Performance analysis of error detection and correction code for wireless sensor networks

Gopinath Balakrishnan

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation

https://digitalscholarship.unlv.edu/rtds/1865

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
PERFORMANCE ANALYSIS OF ERROR DETECTION AND CORRECTION CODE

FOR WIRELESS SENSOR NETWORKS

by

Gopinath Balakrishnan

Bachelor of Science in Electronics and Communication Engineering
University of Madras, India
June 2002

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Electrical Engineering
Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2005
Thesis Approval
The Graduate College
University of Nevada, Las Vegas

December 12, 2005

The Thesis prepared by

Gopinath Balakrishnan

Entitled

"Performance Analysis of Error Detection and Correction
Codes For Wireless Sensor Networks"

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Performance Analysis of Error Detection and Correction Codes for Wireless Sensor Networks

by

Gopinath Balakrishnan

Dr. Mei Yang, Examination Committee Chair
Assistant Professor in Department of Electrical and Computer Engineering
University of Nevada, Las Vegas

Recent advances in wireless communications and electronics have enabled the development of low-cost, low-power, self-organizational, multifunctional wireless sensor networks. Wireless sensor networks can be applied to a wide range of application areas including health, military and homeland security, environment, industry and commercial, and home. A typical wireless sensor network consists of one or more sink nodes and a large number of sensor nodes scattered in a sensor field. Each of these sensor nodes is capable to collect the data and relay the data back to the sink through a multi-hop architecture. The key challenge in sensor networks is to overcome the energy constraint since each sensor node has limited power. Hence, it is important to minimize the energy used to transmit packets over wireless links.

The data transmitted from the sensor nodes are vulnerable to be corrupted by errors induced by noisy channels and other factors. Hence it is necessary to provide a proper error control scheme to reduce the bit error rate (BER) to a desired level without sacrificing other performance. Energy required for error control code has a direct impact
on battery power consumption. Since high error rates are inevitable in the wireless environment, energy efficient error detection and correction scheme is vital in wireless sensor networks. However, in the literature, limited work has been focused on the study of energy efficient error control scheme.

This thesis is focused on energy-efficient error detection and correction schemes for wireless sensor networks. A complete study of energy consumption and error control performance of various error detection and correction codes is conducted. In detail, error control codes with different constraints are implemented and simulated using hardware description language. Implementation on Application Specific Integrated Circuits (ASIC) design and Field Programmable Gate Array (FPGA) is carried out and the energy consumption is measured. The error control performance of these codes is evaluated in terms of Bit Error Rate (BER) by transmitting randomly generated data through a Gaussian channel. Based on the study and comparison of the three different error control codes, we identify that binary-BCH codes with ASIC implementation are best suitable for wireless sensor networks.
# TABLE OF CONTENTS

ABSTRACT ................................................................................................................................. iii

LIST OF FIGURES ...................................................................................................................... vi

LIST OF TABLES ......................................................................................................................... vii

ACKNOWLEDGEMENTS ............................................................................................................. viii

CHAPTER 1 INTRODUCTION ..................................................................................................... 1
  1.1 Wireless Sensor Networks ................................................................................................. 2
  1.2 Components of Sensor Nodes .......................................................................................... 3
  1.3 Application of Sensor Networks ....................................................................................... 5
  1.4 Protocol Stack of Wireless Sensor Network ..................................................................... 6
  1.5 Difference between Sensor Networks and Ad-Hoc Networks ....................................... 8
  1.6 Challenges in Wireless Sensor Networks ....................................................................... 10
  1.7 Research Interests ............................................................................................................ 12
  1.8 Contribution and Overview of the Thesis ....................................................................... 13

CHAPTER 2 BACKGROUND ....................................................................................................... 15
  2.1 Types of Error Control Codes ......................................................................................... 17
  2.2 Binary Primitive BCH Codes ......................................................................................... 20
  2.3 Reed-Solomon Codes ..................................................................................................... 28
  2.4 Convolutional Codes ..................................................................................................... 31
  2.5 Related Work .................................................................................................................. 35
  2.6 Summary ......................................................................................................................... 36

CHAPTER 3 METHODOLOGY ................................................................................................... 37
  3.1 Why BCH, RS and Convolutional Codes? ..................................................................... 38
  3.2 Implementation of Codes ............................................................................................... 38
  3.3 Performance Measure .................................................................................................... 40
  3.4 Power Estimation in FPGA Design ................................................................................ 43
  3.5 Power Estimation in ASIC Design .................................................................................. 46
  3.6 Summary ......................................................................................................................... 47

CHAPTER 4 RESULTS AND DISCUSSION ............................................................................... 48
  4.1 Code Performance ......................................................................................................... 48
    4.1.1 BCH Codes ............................................................................................................... 49
    4.1.2 RS Codes ................................................................................................................... 51
    4.1.3 Convolutional Codes ............................................................................................... 55

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
4.2 Power Estimation on FPGA Design ................................................................. 57
4.3 Power Estimation on ASIC Design ................................................................. 60
4.4 Power Analysis in Sensor Nodes ................................................................. 62
4.5 Discussion ................................................................................................. 65

CHAPTER 5 CONCLUSIONS ............................................................................ 66
5.1 Future Work ............................................................................................. 67

BIBLIOGRAPHY ............................................................................................... 68

VITA ................................................................................................................. 75
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Wireless Sensor Network</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Block Diagram of Wireless Sensor Node</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>The Sensor Networks Protocol Stack</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Digital Communication Systems</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Types of Error Control Schemes</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>The BCH encoder</td>
<td>22</td>
</tr>
<tr>
<td>2.4</td>
<td>Block Diagram of a BCH Decoder</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>The Circuit for Computing Syndrome</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Second Method for Computing $S_3$, $m=4$</td>
<td>25</td>
</tr>
<tr>
<td>2.7</td>
<td>Chien's Search Circuit</td>
<td>28</td>
</tr>
<tr>
<td>2.8</td>
<td>Block Diagram of RS Decoder</td>
<td>31</td>
</tr>
<tr>
<td>2.9</td>
<td>Convolutional Encoder of Rate $\frac{1}{2}$ and $m=3$</td>
<td>32</td>
</tr>
<tr>
<td>2.10</td>
<td>Block Diagram of Viterbi Decoder</td>
<td>34</td>
</tr>
<tr>
<td>2.11</td>
<td>Trellis Diagram for Code Rate $\frac{1}{2}$</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Design Flow in XPower</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>Design Flow in Design Compiler</td>
<td>46</td>
</tr>
<tr>
<td>4.1</td>
<td>BER Performance of (31,11,5), (31,16,3), (31,21,2) and (31,26,1) Binary-BCH Codes</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>BER Performance of (31,11,5) and (15,11,1) Binary-BCH Codes</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>BER Performance of (31,11,10), (31,15,8), (31,21,5) and (31,25,3) RS Codes</td>
<td>52</td>
</tr>
<tr>
<td>4.4</td>
<td>BER Performance of (31,11,10) and (15,11,2) RS Codes</td>
<td>53</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison of RS and BCH Codes</td>
<td>54</td>
</tr>
<tr>
<td>4.6</td>
<td>BER Performance of (2,1,3), (2,1,4), (2,1,5) and (2,1,6) Convolutional Codes</td>
<td>55</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison of Convolutional and BCH Codes</td>
<td>56</td>
</tr>
<tr>
<td>4.8</td>
<td>Complexity Plots</td>
<td>58</td>
</tr>
<tr>
<td>4.9</td>
<td>Power Consumption in FPGA Design</td>
<td>60</td>
</tr>
<tr>
<td>4.10</td>
<td>Power Consumption in ASIC Design</td>
<td>62</td>
</tr>
<tr>
<td>4.11</td>
<td>Power Consumption in Sensor Node for Coded and Uncoded Channel</td>
<td>64</td>
</tr>
<tr>
<td>4.12</td>
<td>Power Consumption in Transceiver and Error Control Unit</td>
<td>64</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to acknowledge the immense assistance and moral support provided by my advisor, Dr. Mei Yang during the course of my masters program at the University of Nevada, Las Vegas. The guidance provided by her in steering this research project from concept to completion has been invaluable.

I would like to thank Dr. Yingtao Jiang, Dr. Muthukumar Venkatesan and Dr. Yoohwan Kin for their direct and indirect support throughout this investigation.

It is important to mention the moral support of my immediate family including my parents, Mrs. Buvaneswari and Mr. Balakrishnan, my brother Chandramouli, my aunt and uncle Mrs. Devaki and Mr. Neelakrishnan and my cousins Shankar and Sangeetha. They were available for me at all times. But for their constant motivation, it would have been impossible for me to get this far in life.

I would like to acknowledge the help of my friends Vikram Marthandam and Utthaman Thirunavukkarasu, whose help and advice has always been an invaluable source of motivation for me.

Finally I would like to thank the Department of Electrical and Computer Engineering at the University of Nevada, Las Vegas for giving me an opportunity to pursue my masters’ degree.
CHAPTER 1

INTRODUCTION

Recent advances in wireless communications and electronics have facilitated the development of low-cost, low-power, small-size, and multifunctional wireless sensor nodes that communicate in short distances [1]. These miniature sensor nodes, which consist of sensing, data processing, and communicating components, have led to the emergence and deployment of wireless sensor networks. The low-cost, rapid deployment, ability of self-organization and cooperative data-processing, have made wireless sensor networks a practical solution for a wide range of application areas, including military and homeland security, health, environment, industry and commercial, and home.

The most significant challenge in sensor networks is to overcome the energy constraints since each sensor node has limited energy to consume. This thesis is focused on the study of energy-efficient error control schemes for wireless sensor networks. This chapter introduces the background of this study and gives the outline of the thesis. First, an introduction of wireless sensor networks and the components in the sensor nodes is given. Next, the applications of wireless sensor networks are addressed. After that, we discuss the protocol stack for wireless sensor networks followed by the discussion of the difference between the traditional wireless networks and wireless sensor networks. Then,

1
challenges in the design of wireless sensor networks are addresses. At the end of this chapter, we discuss the motivation for this study followed by an outline of the thesis.

1.1 Wireless Sensor Networks

A wireless sensor network is composed of a large number of sensor nodes that are randomly scattered in the sensor field as shown in the Figure 1.1 [1]. These sensor nodes are connected to each other through wireless medium, typically radio frequency (RF) channels. These sensor nodes are capable of sensing, processing and relaying the sensed data within their peers or directly to the sink node. The relay path from a sensor node to the sink node can be either single-hop or multi-hop. A multi-hop path consists of multiple forwarding sensor nodes between the source and destination nodes. All the data gathered by the sensors are collected by the sink nodes. A wireless sensor network may contain one or more sink nodes. The sink node accesses the task manager node through the internet or satellite.

![Figure 1.1 Wireless Sensor Network](image-url)
A typical application of wireless sensor networks is surveillance and monitoring where a large number of miniature sensor nodes are densely deployed in the vicinity of the phenomenon. To allow random deployment in inaccessible terrains or disaster relief operations [1], the position of sensor nodes should not to be predetermined. This requires the sensor networks are self organizing in nature. The other important feature is the cooperative effort of the sensor nodes. Rather than transmitting raw sensed data to nodes responsible for data aggregation, the sensor nodes can carry out simple processing and transmit the partially processed data required by other nodes. The low-cost of wireless sensor nodes allows for dense deployment of sensor nodes, which aids in sensing accurate data over a large geographical area. These unique features of wireless sensor networks make it possible to build large-scale, maintenance-free and fault-tolerant networks.

1.2 Components of Sensor Node

As shown in Figure 1.2, a sensor node integrates a power unit, a sensing unit, a processing unit, and a transceiver unit. Some application-specific components such as a position finding system, a mobilizer, and power scavenging units may be included. The heart of the sensor node is the sensing unit which commonly incorporates sensors and analog-to-digital converters (ADCs). The analog signals produced by the sensor in response to the environmental changes are converted to digital signals by the ADCs, which are processed by the processing unit. The processing unit, which generally has a small storage unit for temporary data storage, performs data processing data and task management. The transceiver unit transmits the processed data to the other sensor nodes.
through the wireless medium. The power unit is one of the most important components.

In some applications, once the sensor nodes are deployed in the sensor field, it is difficult to recharge or replace the battery. To keep the sensor node running over the period of time, power units may be supported by power scavenging units like solar cells.

Some applications require accurate position information of the sensor nodes. For these applications, position finding system is included in the sensor node. Mobilizer is added in applications that demand the movement of the sensor nodes with respect to each other or with the sink.

Contemporary research strives in sizing the sensor nodes as small as possible with memory scale in several mega bytes. It is projected to have size [20] of 1 cm$^3$ in its volume and 100 grams in weight with current sensors [2] having a size of 70 cm$^3$ in its
volume with 128 KB of instruction memory, 4 KB of data RAM, and 512 KB of flash memory. The processor is operated at 4 MHz.

1.3 Applications of Sensor Networks

The low-cost, rapid deployment, ability of self-organization and cooperative data-processing, have made wireless sensor networks a practical solution for a wide range of application areas, including military and homeland security, health, environment, industry and commercial, and home.

- Military applications: target tracking and directing, equipment and ammunition, battlefield surveillance and monitoring; attack detection, and damage assessment. In equipment and ammunition the commanders can constantly monitor the status of friendly troops, the condition and the availability of the equipment and the ammunition in a battlefield by the use of sensor networks.

- Environment applications: habitat monitoring, chemical and biological detection, intelligent agriculture, monitoring of marine soil and atmosphere, forest fire/flood detection, pollution detection, etc. [8][9][10][11][12][13][14].

- Health applications: assistance for disabled people, integrated patient monitoring, diagnostics, drug administration in hospitals, tracking and monitoring doctors and patients, etc.[1][11][24][25][26].

- Home applications: control of house-hold appliances [15], “universal” remote control, intelligent home, toys, etc.
• Industry/commercial applications: monitoring of heating, ventilating, and air conditioning (HVAC), monitoring disaster area; robot control and guidance, factory process control and automation; automation, security services, utility metering, interactive toys/museums; smart devices; machine diagnosis; vehicle tracking and detection, etc. [7][8][11][16][17][18][19][20][21][22][23][24].

1.4 Protocol Stack of Wireless Sensor Network

Figure 1.3 shows the protocol stack used by sink and all sensor nodes. The protocol stack consists of the application layer, transport layer, network layer, data link layer, physical layer, the power management plane, mobility management plane, and task management plane [1]. In supplement to the conventional networking protocol [4], this protocol stack merges power and routing awareness, integrates data with networking protocols, communicates power efficiently through wireless medium and promotes cooperative effort of sensor nodes.

The physical layer deals directly with the characteristics of the wireless media used to transmit the data. It is in charge of frequency selection, carrier frequency generation, signal detection, modulation, demodulation and data encryption.

The data link layer deals with medium access. It is responsible for multiplexing of data streams, data frame detection, medium access control (MAC), and error control between two entities. It ensures reliable point-to-point and point-to-multipoint communications in wireless sensor networks.
The network layer handles the routing of data from the sensor nodes to the sink node. In multi-hop networks, the network layer decides the path for the data packets to be transmitted, which ensures that data are sent towards the destination node.

![Sensor Networks Protocol Stack](image)

Figure 1.3 The Sensor Networks Protocol Stack

The transport layer manages the flow of data. It directs each data packet to arrive at the destination correctly. This layer is of interest only when the sensor nodes in the sensor field as shown in Figure 1.1 needs to access other outside networks.

The application layer handles the service modeling, interest advertisement, sensor management, and other functions. Different application-specific software can be built and used in this layer.

In addition to these layers, the three planes help the sensor nodes to coordinate the sensing task and work efficiently. The power management plane minimizes the power consumption and may turn off the receiver when required. The mobility management
plane manages the movement of the sensor nodes. The task management plane activates the nodes for sensing.

1.5 Difference between Sensor Networks and Ad Hoc Networks

Wireless sensor networks are different from the conventional ad hoc networks in the following ways. The sensor nodes are densely deployed in hundreds or thousands of nodes. The sensor networks are randomly deployed and its topology changes frequently. Sensor nodes use a broadcast communication paradigm while most ad hoc networks use point-to-point communications. The sensor nodes have limited power, computational capacities, bandwidth and storage compared to a node in the ad hoc network. Hence the protocols designed for ad hoc networks are not good enough to tackle the unique resource constraints of sensor networks.

In traditional wireless networks the propagation and fading effects in point-to-point long distance communication are minimized at the cost of increased signal power and circuit complexity. However, sensor nodes have limited power and the power may not be replenished. Hence, in designing physical layer protocol for wireless sensor networks, energy minimization techniques have considerable importance than minimizing the propagation and fading effects. In terms of modulation scheme, binary modulation is more power efficient than M-ary modulation schemes [5].

The media access control (MAC) protocols used for traditional ad hoc network are unsuitable for sensor networks. In traditional wireless network, the mobile nodes communicate to the base node using single-hop communication. The MAC protocol of such system is designed to provide high-quality service and efficient bandwidth usage.
Power consumption is the second concern as the base nodes are wired to power supply and the mobile nodes have renewable batteries. In wireless sensor network, energy efficiency MAC protocols are the first objective. The second goal is to efficiently share communication resources among the sensor nodes. In addition, the MAC protocols designed for sensor network must also incorporate features needed for multiple-hop communications and provide self organizing capability.

Since energy is the most important design constraint, the sensor network must include power saving modes. Power saving modes in ad hoc networks saves significant amount of power by switching the transceiver off when it is idle. As turning on the transceiver consumes extra power, blindly turning off/off the sensor node ends up consuming more power to start the sensor than that required to keep the sensor node running. Therefore, power saving modes in wireless sensor nodes is energy efficient only when the time spent in idle state is more than certain threshold. A significant amount of power consumed in for error control coding [3]. An energy-efficient error control scheme is necessary to keep the power consumption low.

The network layer protocols of ad hoc networks are not suitable for the sensor networks. The network layer protocol of sensor network should be designed with energy efficiency as priority. In addition, the following principles should be followed its design: 1) sensor network are mostly data-centric, 2) data aggregation is important only when it does not hinder the collaborative effort of sensor networks, and 3) the ideal sensor network has attribute-based addressing and location awareness.

The transport layer of traditional ad hoc networks uses Transmission Control Protocol (TCP) for communication between different networks. In applications using sensor
networks the TCP splitting may be required for the network to communicate with other networks such as Internet. Usually the communication between user and the sink node is accomplished using TCP or User Datagram Protocol (UDP) and the link between sink node and sensor nodes are achieved using UDP. Unlike the TCP, UDP adds no reliability, flow-control, or error-recovery functions to IP. However, due to its simplicity, UDP consumes consume less network overhead and power than TCP. In addition, TCP requires global addressing which is not the case in sensor networks where attribute-based addressing is considered.

1.6 Challenges in Wireless Sensor Networks

Some of the factors that influence the design of the wireless sensor networks are fault tolerance, scalability, production costs, hardware constraints, sensor network topology, transmission media and power consumption.

- Fault Tolerance: Sensor nodes deployed randomly are prone to failures and may not have a replacement. This failure may be due to lack of power, physical damage or natural obstacles. The reliability or fault tolerance of the sensor network is the ability to continue operation in the event of sensor node failure [1][2].

- Scalability: The advantage of sensor nodes being tiny makes it possible for densely deployed sensor networks. The number of nodes deployed may be in the order of hundreds with some applications demanding millions of them. Scalability refers the capability of increasing the number of sensors within an area without
disturbing the operation of the existing network. The newly designed algorithm should be able to work with this number of nodes.

- **Product Costs:** The random deployment of sensor networks is cost effective only when the cost of deployment of traditional network is more than the cost of implementing a sensor network. Since the sensor networks are densely deployed, the cost of each sensor node has direct impact on the overall cost of the sensor network. Hence the production cost of sensor nodes should be kept as low as possible.

- **Hardware Constraints:** As shown in Figure 1.2, the sensor nodes have 4 basic components with some additional application specific components. These subunits must fit into a compact structure which is small in size [27]. In addition to size limit, these sensor nodes have a stringent power constraint [18]. Hence, the design of hardware for sensor nodes must be optimized for power and size.

- **Sensor Network Topology:** Sensor nodes use wireless media to communicate among them and with the sink as shown in Figure 1.1. These links may be radio, infrared or optical media. Sensor networks can be made global by choosing the wireless media that is available worldwide.

- **Transmission Media and Power Consumption:** The sensor nodes have very limited power source and it is impossible to replenish power in most of the applications. Hence the sensor node lifetime is limited by the battery lifetime. As such, power conservation and power management take an additional importance. It is for this reason that most research work for sensor networks is focused on energy-aware protocols and algorithms.
1.7 Research Interest

As discussed in Section 1.4 the protocols developed for ad hoc networks are not suitable for wireless sensor network. The characteristics and the requirements of the applications bring the following challenges to the design of wireless sensor networks.

- Simple, low power modulation scheme need to be developed.
- Minimization of signal propagation and fading effects need to be addressed. Design of tiny, inexpensive, low power sensor node with power efficient hardware management techniques is essential. Some techniques are managing the operating frequency, minimizing switching power and predicting work load in the processors.
- Media access control (MAC) must be designed for mobile sensor networks. This design should be capable of communication data in the sensor network when both the sensor node and the target node are mobile.
- The life time of sensor networks can be prolonged by using the power efficiently with the aid of power saving mode. For this efficient use of power, the minimum idle time for which the node can enter reduced activity mode needs to be calibrated.
- Existing network layer protocols for communicating between the sensor nodes should be improved or new protocols that can address high topology changes and high scalability should be developed. These protocols should also address the multiple hop communication system used in sensor networks.
- In addition to the power constraint the sensor nodes have very less memory and processing capabilities inbuilt on them. Hence the sensor nodes cannot store large
amount of data and the acknowledgements for received data would be expensive. Therefore new schemes that split the end to end communication at the sink node are needed. The communication between internet and the sink node can use TCP/UDP whereas the communication between sink and the sensor nodes uses UDP.

- Sensor management protocol performs administrative tasks such as moving sensor nodes and time synchronization of nodes. Research developments should also focus on task assignment and data advertisement protocol and sensor query and data dissemination protocol.

- Error control is extremely important for reliable transmission of data. Convolutional coding effects have been considered in [15], whereas the complexity and power consumption of linear block codes need to be addressed.

1.8 Contribution and Overview of the Thesis

This thesis is focused on the study of energy-efficient error control schemes for wireless sensor networks. As discussed earlier the data link layer is responsible for reliable data transmission. Since data transmitted over the wireless media is vulnerable to be corrupted by noise, error control schemes are necessary to keep the bit error rate (BER) low. As discussed in previous sections, due to the stringent energy constraint, it is impossible to increase the signal power of the transmitted signal in wireless sensor networks. Hence an alternative way is to use the error control codes to reduce the BER. The encoding and decoding circuitry in error control codes may consume a sizable
amount of power. This motivates us to study energy-efficient error detection/correction codes.

The rest of the thesis is organized as follows. Chapter 2 gives an overview of different types of error control codes and reviews previous work done in the scope of the thesis. Chapter 3 elaborately explains the constraints used in designing the error control codes and the methodology followed in determining the energy-efficient error control scheme. Chapter 4 discusses and compares the power consumption of various kinds of error correction codes, and justifies the energy-efficient error control code with the results obtained. Chapter 5 concludes the thesis with suggestions for future work.
CHAPTER 2

BACKGROUND

In recent years, there has been a massive growth in wireless communications especially in the field of wireless networking. Figure 2.1 shows the block diagram of digital communications/storage system. In these communication systems, the information is represented as a sequence of binary bits. The binary bits are then modulated and transmitted as analog signal waveforms over a communication channel. The transmitted signals are corrupted by the noise and interference in the communication channel. At the receiver, the demodulator maps the channel corrupted transmitted signal back to binary bits. The received binary information is an estimate of the transmitted binary information. Bit errors may result due to the transmission and the number of bit errors depends on the amount of noise and interference in the communication channel.

Error control coding is often included for reliable digital communication systems. Different types of error control schemes are available to protect the digital information from noise and interference and reduce the number of bit errors. Channel coding is mostly accomplished by selectively introducing redundant bits into the transmitted information stream. These additional bits will allow detection and correction of bit errors in the received data stream and provide more reliable information transmission. The cost of using channel coding to protect the information is a reduction in data rate or an expansion in bandwidth.
Error control coding is achieved with the aid of an encoder in the transmitter and decoder in the receiver. The encoder converts the information sequence $u$ into a discrete encoded sequence $v$ called a codeword. In most of the applications the codeword is also a binary sequence while some applications non-binary codeword have been used. The decoder transforms the received sequence $r$ into a binary sequence $u'$ called the estimated information sequence.

Research over the past decade resulted in design and implementation of different types of error control scheme which minimizes the probability of error. Wireless sensor network are mostly run by batteries have stern constraint on power. Hence the error control scheme for reliable transmission over the noisy channel must be power efficient. The following sections talks about the different types of error control codes and concludes by reviewing the earlier work on power efficient error control.
2.1 Types of Error Control Codes

The error control codes are categorized into two major types the automatic repeat request scheme (ARQ) and forward error correction scheme (FEC) as shown in Figure 1.2. ARQ is a protocol with good error-detecting capability. When the receiver detects an error in the received vector, it automatically requests the transmitter to resend the vector. This process is repeated until the received vector is error free or the error continues beyond a predetermined number of transmissions. ARQ is sometimes used with Global System for Mobile (GSM) communication to guarantee data integrity. In the FEC system a good error-correcting code is used. When the receiver detects an error in the received vector, it attempts to determine the error locations and then corrects the errors. If the receiver fails to locate the exact errors, the received vector would be decoded incorrectly and the erroneous message will be delivered to the user.

The ARQ and FEC have their own advantages and disadvantages. The advantage of ARQ over FEC is that error detection requires less complex circuitry than error correction. In addition, ARQ are adaptive for the reason that the information is retransmitted only when errors occur. In contrast, FEC is extremely good when the channel error rate is high while in ARQ the data must be retransmitted frequently and the rate at which newly generated data are correctly received is lowered.

The ARQ systems are of two types, stop-and-wait ARQ and continuous ARQ. In stop and wait ARQ, the transmitter sends the codeword and waits for one of the two acknowledgements. If the positive (ACK) acknowledgement is received then the transmitter sends the next codeword or if the negative (NAK) acknowledgement is received the transmitter resends the previous codeword. The continuous ARQ are further
categorized into two types go-back-N ARQ and selective-repeat ARQ. In go-back-N ARQ, the transmitter continuously sends the codewords until it receives NAK. Once it receives NAK, the transmitter resends all the N codewords starting from the erroneous codeword. In selective-repeat ARQ, the transmitter continuously sends the codewords and resends only those that are negatively acknowledged.

![Diagram of Error Control Schemes](image)

Figure 2.2 Types of Error Control Schemes

The FEC are further classified into two major types of channel codes, namely linear block codes [28] and convolutional codes. There are many differences between block codes and convolutional codes. Block codes are based rigorously on finite field arithmetic and abstract algebra. They can be used to either detect or correct errors. The encoder for the block codes accept a message blocks of $k$ information bits each. The
message block is represented by the binary $k$-tuple $u = (u_0, u_1, ..., u_{k-1})$, called a message. There are totally $2^k$ different possible messages. The encoder maps each of these messages into a corresponding $n$-tuple $v = (v_0, v_1, ..., v_{n-1})$ called a codeword. Therefore there are totally $2^k$ different codewords possible at the output of an encoder. These $n-k$ redundant bits added to the $k$ information bits to form $n$ bits codeword are the parity bits. This set of $2^k$ codewords of length $n$ is called $(n, k)$ block code. In block codes, the input and output of the encoder does not represent the entire information and coded sequence rather it represents a block of them. On the receiver side the decoder accepts these $n$ bit codeword, detects and corrects error, and maps them back to the $k$ bit information word.

Linear block codes as shown in Figure 2.2 can be cyclic or non-cyclic. Any block code is cyclic if every cyclic shift of the codeword $v$ is also another codeword [28]. Cyclic codes are attractive in the sense that encoding and syndrome computations can be easily implemented using shift registers with feedback connections and because of there algebraic structure, it is possible to decode them using methods. Bose-Chaudhuri-Hocquenghem codes (BCH) [29][30] and Reed Solomon codes (RS) are popular among the cyclic codes. Reed Muller codes are good example for non-cyclic codes.

Convolutional codes are one of the commonly used channel codes in communication systems. These codes are developed with a strong mathematical structure and are primarily used for real time error correction. The encoder for convolutional codes also converts the $k$-bit blocks of message word into $n$-bit blocks of codeword. The encoder of convolutional codes uses $m$ memory hence the encoded block not only depends on the current input block but also on the previous $m$ block. In convolutional codes, the input
and output of the encoder represents the entire information and coded sequence. The main decoding strategy for convolutional codes is based on Viterbi algorithm.

Discussing about all types of error control codes is beyond the scope of this thesis; hence we will divert our focus towards some of the commonly used block codes and convolutional codes. The following section briefly explains the binary-BCH codes, Reed Muller codes (RM), Reed Solomon codes (RS) and convolutional encoding with Viterbi decoding. The complete study of these codes can be done in [28]37.

2.2 Binary Primitive BCH Codes

The Bose-Chaudhuri-Hocquenghem codes (BCH) [29][30][31][33] form a large class of random error correcting cyclic codes that occupies a prominent place in the practice of error correction. They use relatively simple encoding and decoding techniques. In this thesis, a subclass of binary BCH codes, the primitive BCH codes, is considered as they need less complex circuitry to implement in digital hardware.

For any positive integer $m \geq 3$ and $t < 2^{m-1}$, there exists a binary BCH code $(n, k)$ with the following parameters:

\begin{align*}
  n & = 2^m - 1 \quad \text{: Number of bits in the codeword.} \\
  k & = \text{Number of bits in the message word.} \\
  t & = \text{Maximum number of error bits that can be corrected.} \\
  n - k & \leq m \cdot t \quad \text{: Number of parity bits in a codeword} \\
  d_{\min} & \geq 2^t + 1 \quad \text{: Minimum distance.}
\end{align*}
Hence this code is capable of correcting \( t \) or fewer errors in an \( n \) bit codeword. The code rate \( r = k/n \). The large minimum distance and low error probabilities are obtained by increasing the number of parity bits at the cost of low code rate.

In cyclic codes all the message word, codeword and generator are represented as polynomials, as they are easy to manipulate. The codeword polynomial with above parameters is formed by multiplying the message polynomial with the generator polynomial.

\[
v(x) = u(x) \cdot g(x). \tag{Eq. 2.1}
\]

Where, \( u(x) = u_0 + u_1x + \ldots + u_{k-1}x^{k-1} \) is the message polynomial and \( v(x) = v_0 + v_1x + \ldots + v_{n-1}x^{n-1} \) is the codeword polynomial. The \( g(x) = g_0 + g_1x + \ldots + g_{n-k-1}x^{n-k-1} \) is the generator polynomial given by Eq. 2.2. Where,

\[
g(x) = LCM\{ h_1, h_2, h_3, \ldots, h_{2t} \} \tag{Eq. 2.2}
\]

where, \( h_j \) is the minimal polynomial of \( \alpha^j \) \((0 < j < 2t + 1)\).

The codewords obtained using Eq 2.1 is non-systematic in the sense that the information bits and parity bits in the codewords do not assume specific order. Systematic codes are those in which the information bits and parity bits in the codewords appear explicitly. The systematic encoders are desirable because of there relatively simple decoding structure. To obtain systematic codes, let

\[
v(x) = x^{n-k} \cdot u(x) + b(x) \tag{Eq. 2.3}
\]

where \( b(x) = b_0 + b_1x + \ldots + b_{n-1}x^{n-1} \) and \( b_j \in GF(2) \). Then from Eq 2.1 and Eq 2.3 we get

\[
x^{n-k} \cdot u(x) = u(x) \cdot g(x) - b(x) \tag{Eq. 2.4}
\]

Thus, the \( k \) data bits, in the codeword generated from Eq 2.3 are present explicitly resulting in systematic codes.
The encoder circuitry of BCH codes [31] are easily implemented using shift register circuits as shown on Figure 2.3. The remainder $b(x)$ [28][29] can be obtained in a linear $(n-k)$-stage shift register with feedback connections corresponding to the coefficients of the generator polynomial $g(x) = 1 + g_1x + g_2x^2 + ... + g_{n-k-1}x^{n-k-1} + x^{n-k}$. The working of the encoder shown in Figure 2.3 is illustrated as follows:

- For first $k$ clock cycles, with the switch S2 in position 2, the information bits are transmitted and with the switch S1 on, parity bits are calculated simultaneously using the Linear Feedback Shift Register (LFSR).
- For clock cycles $k+1$ to $n$, with switch S2 in position 1 and the switch S1 off, the parity bits in the LFSR are transmitted through switch S2.

![Figure 2.3 The BCH Encoder](image)

The decoding of Binary-BCH code [1][34][35][36][37][38][41] $(n, k)$ is more complicated than the encoding process. The decoder takes in the received vector $r(x)$, locates the errors and corrects it to the codeword $v(x)$. Let,

$$v(x) = v_0 + v_1x + v_2x^2 + ... + v_{n-1}x^{n-1}$$
\[ r(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{n-1} x^{n-1} \]
\[ e(x) = e_0 + e_1 x + e_2 x^2 + \ldots + e_{n-1} x^{n-1} \]

be the transmitted polynomial, the received polynomial and the error polynomial respectively such that
\[ r(x) = c(x) + e(x). \]  
(Eq. 2.5)

As shown in Figure 2.4 [40][57], the decoding is broken into three steps:

2. Find the key equation \( \sigma(x) \).
3. Find the error locations.

![Figure 2.4 Block Diagram of a BCH Decoder](image)

The first step of the decoding process is, to calculate the syndromes \( S_j \) (for \( 1 \leq j \leq 2t-1 \)) and to store the received polynomial \( r(x) \) in a buffer register. One of the important features of any linear block code is that the syndromes do not depend on the received
vector \( r(x) \) but depends on the error locations \( e(x) \), as shown in Eq. 2.6. If no errors occur, the syndromes will all be zero.

\[
S_j = \sum_{i=0}^{n-1} e_i \alpha^{i+j}.
\]  
(Eq. 2.6)

The circuit calculating the syndrome \( S_j \) carries out \((n-1)\) multiplications by the constant value \( \alpha \) and \((n-1)\) single bit summations. For example, a circuit calculating \( S_3 \) for \( m = 4 \) and primitive polynomial \( p(x) = x^4 + x + 1 \) is shown in Figure 2.5. At first, the registers \( s_i \) \((0 \leq i \leq 3)\) are set to zero. Then the register \( s_0 - s_3 \) is shifted 15 times and the received bits \( r_i \) \((0 \leq i \leq 14)\) are clocked into the syndrome calculation circuit. Then the \( S_3 \) is obtained in the \( s_0 - s_3 \) register.

The other way of calculating syndrome is using Eq.2.7 [28]. In this method, \( S_j \) is the remainder obtained by dividing the received polynomial by minimal polynomial \( h_j(x) \), that is,

\[
r(x) = a_j \ast h_j(x) + b_j(x)
\]  
(Eq. 2.7)

where

\[
S_j = b_j(\alpha^j).
\]  
(Eq. 2.8)
For example the circuit calculating $S_3$ for $m = 4$ is shown in Figure 2.6. The minimal polynomial of $\alpha^3$ is

$$h_3(x) = 1 + x + x^2 + x^3 + x^4$$

and let $b(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$ be the remainder on dividing $r(x)$ by $h_3(x)$. Then

$$S_3 = b(\alpha^3) = b_0 + b_1 \alpha^3 + b_2 \alpha^6 + b_3 \alpha^9$$

$$= b_0 + b_2 \alpha + b_3 \alpha^2 + (b_1 + b_2 + b_3) \alpha^3.$$

The circuit in Figure 2.6 therefore operates by first dividing $r(x)$ by $h_3(x)$ to generate $b(x)$ and then calculating $b(\alpha^3)$. The result is obtained after the register $b_0 - b_3$ have been clocked 15 times.

![Figure 2.6 Second Method for Computing $S_3$, m=4](image)

The second step of the decoding process is computing the error location polynomial $\sigma(x) = \sigma_0 + \sigma_1 x + \ldots + \sigma_t x^t$. The Eq 2.9 [28] calculates the coefficients of error location polynomial $\sigma$ using the syndromes $S_j (1 \leq j < 2t)$.

$$\sum_{j=0}^{t} S_{t+i-j} \sigma_j = 0 \quad (i=1, \ldots, t)$$

(Eq. 2.9)
Then solving the \( \sigma(x) \) for roots gives the error positions. The coefficients of \( \sigma(x) \) can be calculated by using Peterson-Gorenstein-Zieler algorithm [32][42], Euclid’s algorithm [43] or the Berlekamp-Massey Algorithm (BMA) [35][37].

The error location polynomial \( \sigma(x) \) in BMA is obtained iteratively in \( t-1 \) cycles. For each iteration \( r \), the degree of \( \sigma(x) \) is incremented by one. The degree of \( \sigma(x) \) equals the number of corrupted bits because the roots of \( \sigma(x) \) are associated with the transmission errors. The discrepancy \( d_r \), is given by Eq (2.10)

\[
d_r = \sum_{j=0}^{t} S_{2r-j+1} \cdot \sigma_j .
\]  
(Eq. 2.10)

The discrepancy is zero, when the number of iterations \( r \) is greater than or equal to the number of errors \( t \). If \( r < t \), the discrepancy is non-zero and is used to modify the degree and coefficients of \( \sigma(x) \). Therefore BMA computes the smallest degree \( \sigma(x) \) such that Eq 2.9 holds. BMA can be implemented with inversion or inversionless. The inversionless BMA is more complicated and requires a greater number of multiplications than the BMA with inversion. On the other hand, inversion can takes \((m-1)\) clock cycles; as a result it is slow.

The other common method for computing the coefficients of error location polynomial \( \sigma(x) \) is using Euclid’s Algorithm (EA) [43]. In EA recursively calculates the Greatest Common Divisor (GCD) of the polynomials. The error evaluator polynomial is defined as in Eq 2.11.

\[
\delta(x) = \sigma(x)S(x) \mod x^{2^{r+1}}
\]  
(Eq. 2.11)

The problem is minimized in determining \( \delta(x) \) satisfying Eq 2.11. This is achieved by using EA to the polynomials \( r_0(x) = x^{2^{r+1}} \) and \( r_1(x) = S(X) \), such that at the \( j^{th} \) step
the degree of \( r_j(x) \) less than or equal to \( t_d \) such that \( r_j(x) = q_j(x)x^{2^{t_d}+1} + b_j(x)S(x) \). Then, \( \delta(x) = r_j(x) \) and \( \sigma(x) = b_j(x) \).

The last step in BCH codes decoding is to find the error location from the error location polynomial \( \sigma(x) \). The reciprocals of the roots of \( \sigma(x) \) gives these values and can be found by substituting \( 1, \alpha, \alpha^2, \ldots, \alpha^{t_d-1} \) in \( \sigma(x) \). Chien [44] proposed a method of obtaining this using sequential substitution.

The Chien search calculates the sum for every clock cycle as given by Eq 2.12

\[
\sigma_0 + \sigma_1 \alpha + \sigma_2 \alpha^2 + \ldots + \sigma_{t_d-1} \alpha^{t_d-1} \quad (j = 0, 1, \ldots, k-1)
\]  

(Eq. 2.12)

The received bit \( r_{n-1,j} \) is corrupted only if the sum equals zero. Therefore, the clock cycle \( j \) for which the sum equals zero identifies the location of error as \( r_{n-1,j} \). The errors found are then corrected. Figure 2.7 shows the circuit implementing the Chien search. The working of this circuit is as follows.

- First the registers are initialised with the coefficients of the error location polynomial \( \sigma_0, \sigma_1, \ldots, \sigma_t \) obtained from BMA.

- Then the sum \( \sum_{j=0}^{t} \sigma_j \) is calculated. If the sum equals zero, the error is found and the faulty received bit after being delayed in the buffer is corrected using an XOR gate.

- On each clock cycle the value in \( \sigma_t \) register is multiplied by \( \alpha \) and the sum \( \sum_{j=0}^{t} \sigma_j \) is calculated again. The above operations are carried out for every transmitted information bit (that is \( n \) times).
2.3 Reed-Solomon Codes

Reed-Solomon codes are the most important and widely used non-binary BCH codes [33][42][39]. Reed Solomon [46] codes are cyclic, linear block codes. Reed-Solomon codes are easier to decode than most other non-binary codes. Reed Solomon (RS) codes are very effective in correcting random symbol errors and random burst errors [47]. They are widely used in communication systems and data storage systems.

The t error correcting code with symbols from $GF(q)$ has the following parameter:

- $n = q - 1$ : Number of symbols in the codeword.
- $k = q - 1 - 2t$ : Number of symbols in the message word.
- $t$ : Maximum number of error symbols that can be corrected.
- $n - k = 2t$ : Number of parity symbols in a codeword
- $d_{min} = 2t + 1$ : Minimum distance.

Important features to notice are, RS codes operate on symbols of length $m$-bits which are elements of $GF(2^m)$, the length of the code is one less than the size of the code.

Figure 2.7 Chien’s Search Circuit
alphabet and the minimum distance is one greater than the number of parity check symbols. RS codes fall into the class of Maximum Distance Separable (MDS) [47] codes. As in the case of binary-BCH codes the large minimum distance and low error probabilities are obtained by increasing the number of parity bits at the cost of low code rate.

Encoding of RS \((n= 2^m-1, k= 2^m-1-2t, t)\) code is similar to that of the binary-BCH code. The non-systematic encoding is achieved by multiplying the message polynomial with the generator polynomial as shown in Eq 2.1. The generator polynomial is given by Eq 2.2. Because of there ease to decode, systematic encoding are preferred over non-systematic encoding. The systematic encoding is achieved using Eq 2.3.

The encoding circuitry of RS code is the same as shown in Figure 2.3 but the components used like adder; multiplier and the storage device must be capable of handling symbols which are several bits.

The decoding of RS \([1][28][32][34][35][36][37][38][39][41][42]\) codes is far more complicated, is shown in Figure 2.8. There are two different ways of decoding RS codes \([42][45]\) in the time or in the frequency domain. Here frequency domain decoding process will be considered. The decoding of RS code is similar to BCH code with one additional step required to estimate the magnitude of the error.

The [56] first step of decoding is the syndrome calculation using Eq 2.13:

\[
S_i = \sum_{j=0}^{n} r_j \alpha^{i-j} \quad 0 \leq i < 2t \tag{Eq. 2.13}
\]

where the \(r_j \in GF(2^m)\) are the received symbols. Note that the syndrome calculation of BCH codes is simpler. For RS codes the equation \(S_i^2 = S_{2i}\) does not hold because \(r_i \in GF(2^m)\). As shown in Figure 2.5 and 2.6 the adder and storage devices used to
implement syndrome calculation circuitry for BCH decoding are simple. In RS codes, these devices are little more complex as the operations are carried out between symbols.

The second step in decoding is finding the coefficients of error polynomial $\sigma(x)$. This is accomplished in the same way as BCH codes either using Berlekamp-Massey Algorithm or Euclid’s Algorithm. The BMA for RS codes requires twice as many iterations as BCH codes.

The third step is recursive extension, computing the equation

$$E_i = -\sum_{j=1}^{t} E_{i-j}\sigma_j \quad 2t \leq i \leq n-1$$

(Eq. 2.14)

where $E_i = S_i \quad 0 \leq i \leq 2t-1$

Youzhi [48] has shown that this step can be implemented with a BMA circuit by adding only additional control signals. If EA is used in determining the $\sigma(x)$ then the Chein search is used to find the error locations.

The last step is obtaining the error magnitudes $e_i$ by computing the inverse transforms this obtained by using Forney algorithm [41].

$$e_i = \frac{1}{n} \sum_{j=0}^{n-1} E_j \alpha^{-ij} \quad 2t \leq i \leq n-1$$

(Eq. 2.15)

This step is not required for BCH codes and in comparison to the Chien search is rather more complicated.
2.4 Convolutional Codes

Convolutional codes were first introduced by Elias [49] as an alternative to block codes. The Binary convolutional codes are the most popular error correcting codes which were widely used in wireless and satellite communications and broadcasting systems [50]. The Convolutional encoders have a natural trellis structure, and the decoding algorithms for convolutional codes are based on this structure. Viterbi introduced a decoding algorithm for convolutional codes. It is known as Viterbi algorithm.

Convolutional codes are complicated than linear block codes, difficult to implement, and have lower code rates, but have powerful error correcting capabilities. Convolutional codes are more difficult to decode because they are encoded using finite state machines that have branching paths for encoding each bit in the data sequence.

Convolutional code encodes information serially in short block lengths using Convolutional encoder. Convolutional codes uses memory of order $m$ and the output of the convolutional encoder not only depend on the current input but also on $m$ previous
inputs. A convolutional encoder with rate \( R = k/n \) and memory order \( m \), \((n, k, m)\), is a \( k \)-input, \( n \)-output linear sequential circuit with input memory \( m \); that is, inputs remain in the

![Diagram of convolutional encoder](image)

Figure 2.9 Convolutional Encoder of Rate \( \frac{1}{2} \) and \( m=3 \)

encoder for an additional \( m \) time units after entering the encoder. The large minimum distance and low error probabilities for convolutional codes are achieved by increasing the memory order \( m \) and not by increasing \( n \) and \( k \) as in the case of block codes.

Convolutional encoders can be classified into two categories based on the: feedforward and feedback encoders. Within each category it can either be systematic or nonsystematic encoders. The main focus on this thesis is feed forward nonsystematic encoder.

Convolutionally encoding the data is accomplished using a shift register and adders. In general a rate-\( k/n \) convolutional encoder has \( k \) shift registers, one per information bit, and \( n \) modulo-two adders. The output \( k \) bits are the linear combinations of the contents in the registers and the input information bits. A shift register is a chain of flip-flops wherein the output of the \( i^{th} \) flip-flop is tied to the input of the \((i+1)^{th}\) flip-flop. The

32

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
exclusive-or gate performs modulo-two addition of its inputs. A feedforward nonsystematic encoder for a rate 1/2, K = 4, m = 3 is shown in Figure 2.9 and the working is explained as follows.

The encoder cycle starts when an input clock edge occurs. When the input clock edge occurs, the output of the left-hand flip-flop is clocked into the right-hand flip-flop, the previous input bit is clocked into the left-hand flip-flop, and a new input bit becomes available. Then the outputs of the upper and lower modulo-two adders become stable. The output selectors (SEL A/B block) selects and transmits the output of the upper modulo-two adder in the first cycle and then in the following cycle the output of the lower modulo-two adder.

There are a number of techniques for decoding convolutional codes. The most important of these methods are, the Viterbi algorithm (VA) [52] which performs maximum likelihood decoding [53] and the Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm [54] was introduced as a Maximum a posteriori probability (MAP) decoding of convolutional codes. The Viterbi algorithm is an asymptotically optimum approach for decoding of convolutional codes in memory less noisy channel. This algorithm was applied to a host of problems encountered in the design of communication systems. Viterbi algorithms are usually preferred over the BCJR algorithm because the performance of these two algorithms in decoding convolutional codes is essentially identical and the Viterbi algorithms are simpler to implement.

The Viterbi decoder is a maximum likelihood decoder in that the decoded word is always the codeword which maximizes the conditional property of the received sequence. The conditional probability is given in Eq 2.16.
\[ P(r | v) = \prod_{i=0}^{n-1} P(r_i | v_i) \]  
(Eq. 2.16)

Where \( r \) is the received word and \( v \) is the transmitted codeword.

The trellis of convolutional codes has regular structure as shown in Figure 2.11. This repetitive pattern is used for decoding. The Viterbi algorithm can be either hard decision or soft decision decoding. The soft decision decoding is efficient in correcting errors when compared to hard decision decoding but it requires more complex circuitry. Hence, Hard Decision Viterbi Decoder (HDVD) is considered for the purpose of this thesis. It is implemented in three steps as shown in [41] Figure 2.10 and explained as follows.

![Block Diagram of Viterbi Decoder](image)

Figure 2.10 Block Diagram of Viterbi Decoder

Let, \( S_{k,t} \) is the state in the trellis diagram that corresponds to state \( S_k \) at time \( t \). Every state in the trellis is assigned a value denoted \( V(S_{k,t}) \). In the first step, Initialize time \( t = 0 \), \( V(S_{0,0}) = 0 \) and all other \( V(S_{k,t}) = +\infty \). At time \( t = t+1 \), compute the \( t^{th} \) branch metric

\[ M(r_i | v_i) = \sum_{j=1}^{n} M(r_i^{(j)} | v_i^{(j)}) \] from hamming distance \( \sum_{j=1}^{n} |r_i^{(j)} - v_i^{(j)}| \).

In the second step, the \( t^{th} \) partial path metric \( V(S_{k,t}) \) is computed from \( V(S_{k,t-1}) + M(r_i | v_i) \). Set \( V(S_{k,t}) \) to the best partial path metric going to state \( S_k \) at
time \( t \). The best partial path metric is obtained by comparing and selecting the smallest partial path metric. If there is more than one of the smallest partial path metric, then any one of them is arbitrarily chosen.

In the last step, Store the best partial path metric and its associated survivor bit and state paths. If \( t < L + m - 1 \), return to second step. The result of the Viterbi algorithm is a unique trellis path that corresponds to the maximum likelihood codeword.

![Trellis Diagram for Code Rate \( \frac{1}{2} \)](image)

2.5 Related Work

The recent research on wireless sensor network is mostly related on minimizing the energy consumption. As mentioned in Chapter 1, the protocols that are efficient for wireless networks may not be efficient for wireless sensor networks. The coding techniques used for wireless networks are expensive in terms of power while sensor networks need an energy-efficient coding technique for reliable data transmission. In the literature, there is limited work on energy-efficient error control schemes for sensor networks and some of them are reviewed as follows.

In [1], it showed that usage of ARQ is limited sensor networks due to the additional retransmission energy cost and overhead. In [5], convolutional codes are analyzed for
power in a frequency nonselective, slow Rayleigh fading channel. Results show that the energy required for encoding data is negligible. However, performing Viterbi decoding on a Strong-Arm using C is energy-intensive with average energy consumption per useful bit grows exponentially with the constraint length of the code and independent of code rate. Using forward error correction is inefficient if the decoding is performed using a microprocessor and an onboard dedicated Viterbi decoder is suggested. Simple encoding techniques that enable easy decoding might present an energy-efficient solution for sensor networks.

In [55], the Minimum Energy (ME) coding scheme for sources with unknown statistics and a new method of code-by-code detection that can detect and correct certain errors in the received codeword is proposed. This research combines the modulation with the error control scheme to minimize power. The on/off key modulation performance is improved with a ME-Coding.

2.6 Summary

This chapter provides the background information for framework of the thesis. Review of existing work related to this thesis is also presented. To the best of our knowledge, only the convolutional codes implemented in C program is evaluated for energy consumption and other FEC codes schemes remain unexplored. Hence in this thesis the study on energy consumption of various forward error correction (FEC) codes implemented in Application Specific Integrated Circuits (ASIC) and Field Programmable Gate Array (FPGA) level there corresponding bit error rate performance is carried. These two performance measure is then used to find the energy efficient error correction code.
CHAPTER 3

METHODOLOGY

As discussed, the energy constraint is the driving factor for all ongoing research in the field of wireless sensor networks. A lot of research work has been focused on devising energy-aware protocols for different layers of the protocol stack. Error control schemes used for reliable data transmission over traditional wireless networks consumes significant amount of power. The goal of this thesis is to find energy-efficient error control schemes for wireless sensor networks.

In this work, we consider forward error correction (FEC) codes and evaluate the power consumption of three different FEC codes, BCH, RS, and convolution codes on different platforms. We implemented the three codes with different constraints using hardware description language. Estimation of energy consumption is performed by synthesizing these codes under register transfer level (RTL) and field programmable gate array (FPGA). The code with the least power consumption is identified. All the comparison if under the assumption of the same error control performance which is evaluated by the Bit Error Rate (BER) test. This chapter discusses the method followed in obtaining the goal.
3.1 Why BCH, RS and Convolutional Codes?

Power consumption in a circuit is directly proportional to its complexity. In general, the encoder consumes negligible power when compared to the decoder. Thus the challenge is in choosing an error correcting code, which uses a less complex decoding circuit. As explained in Chapter 2, linear block codes can be either cyclic or non-cyclic. Cyclic codes are of interest and importance due to their rich algebraic structure which has extremely concise specifications and can be efficiently implemented using simple shift registers. Hence cyclic linear block codes are chosen. The most widely used cyclic codes for wireless applications are BCH codes [28]. For the purpose of this thesis, binary-BCH codes and the most popular non-binary BCH codes, namely RS codes, are studied and analyzed. We also studied and analyzed the convolutional code with Viterbi algorithm for comparison purpose.

The constraints including the length of the message word, the length of the codeword for block codes and convolutional codes, and the order of memory for convolutional codes are appropriately chosen for comparing the power consumption.

3.2 Implementation of Codes

After choosing the error correction codes, the next step is to implement the encoder and decoder circuits using Hardware Description Language (HDL). Hardware Description Language (HDL) is used for designing hardware systems that have multiple modules working concurrently and sensitive to time. HDLs can be used for designing the Application Specific Integrated Circuit (ASIC) or for programming the Programmable Logic Device (PLD), such as Field Programmable Gate Array (FPGA). Very high speed
integrated circuit Hardware Description Language (VHDL) [58][59][60] and Verification Logic Hardware Description Language (Verilog HDL) are the widely used HDLs for designing the Very Large Scale Integrated (VLSI) Circuits. VHDL and Verilog are IEEE standards and are supported by all the Electronic Design Automation (EDA) vendors. For system level modeling and simulation, VHDL is far more comprehensive than Verilog HDL. Hence, VHDL is used for implementation.

VHDL supports three different levels of abstraction for modeling hardware. They are behavioral level, structural level, and Register Transfer Level (RTL). The behavioural level describes the behaviour of the system with a set of sequentially executable instructions. The structural level describes the system as a network of components. The register transfer level describes the data flow in the system. In VHDL, the design can be hierarchical in which a complex module is broken into smaller sub-modules and is called from the top level. These models are then simulated and verified using EDA tools. Active-HDL is the tool we used for simulations.

The algorithms used for implementing the encoder and decoder circuits are explained in Chapter 2. The encoder circuits of linear block codes and convolutional codes have simple hardware and are easy to implement. Some of the issues considered while implementing the decoder circuits are as follows.

- In binary-BCH and RS codes, the Euclid's Algorithm (EA) [44] the Berlekamp-Massey Algorithm (BMA) [1][38] can be used to compute the coefficients of error polynomial \( \sigma(x) = \sigma_0 + \sigma_1 x + ... + \sigma_p x^p \). In EA, all the steps used for computation are identical and easy to implement in hardware. Hence for efficient hardware implementation EA is preferred over BMA [41].
The information received by the receiver shown in Figure 2.1 is quantized before decoding. Depending on the level of quantization, the decoding can be classified into Hard Decision Decoding (HDD) or Soft Decision Decoding (SDD). The HDD is used when the quantization level is two while SDD is used for quantization level greater than two. The SDD performs better than HDD but requires highly complex circuitry. Hence HDD is implemented to minimize the power in the decoder [41].

3.3 Performance Measure

The next step is to measure the performance of the implemented codes which is given by Bit Error Rate (BER). Where,

\[
BER = \frac{\text{Number of erroneous bits}}{\text{Total number of transmitted bits}} \quad \text{(Eq. 3.1)}
\]

The BER measures the performance of the error correction code and aids in comparing other codes. BER is affected by the several factors including noise in the channel, quantization technique used, code rate \( R \), energy per symbol to noise ratio \( E_s/N_0 \) and transmitter power level \( P_{out} \). The code rate is given by

\[
R = \frac{k}{n} \quad \text{(Eq. 3.2)}
\]

where \( k \) is the number of bits at the input of the encoder and \( n \) is the number of bits at the output of the encoder. The BER is shown to be directly proportional to the code rate and inversely proportional to energy per symbol noise ratio and transmitter power level [3].

The encoder encodes the data with code rate \( R \) and transmits it over the noisy channel. If the transmitter power level \( P_{out} \) is unchanged then the received energy per symbol
decreases to $R^*E_r$. Hence, the BER measured at the input of the decoder is larger than the BER of the data transmitted without coding [3]. This increase in BER is overcome by using a decoder that can correct errors. Proper choice of error correction coding will reduce the BER to several orders of magnitude. The difference in BER achieved by using error correction codes to that of uncoded transmission is referred to as coding gain. The coding gain can also be defined as the additional transmitter power level required in obtaining the same BER without coding. Hence BER aids in calculating the coding gain of the error correction codes.

The BER test is performed by simulations following the procedure shown in Figure 2.1.

Step 1: generating information bits.
Step 2: encoding the information bits into code words.
Step 3: transmitting the code words through a Gaussian channel.
Step 4: decoding the received code words and calculate the BER.

In the following, we explain the details performed in each step.

In Step 1, the information bits are generated by using a random number generator. The *rand* statement in Matlab is used for this purpose. It produces a uniform distribution of numbers in the interval 0 to 1. These numbers are converted to binary bits by taking the floor value of the generated number plus 0.5.

In Step 2, the randomly generated data is then sent to the encoder circuit and encoded into code words. In Step 3, the encoded cod words are transmitted over the noisy channel. Before transmitting, the encoded data is modulated using Phase Shift Keying (PSK) [61].
To avoid complexity in the decoder circuit, Binary-PSK (BPSK) is used, which is done by mapping 1/0 at the output of the encoder to -1/+1 of an antipodal baseband signal [61].

To understand the performance of the error control codes in the noisy channel, an Additive White Gaussian Noise (AWGN) channel is modeled. AWGN channel is the most important and widely used channel for modeling digital communications [Rob]. Adding Gaussian noise to the encoded data is done by generating Gaussian random numbers with desired energy per symbol to noise ratio. The variance $\sigma^2$ of additive Gaussian noise which has the power spectrum of $N_o/2$ Watts/Hz is equal to $N_o/2$. If the energy per symbol $E_s$ is set to 1, then we have $E_s/N_o = 1/2\sigma^2$ and the standard deviation $\sigma$ is given by Eq 3.4.

$$\sigma = \sqrt{1/2(E_s/N_o)}$$  \hspace{1cm} (Eq. 3.4)

Hence, the standard deviation $\sigma$ with desired $E_s/N_o$ is calculated and used in Eq. 3.5 to obtain a Rayleigh random variable $R$.

$$R = \sqrt{2*\sigma^2*\ln(1/(1-U))},$$  \hspace{1cm} (Eq. 3.5)

where $\sigma^2$ is the variance of the Rayleigh random variable and $U$ is a uniformly distributed random number.

The Gaussian random number $G$ obtained by using Rayleigh random variable $R$ is given by Eq. 3.6

$$G = \mu + R * \cos(2 * \pi * T),$$  \hspace{1cm} (Eq. 3.6)

where $T$ is a uniformly distributed random number and $\mu$ is the mean of the Gaussian random variable. The generated Gaussian noise is then added to the encoded bits and transmitted.
In Step 4, the received symbols are quantized and fed to the decoder to obtain the information bits. Quantization refers to the process of approximating the continuous set of values with a finite set of values. As explained in Section 3.1, the 2-level quantization is used to reduce the complexity of the decoder. In 2-level quantization, the received signal is mapped to zero if the signal level is greater than zero and mapped to one if the signal level is less than zero. The result obtained in this way is called hard decision. The hard decision decoder is used to decode the quantized data which are ones and zeros.

The decoded data is then compared with the corresponding input given to the encoder and the number of errors is calculated. Then Eq 3.1 is used to calculate BER for the coded channel.

The BER of the uncoded channel is theoretically calculated using Eq. 3.7 and compared with the simulation results obtained by following the above mentioned steps.

\[
P(e) = 1/2 * \text{erfc}(\sqrt{E_b / N_o}) = Q(\sqrt{2E_b / N_o})
\]

(Eq. 3.7)

The performance of the coded and uncoded channels is compared using the calculated BER.

3.4 Power Estimation in FPGA Design

Once the codes were written in VHDL and tested for performance, the power consumed by the each error control scheme is measured by targeting it on a FPGA. The FPGA used for this thesis is Xilinx Virtex-E XCV200E-6CS144 [62]. Where the XCV200E is the device type, -6 is the speed grade, Chip Scale (CS) is the package type and 144 is the number of pins. The Virtex family sets the standard for FPGA performance.
and density. The Virtex-E FPGAs, delivering new attributes that were only previously addressed by ASIC solutions, are ideally suited for data communications applications.

The Xilinx Integrated Software Environment (ISE) 7 along with Xilinx Power (XPower) [62] is used for estimating power. The Xilinx ISE 7 uses the steps shown in Figure 3.1 to generate the files required for the XPower to compute power.

![Figure 3.1 Design Flow in XPower](image)

The first step is the VHDL design entry. During the design entry the codes implemented using VHDL are added to the project created in the Xilinx Project Navigator. After design entry, the code is synthesized in the next step. During this step the VHDL program is converted to a netlist file which is accepted as an input to the following step. The synthesis is followed by the MAP step, during which the design is fit
into the available resources on the target device. An ASCII Physical Constraints File (PCF) is produced in MAP. The PCF contains timing constraints that are used by XPower to identify clock nets. It also provides temperature and voltage information if these constraints have been set in the User Constraints File (UCF). After mapping, in the Place and Route (PAR) step, the design is placed and routed according to the timing constraints. At the end, a physical design file called Native Circuit Description file (NCD) that contains information on an FPGA is produced in the MAP and PAR steps.

The timing simulations done in Active-HDL are verified using the waveform editor. The waveforms are exported as Value Change Dump (VCD) files.

In the last step, the XPower uses the files generated in the previous steps to estimate power consumed by error control codes. XPower calculates the power in the design by summing up the power consumed by each element. The power consumed by each switching element in the design is given by Eq 3.8.

\[ P = C \times V^2 \times E \times F , \]  

(Eq. 3.8)

where \( P \) represents the power in mW, \( C \) represents the capacitance in Farads, \( V \) represents the voltage in Volts, \( E \) represents the switching activity (average number of transitions per clock cycle), \( F \) represents the frequency in Hz.

The capacitance is determined from the NCD file. The voltage is a fixed value for a specific device and it is set by default in the XPower interface. \( F \times E \) gives the activity rate of the signal in the design. The activity rate is defined as the rate at which a net or logic element switches. For dynamic power calculation activity rates are expressed as a frequency, which is the most variable element in Eq 3.8. The activity rates are set in the VCD file generated from the timing simulation in Active-HDL.
3.5 Power Estimation in ASIC Design

After estimating power in the FPGA the codes are designed using ASIC and the power consumption in these dedicated boards are studied. Synopsys’s Design Compiler (DC) and Design Vision are used in for this purpose [63]. Design Vision is the GUI of Design Compiler.

![Design Flow in the Design Compiler](image)

To analyze power in ASIC, the codes are synthesized using the following method as shown in Figure 3.2. First the VHDL design is analyzed to check if it uses the synthesizable VHDL subset. Then the design is elaborated, where the design is built with generic and technology-independent components like Gate, Flip Flops, MUX, etc. It is
followed by uniquify. This step makes copies of the sub-design whenever it is referred in
the upper level of the hierarchy, and optimizes each copy in a unique way according to
the conditions and constraints. Then the last step of synthesis is compiling, where the
network generic components is translated into a netlist of the target library. Compilation
can be constrained in terms of power. After synthesis, the power report gives the power
consumed by the device.

3.6 Summary

This chapter explains the methodology followed in estimating the energy of the error
control codes. Tools used in this work are also explained.
CHAPTER 4

RESULTS AND DISCUSSION

This chapter is focused on the results obtained, which are plotted, tabulated and discussed. The performance of the implemented code is evaluated with the performance metrics, BER and coding gain. As mentioned in Chapter 3, the energy consumption of these implemented codes is estimated and tabulated for both XILINX FPGA and ASIC designs. The complexity involved in programming FPGA for each of the codes is also summarized.

4.1 Code Performance

The performance of the codes implemented is measured using BER and coding gain as metrics as explained in Section 3.3. As mentioned earlier, the theoretical expression for the BER of BPSK on AWGN channel which is otherwise referred to as an uncoded channel is represented in Eq. 3.7. The BER of uncoded channel is proportional to the energy per bit to noise ratio $E_b/N_o$. Hence, to compare the performance of the coded channel with the uncoded channel, it is convenient to relate the BER of the coded channel with $E_b/N_o$. However we have to solve the problem of comparability. Let, $E_s/N_o$ be the energy per symbol to noise ratio and $R = k/n$ be the
code rate of the encoder. For the uncoded channel, there is only one channel symbol per bit, therefore we have $E_s/N_o = E_v/N_o$. However, coding introduces redundant bits such that for every block of $k$-input bits the encoder generates $n$-output bits with $n>k$. Hence, for the coded channel,

$$E_s/N_o = E_v/N_o + 10 \log_{10} (1/R)$$  \hspace{1cm} (Eq. 4.1)

The BER of the uncoded channel vs. $E_o/N_o$ is plotted in the graph as the baseline. The value of $E_o/N_o$ is usually measured in dB and the BER is plotted as a logarithmic scale. Then by using the steps discussed in Section 3.3, the BER of coded channel is obtained through simulations and is plotted in the same graph using the relation shown in Eq. 4.1. For every point on the uncoded channel curve, there is a corresponding point on the coded channel curve showing the difference in performance. In most cases, the coded performance curve has less BER when compared to the uncoded performance curve. The reduction in BER by using a coded channel is referred as *coding gain*.

As mentioned in Chapter 3, the linear, cyclic block codes like RS codes and BCH codes and convolutional codes with Viterbi decoding algorithm are the only interests of this thesis. For this purpose, the BER performance of these codes with different constraints are simulated and plotted as follows.

### 4.1.1 BCH Codes

Figures 4.1 and 4.2 show the performance of binary-BCH codes compared to the uncoded channel. As shown in Section 2.2, a $(n, k, t)$ binary-BCH code can correct up to $t$-errors, where $t$ is directly proportional to the number of parity bits $(n-k)$. Hence the performance of various binary-BCH codes is analyzed by varying $n$ and $k$ and their
corresponding energy consumption is measured. For the purpose of this thesis, \( n = 31 \) and \( n = 15 \) is chosen with \( k \) varying in \{26, 21, 16, and 11\}.

Figure 4.1 shows the performance curve of binary-BCH codes implemented by keeping \( n \) as 31 and varying \( k \) in \{26, 21, 16, and 11\}. It is clear from Figure 4.1 that the BCH(31,11,5) code, which can correct up to 5 errors has the lowest BER curve with the highest coding gain of 5 dB and the lowest code rate of 0.355. While the BCH(31,26,1) code, which can correct up to 1 error has very minimal coding gain with the highest code rate of 0.839. The BCH(31,16,3) and BCH(31,21,2) has coding gain of 4 dB and 2 dB and code rate of 0.516 and 0.677 respectively.

![Figure 4.1 BER Performance of (31,11,5), (31,16,3), (31,21,2) and (31,26,1) Binary-BCH Codes](image)

Figure 4.1 BER Performance of (31,11,5), (31,16,3), (31,21,2) and (31,26,1) Binary-BCH Codes
Figure 4.2 shows the performance curve of binary-BCH codes implemented by fixing $k$ at 11 and by varying $n$ in $\{31, 15\}$. The $(15, 11, 1)$ code which can correct up to 1 error has very minimal coding gain. The coding gain of $(31, 11, 5)$ code is 5 dB.

It shows that with the number of parity bits increasing, the error correcting capability of the binary-BCH code increases, resulting in lower BER. Hence, increasing $n-k$ increases the coding gain but lowers the code rate $R$.

Figure 4.2 BER Performances of $(31,11,5)$ and $(15,11,1)$ Binary-BCH Codes.

4.1.2 RS Codes

RS codes are linear cyclic block codes and are explained in Section 2.3. A RS $(n, k, t)$ code can correct up to $t$ errors whose error correcting capability like BCH depends on the
number of parity bits \((n-k)\). Hence, to understand the performance of these codes, various length RS codes are implemented and their corresponding power consumption is measured. In this thesis, \(n = 31\) and \(n = 15\) are chosen and \(k\) varying in \(\{11, 15, 21, 25\}\).

Figures 4.3 and 4.4 clearly show that the coding gains of RS codes are not the same for all the BER. They exhibit something of a brick wall characteristic with the codes work poorly at high BER but show a sudden transition to extremely effective operation as the BER reduces. In other words, they perform worse than the uncoded channel for higher values of BER and gives asymptotically high gain at lower values of BER.

![Figure 4.3 BER Performance of (31,11,10), (31,15,8), (31,21,5) and (31,25,3) RS Codes](image)

Figure 4.3 BER Performance of (31,11,10), (31,15,8), (31,21,5) and (31,25,3) RS Codes
Figure 4.3 shows the performance curve of RS codes implemented by keeping $n$ as 31 and varying $k$ in \{25, 21, 15, 11\}. From Figure, we can see that using RS codes for BER of $10^{-1}$ or higher results in channel loss while using them for BER lesser than $10^{-1}$ results in extremely high gain. Because the RS codes correct symbol errors, hence they are more suitable for applications with BER lesser than $10^{-1}$. For BER lower than $10^{-1}$, the RS(31,11,10) code which can correct up to 10 symbol error has the lowest BER curve with a coding gain of approximately 6 dB and a minimum code rate of 0.355. While the RS(31,25,3) code capable of correcting 3 symbol errors has the largest BER curve with a coding gain of 2 dB and a maximum code rate of 0.806. The RS (31,15,8) code and the RS (31,21,5) code have coding gain of 4 dB and 3 dB and code rate of 0.4838 and 0.677, respectively.

Figure 4.4 BER Performance of (31,11,10) and (15,11,2) RS Codes
Figure 4.4 compares the RS codes by fixing $k$ as 11 and varying $n$ in $\{15, 31\}$. It shows the same properties as Figure 4.3 which has BER of $10^{-1}$ as the threshold. The performance of RS codes improves with the number of parity bits increasing only when the BER is less than threshold. For BER higher than $10^{-1}$, all the RS codes implemented results in coding loss.

Figure 4.5 compares the binary-BCH codes with the RS codes based on code rates. The binary-BCH codes show a better performance at higher BER than the RS codes. But for lower BER, the RS codes perform better than the binary-BCH codes. For example, let us consider the BCH(31,16,3) and RS(31,15,8) with code rate 0.5. For BER greater than $10^{-2}$, the BCH code performs better than the RS code while for BER lesser than $10^{-2}$ the RS codes perform better.

![Figure 4.5 Comparison of RS and BCH Codes](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
4.1.3 Convolutional Codes

Convolutional codes work differently from the linear block codes as explained in Section 2.4. Convolutional encoders \((n, k, m)\) with Viterbi algorithm for maximum likelihood decoding are considered in this thesis. Unlike block codes, low BER and large coding gain in convolutional codes are achieved not by increasing \(k\) and \(n\) but by increasing the memory order \(m\). Hence to analyze the performance of these codes, various convolutional codes of different memory order are implemented and their energy consumption is measured. In this thesis, simulation is done for convolutional encoders with fixed code rate \(R = \frac{1}{2}\) and by varying memory order \(m\) in \(\{3, 4, 5, 6\}\).

![Figure 4.6 BER Performance for (2,1,3), (2,1,4), (2,1,5), and (2,1,6) Convolutional Codes](image)

Figure 4.6 BER Performance for (2,1,3), (2,1,4), (2,1,5), and (2,1,6) Convolutional Codes
Figure 4.6 shows that the coding gain increases with the increased memory order $m$. The $(2,1,3)$, $(2,1,4)$, $(2,1,5)$, and $(2,1,6)$ codes approximately have a coding gain of 1 dB, 2 dB, 3 dB and 4 dB respectively. Moreover, at high BER the coding gain of convolutional codes are larger than the block codes. Thus convolutional codes are suitable for applications with larger BER. Convolutional codes implemented using soft decision decoder results in 3dB better performance than those implemented using hard decision decoding but requires highly complex hardware [41].

Figure 4.7 Comparison of Convolutional and BCH Codes

Figure 4.7 compares the binary-BCH codes with the convolutional codes. The binary-BCH codes show a better overall performance than the convolutional codes. For code rate
0.5, the Vit(2,1,6) code which has the maximum coding gain of all the implemented convolutional codes performs better than the BCH(31,16,3) code; while other implemented convolutional codes perform worse than the BCH(31,16,3).

4.2 Power Estimation on FPGA

As mentioned in Chapter 3, the codes are implemented in FPGA's using Xilinx ISE 7 and the power consumption is measured using XPow[62]. The complexity of the FPGA implementation is measured based on the amount of resources used. The device utilization summary is given by the number of slices used in implementing these codes in FPGA. There are totally 2352 slices available in Xilinx XCV200E FPGA [62] Each CLBs contain two Slices and each CLB can compute a logic function. The slice utilization of each of the codes obtained from the synthesis report is tabulated in Table 4.1.

From Table 4.1, it can be inferred that the complexity of the encoder and decoder circuitry of BCH codes and RS codes increases with the number of parity bits \((n-k)\) increasing. On the other hand, the complexity of the convolutional encoder circuitry and the Viterbi decoder circuitry increases with memory order \(m\) increasing.

We also plotted the complexity of all the codes in Figure 4.8. It can be observed that the encoder circuit of the convolutional codes and the BCH codes are less complex than that of RS codes. This is because the RS codes use non-binary encoders. For decoder, the binary-BCH codes have less complex circuit than those of the RS codes and the convolutional codes. Moreover, the complexity of the Viterbi decoder circuitry increases alarmingly with the memory order \(m\) increasing, while the complexity of the binary-BCH
and RS decoders have a linear increase. Clearly, the binary-BCH codes have the least complexity and hence should consume the least power.

Table 4.1 Slice utilization summary

<table>
<thead>
<tr>
<th>Codes</th>
<th>Encoder</th>
<th>Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCH(15,11,1)</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>BCH(31,26,1)</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>BCH(31,21,2)</td>
<td>12</td>
<td>32</td>
</tr>
<tr>
<td>BCH(31,16,3)</td>
<td>15</td>
<td>119</td>
</tr>
<tr>
<td>BCH(31,11,5)</td>
<td>18</td>
<td>169</td>
</tr>
<tr>
<td>RS(15,11,2)</td>
<td>24</td>
<td>445</td>
</tr>
<tr>
<td>RS(31,25,3)</td>
<td>40</td>
<td>783</td>
</tr>
<tr>
<td>RS(31,21,5)</td>
<td>50</td>
<td>1088</td>
</tr>
<tr>
<td>RS(31,15,8)</td>
<td>68</td>
<td>1190</td>
</tr>
<tr>
<td>RS(31,11,10)</td>
<td>90</td>
<td>1391</td>
</tr>
<tr>
<td>Vit(2,1,3)</td>
<td>13</td>
<td>392</td>
</tr>
<tr>
<td>Vit(2,1,4)</td>
<td>14</td>
<td>768</td>
</tr>
<tr>
<td>Vit(2,1,5)</td>
<td>15</td>
<td>1398</td>
</tr>
<tr>
<td>Vit(2,1,6)</td>
<td>16</td>
<td>2277</td>
</tr>
</tbody>
</table>

Figure 4.8 Complexity Plots

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Power estimation of all the codes from XPower is tabulated in Table 4.2. It is shown that the encoding consumes less power when compared to the decoding for all three types of codes. It is clear that the power consumption in binary-BCH and RS codes are directly proportional to the number of parity bits and that of the convolutional code is proportional to the memory order $m$. The decoding of RS codes and convolutional codes consume significantly larger amount of power compared to that of the binary-BCH codes. Moreover, the power consumption of convolutional codes grows exponentially with the memory order $m$.

Table 4.2 Energy Analysis in FPGA Design

<table>
<thead>
<tr>
<th>Codes</th>
<th>Encoder(mW)</th>
<th>Decoder(mW)</th>
<th>Error Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCH(15,11,1)</td>
<td>1.299</td>
<td>1.7663</td>
<td>3.0654</td>
</tr>
<tr>
<td>BCH(31,26,1)</td>
<td>0.5346</td>
<td>0.7765</td>
<td>1.3111</td>
</tr>
<tr>
<td>BCH(31,21,2)</td>
<td>0.7238</td>
<td>2.1571</td>
<td>2.8809</td>
</tr>
<tr>
<td>BCH(31,16,3)</td>
<td>1.1206</td>
<td>10.0756</td>
<td>11.1962</td>
</tr>
<tr>
<td>BCH(31,11,5)</td>
<td>1.7509</td>
<td>16.25</td>
<td>18.0009</td>
</tr>
<tr>
<td>RS(15,11,2)</td>
<td>1.99</td>
<td>15.5763</td>
<td>17.5663</td>
</tr>
<tr>
<td>RS(31,25,3)</td>
<td>1.164</td>
<td>20.4892</td>
<td>21.6532</td>
</tr>
<tr>
<td>RS(31,21,5)</td>
<td>1.3233</td>
<td>42.53</td>
<td>43.8533</td>
</tr>
<tr>
<td>RS(31,15,8)</td>
<td>1.586</td>
<td>81.0066</td>
<td>82.5926</td>
</tr>
<tr>
<td>RS(31,11,10)</td>
<td>2.0172</td>
<td>141.72</td>
<td>143.7372</td>
</tr>
<tr>
<td>Vit(2,1,3)</td>
<td>1.4937</td>
<td>44.8875</td>
<td>46.3812</td>
</tr>
<tr>
<td>Vit(2,1,4)</td>
<td>1.725</td>
<td>101.8037</td>
<td>103.5287</td>
</tr>
<tr>
<td>Vit(2,1,5)</td>
<td>1.9425</td>
<td>189.2862</td>
<td>191.2287</td>
</tr>
<tr>
<td>Vit(2,1,6)</td>
<td>2.1375</td>
<td>386.2712</td>
<td>388.4087</td>
</tr>
</tbody>
</table>
The power consumption for implementation of different codes in FPGA is plotted in Figure 4.9. It can be shown that the power consumed by the decoder circuit is significantly higher than that consumed by the encoder circuit of all the codes. The binary-BCH codes consume less energy than the RS codes and convolutional codes. The power consumed by Viterbi algorithm for convolutional code increases exponentially with the memory order $m$ increasing. The binary-BCH codes consume the least power.

4.3 Power Estimation on ASIC Design

Power in ASIC design is obtained as explained in Section 3.5. The target library used is $lsi_{10k}.db$ [63]. This target library only gives the net switching power of the implemented designs. The internal power of the design is not included in this comparison due to the unavailability of the technology library.
Table 4.3 gives a clear picture of the power consumed by the encoder and decoder circuitry of the binary-BCH codes, RS codes and convolutional codes implemented in ASIC. The power consumption of binary-BCH codes and the RS codes increases with the number of parity bits \((n-k)\) increasing. On the other hand, the power consumption of the convolutional encoder circuitry and the Viterbi decoder circuitry increases with memory order \(m\) increasing.

Table 4.3 Energy Consumption in ASIC Design

<table>
<thead>
<tr>
<th>Codes</th>
<th>Encoder(nW)</th>
<th>Decoder(nW)</th>
<th>Error Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCH(15,11,1)</td>
<td>1.3725</td>
<td>5.7295</td>
<td>7.1021</td>
</tr>
<tr>
<td>BCH(31,26,1)</td>
<td>1.1598</td>
<td>3.5673</td>
<td>4.7271</td>
</tr>
<tr>
<td>BCH(31,21,2)</td>
<td>1.909</td>
<td>6.4619</td>
<td>8.3709</td>
</tr>
<tr>
<td>BCH(31,16,3)</td>
<td>3.12</td>
<td>21.9131</td>
<td>25.0331</td>
</tr>
<tr>
<td>BCH(31,11,5)</td>
<td>5.554</td>
<td>45.0825</td>
<td>50.6365</td>
</tr>
<tr>
<td>RS(15,11,2)</td>
<td>15.3709</td>
<td>109.0909</td>
<td>124.4618</td>
</tr>
<tr>
<td>RS(31,25,3)</td>
<td>8.4068</td>
<td>63.688</td>
<td>72.0948</td>
</tr>
<tr>
<td>RS(31,21,5)</td>
<td>14.0452</td>
<td>86.7809</td>
<td>100.8261</td>
</tr>
<tr>
<td>RS(31,15,8)</td>
<td>28.2573</td>
<td>130.1933</td>
<td>158.4506</td>
</tr>
<tr>
<td>RS(31,11,10)</td>
<td>71.2254</td>
<td>188.3363</td>
<td>259.5618</td>
</tr>
<tr>
<td>Vit(2,1,3)</td>
<td>12.7757</td>
<td>51.95</td>
<td>64.7257</td>
</tr>
<tr>
<td>Vit(2,1,4)</td>
<td>14.613</td>
<td>76.2112</td>
<td>90.8242</td>
</tr>
<tr>
<td>Vit(2,1,5)</td>
<td>15.8</td>
<td>132.775</td>
<td>148.575</td>
</tr>
<tr>
<td>Vit(2,1,6)</td>
<td>19.1723</td>
<td>282.675</td>
<td>301.8473</td>
</tr>
</tbody>
</table>

The power consumption for different codes in ASIC is also plotted in Figure 4.10. It can be inferred that the power consumed by the decoder circuit is significantly higher than that consumed by the encoder circuit of all the codes. The binary-BCH codes
consume less energy than the RS codes and convolutional codes. It is evident that the power consumed by Viterbi algorithm for convolutional code increases exponentially with the memory order $m$ increasing. As we expect, the binary-BCH codes consume the least power.

Though not shown in the thesis, we have conducted simulations of part of codes using the TSM 0.18μm technology [64]. In general, the internal power is 1.5 times larger than the switching power. Thus, the total power consumed is in the order of 2.5 times the tabulated results, which is less than the power consumed on FPGA.

![Power Analysis in ASIC Design](image)

Figure 4.10 Power Consumption in ASIC Design

4.4 Power Analysis for Sensor Nodes

From the results it is very clear that the power consumption in ASIC implementation design is very less compared to that of FPGA implementation. The total power consumed in the sensor node for communication with coded channel is given by Eq. 4.2 [5].
\[ E(k, d) = \{E_T(k, d) + E_{encode}\} + \{E_R(k) + E_{decode}\} \quad \text{(Eq 4.2)} \]

where,

- \( E_T \) is the energy used by the transmitter circuitry, which is given by \( E_{elec} \cdot k \);
- \( E_R \) is the energy used by the receiver circuitry, which is given by \( E_{elec} \cdot k \);
- \( E_{encode} \) is the energy used to encode, which can be obtained from the results in Section 4.3;
- \( E_{decode} \) is the energy used to decode, which can be obtained from the results in Section 4.3.

For uncoded channel, the receiving power is same as that of the coded channel, while the transmission power is given by

\[ E_T(k, d) = E_{elec} \cdot k + E_{amp} \cdot k \cdot d^2 \]

where,

- \( E_{elec} \) is the energy consumed by transmitter/receiver electronics
- \( E_{amp} \) is the energy consumed by the amplifier.
- \( k \) is the no. of message bits.
- \( d \) is the distance between the sensor nodes.

Figure 4.11 compares the power consumption of the coded channel and uncoded channel with encoders/decoders implemented in ASIC. It can be inferred that the power consumed by the binary-BCH codes and RS codes for coded channel is less than that of the uncoded channel. While coded channel using convolutional encoder with Viterbi decoder consumes more power than that of the uncoded channel. Hence it is clear that convolutional codes are not suitable for wireless sensor networks.
Figure 4.11 Power Consumption in Sensor Node for Coded and Uncoded Channels

Figure 4.12 Power Consumption in Transceivers and Error Control Circuitry

Figure 4.12 clearly shows that the significant amount of power consumed by the sensor node goes for error control if convolutional codes are used while power consumed
by transceivers dominate the total power consumption if binary-BCH codes or RS codes are used.

4.5 Discussion

The following inferences can be drawn from the results obtained.

- The binary-BCH codes have the least complexity when compared to the RS codes and the convolutional codes.
- The binary-BCH codes consume less power compared to the other two types of codes.
- For channels with BER higher than $10^{-1}$, the binary-BCH codes perform better than the RS codes while for BER lower than $10^{-1}$ the RS codes are better. However, by reducing the code rate, an equivalent BER can be achieved by the binary-BCH codes which consume less power than the RS codes. For example, from Figure 4.5, the BCH $(31,11,5)$ and RS $(31,15,8)$ codes perform the same when the $E_b/N_o$ is between 5 and 6. While the BCH $(31,11,5)$ code consumes significantly less power when compared to the RS $(31,15,8)$ code.
- The binary-BCH codes are simple to implement and achieves comparable BER performance. The energy per bit performance of binary-BCH codes is also better than the other kinds of codes. Hence it is best suitable for wireless sensor networks.
- It's expected that implementation in ASIC design saves a lot of power when compared to the FPGA design.
CHAPTER 5

CONCLUSIONS

This thesis is focused on the study of energy-efficient error control scheme for wireless sensor networks. In Chapter 1, we first address the design factors of wireless sensor networks and point out energy is the major constraint in wireless sensor networks. We then describe the importance of error control and the need to implement an energy-efficient error control scheme. In Chapter 2, we describe error control schemes with emphasis on BCH codes, RS codes and convolutional codes and review existing work on error control schemes for wireless sensor networks. In Chapter 3, the approaches and methodology for evaluating error correction capability and power consumption are described. In Chapter 4, we present and discuss the testing results of the three error control schemes in terms of error correction capability (in BER and coding gain) and power consumption evaluated on FPGA and ASIC.

The testing results show that the binary-BCH codes consume the least power with comparable coding gain, the RS codes consume more power with poor coding gain for channels with high BER, while convolutional codes perform well on high BER channels but have the energy consumption increasing exponentially with their memory order. The energy per bit performance of binary-BCH codes is also better than the other kinds of codes. It is also expected that the ASIC implementation of these codes further reduces the power consumption, which fits the needs for wireless sensor networks.
5.1 Future work

The power consumption of these codes on ASIC with more advanced design libraries will be conducted. We will also consider other coding schemes with simple encoding techniques that enable easy decoding. In addition, application-specific error control schemes will be devised and evaluated.
BIBLIOGRAPHY


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


VITA

Graduate College
University of Nevada, Las Vegas
Gopinath Balakrishnan

Home Address:
4210, Fairfax Circle, #1
Las Vegas, NV 89119

Degree:
Bachelor of Engineering, Electronics and Communication Engineering, 2002
University of Madras, India

Thesis Title: Performance Analysis of Error Detection and Correction Codes for Wireless Sensor Networks

Thesis Examination Committee:
Chairperson, Dr. Mei Yang, Ph.D.
Committee Member, Dr. Venkatesan Muthukumar, Ph.D.
Committee Member, Dr. Yingtao Jiang, Ph.D.
Graduate Faculty Representative, Dr. Yoohwan Kim, Ph.D.