A parallel BCH decoder
by Benjamin Arthur Laws

A thesis submitted to the Graduate Faculty in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY in ELECTRICAL ENGINEERING
Montana State University

Abstract:
The subject of this thesis is the design of a special purpose digital computer to decode the binary BCH codes.

The content of the thesis may be summarized as follows: First, a review of BCH decoding algorithms and decoders is made, including the applicable micro cellular and macrocellular techniques. Second, the BCH decoding algorithms are discussed and organized for execution by the parallel processor. Third, a detailed design for a parallel processor is described. Also included are discussions of fault detection and correction, operation for different code word lengths, and operation for different numbers of errors. Fourth, a computer program which simulates in detail the logic of the parallel processor is described. Fifth, a recommendation is made for further research ip the areas of parallel processor and algorithm design.

A PARALLEL BCH DECODER

by

BENJAMIN ARTHUR LAWS, JR.

A thesis submitted to the Graduate Faculty in partial fulfillment of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

Approved:

Paul E. Ehrlich
Head, Major Department

C.K. Rushforth
Chairman, Examining Committee

K. Goering
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

June, 1970

ACKNOWLEDGMENT

The author wishes to thank Dr, C. K. Rushforth for his guidance and encouragement during the course of this graduate work and thesis research.

The financial support ot the graduate work provided by the Electrical Engineering Department and National Defense Education Act Title IV Fellowship No. 67-06601 is gratefully acknowledged, as is the thesis support of National Defense Education Act Title IV Fellowship No. 67-06601 and Office of Naval Research Contract No. 8-0009-603.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The subject of this thesis is the design of a special purpose digital computer to decode the binary BCH codes.

The content of the thesis may be summarized as follows: First, a review of BCH decoding algorithms and decoders is made, including the applicable microcellular and macrocellular techniques. Second, the BCH decoding algorithms are discussed and organized for execution by the parallel processor. Third, a detailed design for a parallel processor is described. Also included are discussions of fault detection and correction, operation for different code word lengths, and operation for different numbers of errors. Fourth, a computer program which simulates in detail the logic of the parallel processor is described. Fifth, a recommendation is made for further research in the areas of parallel processor and algorithm design.

CHAPTER 1

INTRODUCTION TO THE BCH DECODING PROBLEM

1.1 Introduction

One of the fundamental requirements of our technology is reliable communications. The communication process can be modeled as shown in Figure 1.1. The block labeled SOURCE is the originator of some form of information. The ENCODER translates the source information into a form suitable for transmission across the CHANNEL. As the information flows across the CHANNEL, it is changed or corrupted by noise. The DECODER then translates the received signal into a form acceptable to the SINK. The SINK is the destination of the information.

The transmission of binary information is becoming increasingly important. Although channels require an analog representation of binary data, the analog translator may be considered a part of the channel. The output of the encoder is a stream of binary digits, as is the input to the decoder. The channel is probablistic, with some probability for complementing any given digit which is transmitted across the channel.

If a particular sequence of binary digits is transmitted, there is probability that some of these digits will be changed during transmission. The transmitted message will be lost if this happens. It is the task of the encoder to provide redundancy in the transmitted signal so that the correct bit sequence will be received even though some of the digits were complemented by the channel.

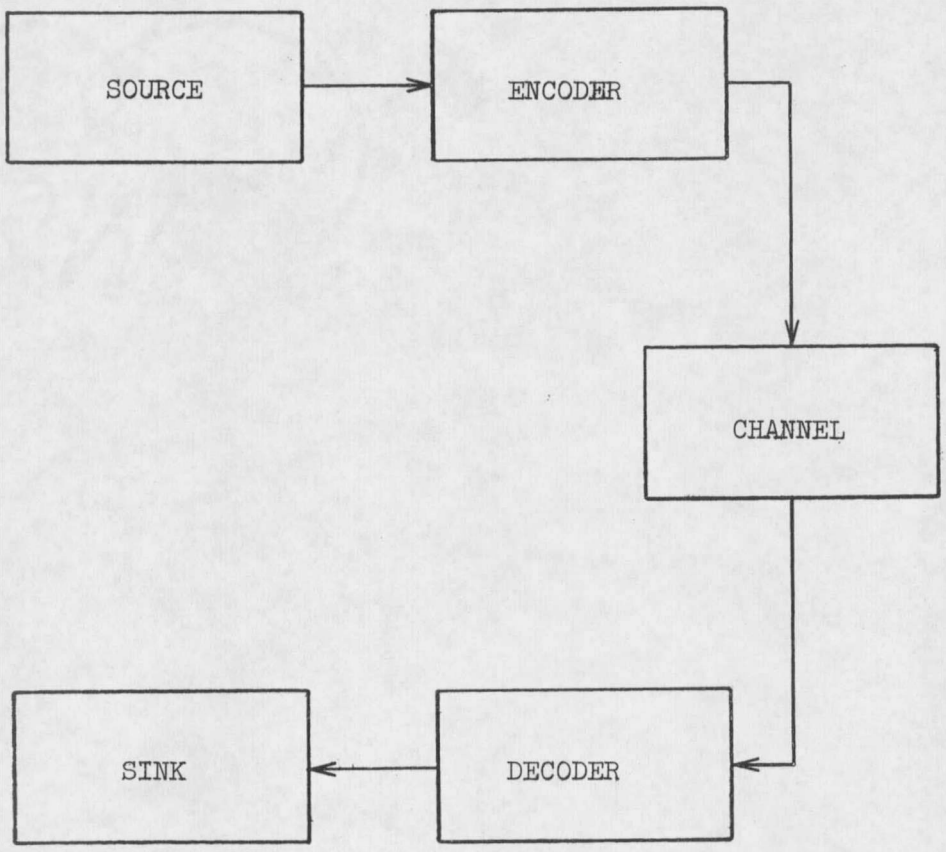

Figure 1.1. The Communication Process

1.2 Coding Theory

The motivation for coding theory was provided in 1948 by C. E. Shannon. He proved that messages can be transmitted over a channel, with an arbitrarily small probability of receiving an incorrect message, as long as the transmission rates are below channel capacity. In order to achieve this low probability of error, long complex codes are required. The central problem of Coding Theory is to find codes which will yield a low probability of error and will also be realizable in terms of hardware.

Consider the problem of encoding a sequence of binary digits. Suppose the source digits are broken into blocks of L digits. The encoder must add a set of N-L redundant (check) digits so the decoder may correct channel errors. Encoding is relatively easy to accomplish in comparison to decoding, and many simple encoders have been proposed.

The most difficult problem is the decoding--to determine which bits in the received word are in error (or to determine which L message digits were sent). Suppose the encoder provides code words such that the minimum Hamming distance between any two code words is d. Now define t to be the largest integer less than or equal to (d-1)/2. When a code word is transmitted across the channel, if the number of errors is less than or equal to t, the transmitted code word can be determined by comparing the received word with all possible code words until the

smallest Hamming distance is found. This procedure is known as maximum likelihood decoding. The process is simple conceptually, but it requires the complexity of the decoder to increase exponentially with code word length. Even if there were only 2^{32} code words, not an unreasonable number, over 2 billion comparisons would have to be made. If a comparison could be done every microsecond, over 2,000 seconds would be required to decode each received word.

One of the primary goals of Coding Theory is to find codes which can be decoded without requiring an unreasonable amount of time or equipment. The BCH codes satisfy this requirement-- although with few exceptions, the BCH decoder cannot correct as many errors as a maximum likelihood decoder.

The BCH codes were discovered in 1960 by R. C. Bose and D. K. Ray-Chaudhuri, and independently by A. Hocquenghem in 1959. They provide at least one solution to the coding problem. Only the primitive binary BCH codes will be considered, where the length of the code word is $N=2^m-1$, for $m>2$. The error-correcting capability of a length 2^m-1 code is specified by a parameter, t, in the encoding and decoding algorithms. No more than mt parity check bits are required by a t-error-correcting code. The length and error correction capability of these codes are quite flexible.

Non-binary BCH codes of length $N=p^m-1$ may be defined, where p is a prime number. Since we are concerned with transmission of

binary data, the non-binary BCH codes will not be considered.

The encoding problem for BCH codes may be solved by using a linear feedback shift register or a cellular array (1, 2, 4)*. The check digits may be computed using a generator polynomial for the particular error-correcting capability and code length desired. A number of tables useful in decoding and encoding BCH codes are given by Peterson (12).

The implementation of the BCH codes is based on arithmetic operations over a Galois field. The Galois field with p^m elements is represented $GF(p^m)$, where p is a prime number. This field defines a BCH code of length $N=p^m-1$. Again, since only the binary BCH codes are being considered, p will always be 2.

1.3 Decoding Algorithms for the BCH Codes

The first decoding algorithm for the BCH codes was published in 1960 by W. W. Peterson (8). This algorithm divides the computation into three parts:

1. Computation of the parity checks or syndrome digits.
2. Transformation of the syndrome digits into the coefficients of an error-locating polynomial--a polynomial whose roots indicate the location of errors in the word.
3. Solution of the error locating polynomial for its roots--each root indicates a correction in the received word.

*Numbers in parenthesis refer to numbered references in the Bibliography.

The algorithms are guaranteed to correct up to t errors. Peterson's decoding algorithm is the easiest to understand. However, if $e \leq t$ errors occur the algorithm requires the computation of (t-e) determinants ranging in size from t by t to e by e. A set of e simultaneous equations must then be solved to obtain the coefficients of the error locating polynomial.

T. C. Bartee and E. I. Schneider (13) have actually built and tested a special purpose computer for decoding a length 127 five-error-correcting code. The machine is a fixed-program digital computer with a single arithmetic processor, and uses Peterson's algorithm. It features a combinational circuit to perform the Galois field multiplication operation.

In 1964, R. T. Chien (9) published an algorithm which showed a simpler way to perform step 3 of the Peterson algorithm, and a way to combine steps 2 and 3 so that less time and equipment are required. Decoders were shown for correcting a small number of errors. However, in the general case, it was still necessary to solve a set of e simultaneous equations to correct e errors.

The biggest step forward came when E. R. Berlekamp (1) described an efficient algorithm for step 2. This algorithm does not require the calculation of determinants. Instead of operating on matrices, the algorithm operates on length t+1 vectors or polynomials. The algorithm requires execution of t iterations and computes a trial

error locating polynomial to determine a new trial polynomial. After t iterations, the roots of the trial error locating polynomial give the location of t or fewer errors which may have occurred anywhere in the received word.

1.4 Decoder Design Using Large-Scale Integration

With the development of Large-Scale Integration (LSI), much attention is being given to cellular arrays for implementing both logic functions and computer systems. A cellular array is a set of interconnected cells arranged in some regular fashion. Each cell in a micro-cellular array contains a small number of gates and/or storage elements, while each cell of a macro-cellular array contains a large number of gates and/or storage elements.

1.4.1 Micro-Cellular Arrays

A comprehensive survey of cellular technology was compiled by R. C. Minnick (11). In addition, Minnick has published a large number of results produced by project work at Stanford Research Institute (10).

K. N. Levitt and W. H. Kautz (2) have published a paper showing cellular arrays for both encoding and decoding certain binary error correcting codes.

1.4.2 Macro-Cellular Arrays and Processors

A number of macro-cellular processors are of interest. The first is the SOLOMON (Simultaneous Operation Linked Ordinal Modular Network) computer introduced in 1962 by Slotnick, Borck and McReynolds (3). This computer has a central control and a square array of processing elements. The central control unit provides instructions to each processing element. Each processing element has storage, an arithmetic processor, and the ability to transfer data serially to and from its four nearest neighbors in the array.

The ILLIAC IV computer is a direct descendent of SOLOMON, but is much larger and has a much greater computational facility in the processor cells (3). It is intended to be supervised by an extremely powerful general purpose computer. The machine is currently under construction by the Burroughs Corporation.

The parallel matrix processor designed by L. E. Cannon (3) is also of interest. This machine is a square array of processor cells interconnected to perform matrix operations efficiently. It is a specially-organized matrix processor intended for use with a general purpose computer.

Berlekamp (1) provides a description of a specially-organized computer for implementing step 2 of the BCH decoding algorithm. Figure 1.2 shows the block diagram given by Berlekamp. Each processor cell or slave processor can do multiplication and addition in a