ON THE IMPLEMENTATION OF FAST ALGORITHMS FOR LINEAR CODES USING T805 MICROCOMPUTER ARRAYS

Domingo Rodríguez, Alberto Rodríguez and Nayda G. Santiago

University of Puerto Rico
Electrical and Computer Engineering Department
Mayagüez Puerto Rico 00681-5000
E-mail: domingo@rmece01.upr.clu.edu
E-mail: albert@rmece01.upr.clu.edu
E-mail: santia11@egr.msu.edu

ABSTRACT

This work deals with the implementation of fast algorithms for linear codes using T805 microcomputer arrays. Special emphasis is given to the design and implementation of algorithms for convolutional encoders. The results show that, by proper computational load distribution and good data techniques, improvements are obtained over direct implementations.

I. INTRODUCTION

Digital Signal Processing is an engineering field with many branches. Among them is the theory of Error Control Codes (ECC). The central problem treated by ECC is the detection and correction of information against errors that may appear during its transmission. It has been demonstrated that, by proper encoding of the information to be transmitted (or stored), certain errors induced by a noisy channel can be detected and corrected; and, thus, maintain an acceptable transmission rate. Currently, the use of error control techniques has become and integral part of any digital communications system.

II. BASIC COMMUNICATIONS SYSTEM

A basic communications system connects an information source and an information user through a channel, (see Figure 1). The information source transmits sequences, which enter the communications system, and are first operated on or processed by the *source coder*. The source coder is used to remove redundancies from the signals or sequences coming from the information source. The source coded sequences are called *information words*. The information words enter the *channel coder* which transforms them into encoded sequences called *code words*. A set of code words is called a *code*. The code words are longer sequences which contain more redundancy than the information words. The code words are then transmitted through the channel using a *modulator* (not shown in Figure 1).

Because the channel is subject to various types of noise, distortion, or interference, its output may be different from the channel input. Typical transmission media or channels include telephone lines, microwave links, satellite links, and so on. The *demodulator* (not shown in Figure 1), converts each received channel output signal into a *received word*.

The *channel decoder* transforms the received words, using the redundancy of the signals, and corrects the errors to produce sequences called *estimated words*. Ideally, estimated words will be equal to the information words. Finally, the *source decoder* performs the inverse operation of the source encoder, and delivers its output to the information user. ECC are dedicated to the channel coder/decoder stages only.

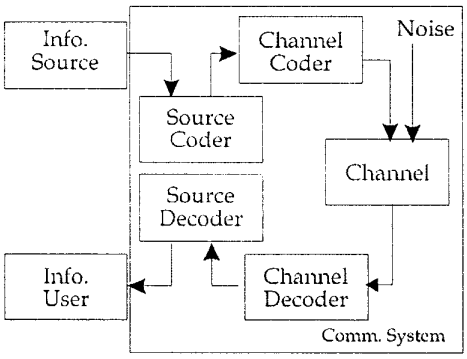

Figure 1. Basic Communications System

III. LINEAR CODES

This work deals with linear codes exclusively. Some basic definitions must be presented to define linear codes as entities of well defined mathematical structures. Let $GF(q)$ be a Galois field with q elements. Let $GF(q)^n$ be an n-dimensional vector space over $GF(q)$. Let $GF(q)[x]$ be the ring of polynomials over the field $GF(q)$ in the indeterminate x. A polynomial $a(x)=a[l-1]x^{l-1}+a[l-2]x^{l-2}+\ldots+a[1]x+a[0]$ is called the monic polynomial if $a[l-1]=1$. The polynomial $a(x)\in GF(q)[x]$ is called a prime polynomial if it is monic and irreducible; that is, if it is divisible only by itself. The polynomial ring $GF(q)[x]$ can be turned

0-7803-2972-4/96$5.00©1996 IEEE

into a quotient ring by taking an arbitrary polynomial $b(x) \in GF(q)[x]$ and performing polynomial addition and polynomial multiplication modulo the polynomial $b(x)$. This quotient ring will be denoted by $GF(q)[x]/b(x)$, and it is composed of all polynomials of degree less than $b(x)$. The quotient ring $GF(q)[x]/b(x)$ becomes a field if the polynomial $b(x)$ is a prime polynomial. Then, a linear code $\mathcal{L}$ is defined as a subspace of $GF(q)^n$. There are two types of codes in common use today: block codes and convolutional codes. From these two, convolutional codes have been selected for implementation.

IV. CONVOLUTIONAL CODES

Convolutional codes are linear codes which divide the incoming data stream into smaller blocks of length k_0, which are called *information frames*. These information frames typically contain no more than a few symbols. The information frames are encoded into *code word frames* of length n_0. Convolutional codes store previous information frames into a single code word. A convolutional code with these characteristics is known as an (mn_0, mk_0) convolutional code.

Any convolutional code can be expressed as a matrix-vector multiplication. But, to appreciate the parallel structure of these codes, they will be expressed by n_0 sets of finite-impulse-response (FIR) filters, each set consisting of k_0 FIR filters (see Figure 2).

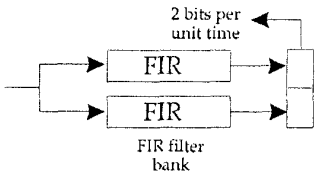

Figure 2. 1/2 Rate CE FIR Filters.

The impulse response of any FIR filter can be represented by a polynomial of finite degree. These polynomials are called the *generator polynomials* of the code. The rate of a convolutional code is defined as $R = k_0 / n_0$.

Parallel Structure of Convolutional Codes

A set of FIR filters can be put together in a generator-polynomial matrix, a $k_0 \times n_0$ matrix of polynomials given by $G(x) = \left[g_{ij}(x)\right]$. The input information frames should be considered as k_0 symbols in parallel. These frames may be represented by k_0 information polynomials $d_i(x)$ for $i = 1, 2, \cdots, k_0$; or as a row vector of such polynomials:

$$d(x) = \left[d_1(x), d_2(x), \cdots, d_{k_0}(x)\right]. \quad (1)$$

Similarly, the output code word can be represented by n_0 code word polynomials

$$c_j(x) = \left[c_1(x), c_2(x), \cdots, c_{n_0}(x)\right]. \quad (2)$$

The coefficients of the code word polynomials are interleaved in order to pass them trough the channel. The encoding operation can now be described compactly as the vector-matrix product, $c(x) = d(x)G(x)$, or equivalently,

$$c_j(x) = \sum_{i=1}^{k_0} d_i(x) g_{ij}(x). \quad (3)$$

By expanding equation (3), it can be demonstrated that the CE perform different filtering operations over the same data set (see Figure 3). This is the corner stone for the parallel implementation process.

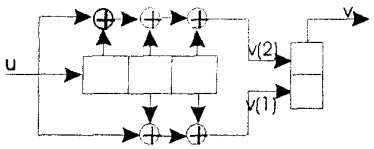

Figure 3. 1/2 Rate CE shift register.

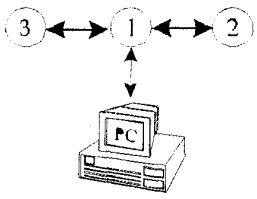

Figure 4. Transputer Topology for the Parallel Implementation of the 1/2 Rate CE

V. IMPLEMENTATIONS

Several convolutional encoders (CE) have been implemented using transputer computational structures. The (16,8) 1/2 rate CE was implemented in one, two and three transputer nodes where a transputer node is a single microprocessor chip.

(16,8) 1/2 Rate CE Implementation Concept

As mentioned before, a 1/2 rate CE performs two

FIR filtering operations on the same data set at the same time (see Figure 3). The generator polynomials are used to generate the code words. This is done by polynomial multiplication, but for these codes the operations are done strictly in *GF(2)*. Therefore, the polynomial multiplication and addition are done modulo-2. The polynomial multiplication modulo-2 can be represented as a shift register with connections to modulo-2 adders (see Figure 3). These connections were represented using pointers. The shift register has been implemented using a double linked list and the adders connections as memory pointers. Thus, every generator polynomial coefficient possesses a pointer to the memory address of its respective node in the linked list. Using this concept the polynomial multiplication has been reduced to data manipulation and additions making the implementation faster.

(16,8) 1/2 Rate CE Parallel Implementations

The 1/2 Rate CE representation shown in Figure 2 is very useful to see the inherent parallelism that these codes possess. It can be seen that the coder possesses two FIR filters plus an interleaving process at the filter's output. Therefore, 1/2 rate CE can be implemented using a maximum of three transputer nodes. Using three transputer processors each task can be easily distributed. The implementation is very simple, the transputer topology, or network, used for the implementation is shown in Figure 4. The network load distribution is as follows; The root node, or node number one, reads the data from the PC computer and delivers it to the entire network. The transputer nodes number two and three perform the FIR filtering process, while the root node is dedicated to receive and interleave the received data. This paralelization along with the software implementation techniques improves the implementation execution time.

(30,6) 1/ 5 Rate CE Transputer Implementation

The principles used to implement the 1/2 rate convolutional encoders can be used to implement the 1/5 rate. The major difference is the number of generator polynomials which span the (30,6) 1/5 rate convolutional encoder, which are five. Each generator polynomial can be seen as an FIR filter, as mentioned before, see Figure 5.

Figure 5 also shows the transputer topology used to implement this 1/5 rate convolutional encoder. In this parallel implementation five nodes are used to perform the filtering process and one extra node for the interleaving process. The FIR filtering operations should be assigned to the nodes wisely. It can be seen that nodes number two and four must deal with a high communications load, see Figure 5. These nodes are responsible of reading and transmitting data to and from nodes three, five and six. Therefore, the FIR filter that requires more computational power should be assigned to the nodes three, five, and six. One of the most important nodes is node number one or the root node, see Figure 4.

The root node is responsible of reading and distributing all the input data to the entire transputer network. When the transputer network finishes with the FIR filtering process, the root node collects the data for the interleaving process.

These implementations were performed using a data set that varies from 400 to 5000 symbols, where each symbol is represented by x bits, which depends on the code type. Also, different node to node communication rates were implemented to verify their performance.

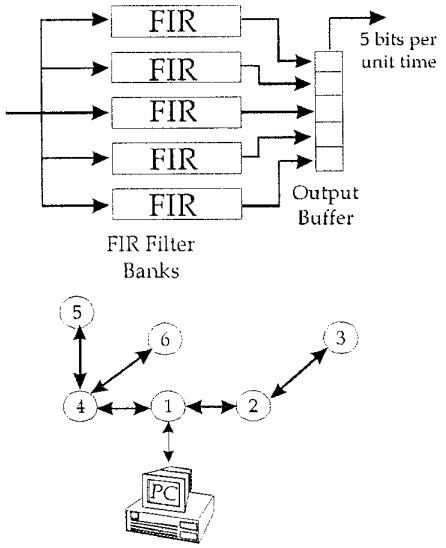

Figure 5. Six Nodes Transputer Topology and Filter Bank for the 1/5 Rate CE.

(16,6) 3/8 CE Parallel Implementation

The (16,6) 3/8 convolutional encoder possesses a 24 FIR filter bank. Instead of doing one shift register, as the 1/2 and 1/5 rate convolutional encoders cases, this encoder needs three double linked lists because it possesses eight generator polynomials. The implementation principle stills intact, the use of shift registers and memory pointers. In this implementation eight nodes are used for the filtering process and one for interleaving. The topology used for this implementation is shown in Figure 5. The task distribution is a follows; The filters labeled G1 were implemented in one transputer node, the filter labeled

G2 in another, and so forth. The filtering load is not the same for the entire network either. Transputer nodes two and nine has the highest communications load in the network. Therefore, the longest generator polynomials or the polynomials with the highest number of coefficients should be assigned to the nodes with less communications load.

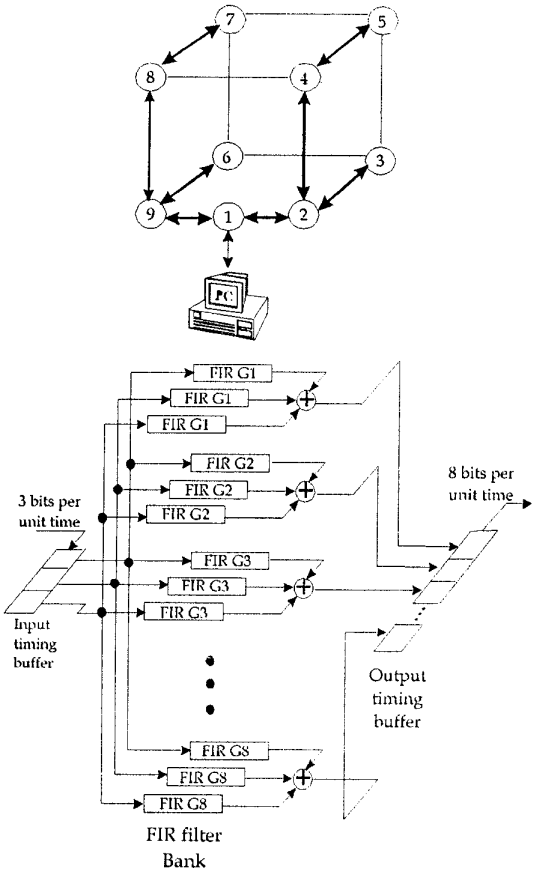

Figure 5. Nine Nodes Transputer Topology and Filter Bank for the (16,6) 3/8 CE.

VI. RESULTS

The experimental results presented in this section are based on the (16,8) 1/2 rate convolutional encoder sequential and parallel implementations. To understand the information provided in Table 1 the following definitions should be presented. Speed up, efficiency, and transmission rate. The speed up of an algorithm or implementation is measured as the ratio of the sequential implementation time over the parallel implementation time. The following equation defined the relation,

$$S = \frac{T_s}{T_p}.$$

The efficiency of an algorithm per processor is defined as the ratio of the speed up over the number of processors used. The following equation shows the relation,

$$E = \frac{S}{n}.$$

The implementations presented were tested using 40,000 bits as input. The time indicates how much time it took to complete each task. The table shows the associated transmission rate defined as follows,

$$TR = \frac{40{,}000}{T}.$$

Nodes	Packet Length	Time (s)	Speed Up	Trans. Rate	Eff.
1	--	1.049	--	38.09 Kbps *34.5 Mbps	--
3	1	.888	1.181	45 Kbps	39.38
3	8	.754	1.391	53.02 Kbps	46.39
3	16	.755	1.390	52.97 Kbps	46.34
3	24	.757	1.386	52.83 Kbps	46.23
3	32	.751	1.397	53.24 Kbps	46.58

* TMS320C31 Sequential Implementation

Table 1. (16,8) 1/2 Rate Convolutional Encoder Results

VII. CONCLUSIONS

The implementations reveal that CE possesses an inherent parallel structure suitable for parallel implementation. The parallel implementations showed a decrease in execution time when compared to sequential implementations. It can be experimentally demonstrated that the best communication rate is obtained as a function of the information word length; that is, there exists a relationship between the length of the information word and the transmission rate. Increasing computation time and decreasing communication time in each node decreases the overall time spent encoding a signal; but, this process has a limit.

VIII. REFERENCES

[1] R. E. Blahut, "Principles and Practices of Information Theory," Addison-Wesley Publishing Company, 1991.

[2] G. R. Redinbo, "Finite Fault-Tolerant Digital Filtering Architectures," IEEE Trans. on Computers, Vol. C-36, No.10, Oct. 1987.

[3] R. E. Blahut, "Algebraic Methods for Signal Processing and Communications Coding," Springer Verlag, New York, 1992.

[4] INMOS, "The Transputer Data Book," Consolidated Printers, CA, 1989.

[5] R. Tolimieri, M. An, C. Lu, "Algorithms for Discrete Fourier Transform and Convolution," Springer Verlag, New York, 1989.

[6] S. R. Whitaker, J. A. Canaris, K. B. Cameron, "Reed Solomon VLSI Codes for Advanced Television," IEEE Trans. On Circuits and Systems for Video Technology, Vol. 1, No. 2, June 1991.

[7] J. J. Komo, M. S. Lam, "Primitive Polynomials and M-Sequences Over GF(q^m)."IEEE Trans. of Information Theory, Vol. 39, No. 2, pp. 643-647, March 1992.