed code rates. However, it cannot provide the large range in coding gain obtainable by a multi-rate design. But, a combination of OH and iteration variation can provide several modes with a wider range of operation.

Table 5.2: Evaluation Results of MRPD with iteration = 3

Overhead	**21 %**	**25 %**	**33 %**	**40 %**
Cell area (mm^2)	8.78			
Code rate, R	0.82	0.79	0.75	0.71
Throughput (Gbps)	1628	1257	742	523
Block decoding latency (ns)	32.78	32.78	32.78	32.78
NCG @ BER 10^{-15} (dB)	9.96	10.05	10.16	10.27
Power @ BER 10^{-15} (mW)	940.61	821.75	598.51	523.52
Energy @ BER 10^{-15} (pJ/info. bit)	0.58	0.65	0.81	1.0

Table 5.3: Evaluation Results of MRPD with iteration = 4

Overhead	**21 %**	**25 %**	**33 %**	**40 %**
Code rate, R	0.82	0.79	0.75	0.71
Throughput (Gbps)	1252	967	571	402
Block decoding latency (ns)	42.61	42.61	42.61	42.61
NCG @ BER 10^{-15} (dB)	10.055	10.14	10.24	10.38
Power @ BER 10^{-15} (mW)	829.75	767.59	524.88	460.63
Energy @ BER 10^{-15} (pJ/info. bit)	0.66	0.79	0.92	1.14

Table 5.4: Evaluation Results of MRPD with iteration = 5

Overhead	**21 %**	**25 %**	**33 %**	**40 %**
Code rate, R	0.82	0.79	0.75	0.71
Throughput (Gbps)	1017	785	464	327
Block decoding latency (ns)	52.44	52.44	52.44	52.44
NCG @ BER 10^{-15} (dB)	10.08	10.23	10.35	10.46
Power @ BER 10^{-15} (mW)	769.76	710.14	479.18	422.08
Energy @ BER 10^{-15} (pJ/info. bit)	0.76	0.9	1.03	1.29

Operating clock frequency plays a major role in the selection of number of iterations and OH for generating modes of operation. In our design, with the operating frequency of 610 MHz, we were able to achieve a lowest throughput of 327 Gbps when the product decoder was operated with five iterations and 40 % OH. In order to ramp up the throughput to, for e.g., 400 Gbps, with the same configuration, the operating clock frequency required is 750 MHz. When implemented at such a high clock frequency, however, the cost in terms of area and power consumption increases by 20 %.

The component decoders are implemented as fully block-parallel resulting in high-speed block decoding with latency below 100 ns and estimated throughputs as high as 1.6 Tbps when operating with three iterations and base OH mode. Moreover, owing to the use of clock gating, the design is highly energy-efficient with energy per information bit being below 1.5 pJ/bit for the MRPD design.

Another important aspect to consider is the use of high-speed libraries when synthesizing the design. The results reported above were evaluated with the MRPD design synthesized with a low-power or low-leakage CMOS library. Using a high-speed or low-threshold voltage CMOS library can provide required throughputs at a lower area cost. When implemented with high speed libraries, the MRPD design reduced in area by 10 % but it resulted in a power dissipation increase of 20 % with an increase in leakage power from 1 % of total to 8.7 % of total power. The increase in power dissipation due to leakage power was not sufficient to consider power gating and the increase in total power (dynamic and static) weakened the case of using high speed library in the MRPD design.

5.3 Multi-rate, Multi-t Product Decoder

The MRMTPD design described in section 3.3 was evaluated with $t = 3$ and $t = 4$, decoding iteration of four and at their respective base OHs. The evaluation results are shown in Table 5.5.

Table 5.5: Evaluation Results of MRMTPD with iteration = 4

Error-correction capability	**t = 3**	**t = 4**
Cell area (mm^2)	14.01	
Overhead	21 %	30 %
Throughput (Gbps)	1252	1167
Block decoding latency (ns)	42.61	42.61
NCG @ BER 10^{-15} (dB)	10.055	10.5
Power @ BER 10^{-15} (mW)	1108	1687
Energy @ BER 10^{-15} (pJ/info. bit)	0.88	1.44

The MRMTPD design dissipated 33 % more power than the MRPD with the same configuration with $t = 3$. Ideally, both the designs should have the same power dissipation as during $t = 3$ mode of operation the circuits for $t = 4$ mode are disabled. But the MRMTPD design has an overall increase in area of 60 %

over the MRPD design which increases the power dissipation due to wire load. The MRMTPD design can increase the coding gain range but the increase in area and power consumption could be the limiting factor when using this design. The area of MRMTPD increased by 54 % over the FRPD2 design.

6
Discussion

The primary goal of this thesis was to introduce flexibility in the baseline design keeping the area and power dissipation cost of adding the additional circuit to a minimum. The results obtained from evaluation of our multi-rate product decoder design were promising with a 31 % increase in area and 5 % increase in power dissipation from the fixed-rate product decoder, both designs operated at base OH and number of iterations equal to four. The cost of area and power dissipation of adding reconfigurability was not very high.

It is hard to compare our results on net coding gain with the works presented in section 1.2 as these designs were based on SD decoding which is inherently high performing in terms of net coding gain. However, when we compare the throughput and power dissipation, our design has much better performance owing to the fact that HD decoding is suitable for high-throughput systems. However, the high throughput comes at a cost of a large chip area.

In order for the optical networks to reach very high throughput of 300 Gbps and above, all the modules used in these networks must be able to support the throughput, which is not the case in the present optical network. Our design can be used in the future when the optical networks have the capability to reach such high throughput. In the present system we can trade off the high throughput to reduce the area of the chip further. This can be done by reducing the number of component decoders by sharing component decoders between two or more codewords.

6.1 Limitations

The baseline component decoders used in the MRPD design are highly parallel providing high-throughput operation but at the cost of area. When synthesizing the designs in the Cadence environment, the time required for the synthesis to complete varied based on the size of the chip, with the MRMTPD design taking the maximum time of upto 64 hours for completion. Due to the large synthesis time, analysis of the designs took several hours. Simulations for obtaining BER plots were also time consuming as low post-FEC BER of 10^{-15} were hard to obtain. Emulation of the product decoder on an FPGA can be used to reach such low post-FEC BER, but the product decoders are too large to fit on any FPGA that was available to us. Smaller codes can be used to fit on the FPGA but the net coding gain would be lower for such codes.

6.2 Future Work

In this thesis work, the power analysis was done by simulating the synthesized design in Cadence Genus. Though the power analysis results from these tools are highly reliable, experimental results can authenticate the power results. In the future, the MRPD and MRMTPD designs can be placed and routed and a possible tapeout can be considered. The chip can be used to provide empirical data to corroborate the synthesis results.

A design aspect that was proposed for this thesis work was to use assymetrical product-code memory with component codes of different block lengths. However, when using different block length the finite-field arithmetic used in component decoders also changes which means that different component decoders are required. A solution to this problem would be to use flexible field component decoders and this can be considered for future solution.

7
Conclusion

In this thesis, we have implemented a multi-rate product decoder (MRPD) specifically designed for future rate-adaptable optical communication systems. The decoder is synthesized using 28 nm FD-SOI process technology. The resulting product decoder has a coding gain range of 9.96-10.46 dB. The decoding latency obtained from the evaluation are very low with a maximum of 52.45 ns. The different modes of the product decoder operate at above 300 Gbps throughput with maximum achievable throughput of 1.6 Tbps.

With an efficient clock gating implementation, the design is highly energy-efficient with energy dissipation per information bit from 0.58 to 1.29 pJ/bit. Power dissipation of the different modes are sub 1 W. All of this is achieved with an area cost of 31 % over the fixed-rate product decoder with $t = 3$ design. Keeping in mind that when using fixed-rate product decoders, different chips are required for different modes, the area cost of introducing reconfigurability can be ignored. The MRPD can be used in rate-adaptable optical communication systems to vary the code rate and decoding iterations to achieve higher throughput at the cost of coding gain and vice versa. Also, increasing the OH decreases the power consumption and could be used during periods of low load or traffic.

Finally, the multi-rate, multi-t product decoder design (MRMTPD) implemented can deliver a much higher coding gain range but area of the chip can be a limiting factor. But with the same reasoning as above, the area cost can be overcome by the advantages of having such a multi-rate, multi-t design providing a wide range of coding gain and throughput.

Bibliography

[1] P. J. Winzer, "Scaling optical fiber networks: Challenges and solutions," *Optics and Photonics News*, vol. 26, no. 3, pp. 44–51, 2015.

[2] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A survey on FEC codes for 100 G and beyond optical networks," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 209–221, Firstquarter 2016.

[3] Z. Wang, "Super-FEC codes for 40/100 Gbps networking," *IEEE Communications Letters*, vol. 16, pp. 2056–2059, December 2012.

[4] L. Song, M.-L. Yu, and M. S. Shaffer, "10- and 40-Gb/s forward error correction devices for optical communications," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1565–1573, Nov 2002.

[5] A. Tychopoulos, O. Koufopaulou, and I. Tomkos, "FEC in optical communications - A tutorial overview on the evolution of architectures and the future prospects of outband and inband FEC for optical communications," *IEEE Circuits and Devices Magazine*, vol. 22, no. 6, pp. 79–86, 2006.

[6] C. Condo, P. Giard, F. Leduc-Primeau, G. Sarkis, and W. J. Gross, "A 9.52 dB NCG FEC scheme and 162 b/cycle low-complexity product decoder architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. PP, no. 99, 2017.

[7] B. Li, K. J. Larsen, D. Zibar, and I. T. Monroy, "Over 10 dB net coding gain based on 20% overhead hard decision forward error correction in 100G optical communication systems," in *Eur. Conf. Opt. Commun. (ECOC)*, Sept. 2011.

[8] A. F. Molisch, *Wireless communications*. Chichester: Wiley, 2005.

[9] O. Gerstel, M. Jinno, A. Lord, and S. J. B. Yoo, "Elastic optical networking: a new dawn for the optical layer?," *IEEE Commun. Mag.*, vol. 50, pp. s12–s20, Feb. 2012.

[10] J. G. Proakis and M. Salehi, *Digital communications*. New York: McGraw-Hill Higher Education, 5 ed., 2008.

[11] D. Zou and I. B. Djordjevic, "FPGA-based rate-adaptive LDPC-coded modulation for the next generation of optical communication systems," *Opt. Express*, vol. 24, pp. 21159–21166, Sept. 2016.

[12] K. Sugihara, S. Kametani, K. Kubo, T. Sugihara, and W. Matsumoto, "A practicable rate-adaptive FEC scheme flexible about capacity and distance in optical transport networks," in *Opt. Fiber Commun. Conf. (OFC)*, Mar. 2016.

[13] G. H. Gho and J. M. Kahn, "Rate-adaptive modulation and coding for optical fiber transmission systems," *IEEE J. Lightw. Technol.*, vol. 30, pp. 1818–1828, June 2012.

[14] A. Rasmussen, M. P. Yankov, M. S. Berger, K. J. Larsen, and S. Ruepp, "Improved energy efficiency for optical transport networks by elastic forward error correction," *IEEE J. Opt. Commun. Netw.*, vol. 6, pp. 397–407, Apr. 2014.

[15] C. Fougstedt and P. Larsson-Edefors, "Energy-efficient high-throughput staircase decoders," in *Opt. Fiber Commun. Conf. (OFC)*, p. Tu3C.6, Optical Society of America, 2018.

[16] D. A. Morero, M. A. Castrillón, A. Aguirre, M. R. Hueda, and O. E. Agazzi, "Design tradeoffs and challenges in practical coherent optical transceiver implementations," *IEEE J. Lightw. Technol.*, vol. 34, pp. 121–136, Jan. 2016.

[17] W. E. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge;New York;: Cambridge University Press, 2009.

[18] T. J. Richardson and R. L. Urbanke, *Modern coding theory*. Cambridge;New York;: Cambridge University Press, 1 ed., 2008.

[19] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[20] R. M. Roth, *Introduction to coding theory*. Cambridge: Cambridge university press, 2007.

[21] J. Castiñeira Moreira and P. G. Farrell, *Essentials of error-control coding*. West Sussex, England: John Wiley & Sons, 2006.

[22] R. H. Morelos-Zaragoza, *The Art of error correcting coding*. Chichester: Wiley, 2. ed., 2006.

[23] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, no. 1, pp. 68–79, 1960.

[24] W. Peterson, "Encoding and error-correction procedures for the bose-chaudhuri codes," *IRE Transactions on Information Theory*, vol. 6, pp. 459–470, September 1960.

[25] Y. Jiang, *A practical guide to error-control coding using MATLAB*. Boston: Artech House, 2010.

[26] S. An, H. Tang, and J. Park, "A inversion-less Peterson algorithm based shared KES architecture for concatenated BCH decoder," in *Int. SoC Design Conf. (ISOCC)*, pp. 281–282, Nov. 2015.

[27] R. Chien, "Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 357–363, 1964.

[28] H. Helgert and R. Stinaff, "Shortened BCH codes (Corresp.)," *IEEE Transactions on Information Theory*, vol. 19, pp. 818–820, Nov 1973.

[29] P. Elias, "Error-free coding," *IRE Trans. Inf. Theory*, vol. 4, pp. 29–37, Sept. 1954.

[30] S. Lin and D. J. Costello, *Error control coding*, vol. 2. Prentice Hall Englewood Cliffs, 2004.

[31] G. Forney, *Concatenated Codes*. M.I.T. Press research monographs, M.I.T. Press, 1966.

[32] J. H. v. Lint, *Introduction to coding theory*, vol. 86. Berlin: Springer, 3. rev. and expand ed., 1999.

[33] A. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over GF(2m)," *IEEE Transactions on Computers*, vol. 53, pp. 945–959, Aug 2004.

[34] B. Li, K. J. Larsen, D. Zibar, and I. T. Monroy, "Over 10 dB net coding gain based on 20% overhead hard decision forward error correction in 100G optical communication systems," in *Eur. Conf. Opt. Commun. (ECOC)*, p. Tu.6.A.3, Sept. 2011.

[35] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *IEEE Workshop on Signal Processing Systems*, pp. 303–308, Oct. 2006.

[36] M. Yin, M. Xie, and B. Yi, "Optimized algorithms for binary BCH codes," in *IEEE Int. Symp. on Circuits and Systems*, pp. 1552–1555, May 2013.

[37] S. An, H. Tang, and J. Park, "A inversion-less Peterson algorithm based shared kes architecture for concatenated BCH decoder," in *2015 International SoC Design Conference (ISOCC)*, pp. 281–282, Nov 2015.