United States Patent [19]

Patterson

[11] **Patent Number: 4,623,999**

[45] **Date of Patent: Nov. 18, 1986**

[54] **LOOK-UP TABLE ENCODER FOR LINEAR BLOCK CODES**

[75] Inventor: **Patricia L. Patterson**, St. Petersburg, Fla.

[73] Assignee: **E-Systems, Inc.**, Dallas, Tex.

[21] Appl. No.: **617,167**

[22] Filed: **Jun. 4, 1984**

[51] **Int. Cl.**[4] **G06F 11/08**

[52] **U.S. Cl.** **371/37**; 371/40; 371/43; 371/45

[58] **Field of Search** 371/37, 38, 39, 40, 371/43, 45, 44; 364/200, 900

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,020,461	4/1977	Adams	371/37
4,030,067	6/1977	Howell	371/37
4,099,160	7/1978	Flagg	371/37
4,291,406	9/1981	Bahl	371/44
4,312,069	1/1982	Ahamed	371/37
4,359,772	11/1982	Patel	371/38
4,414,667	11/1983	Bennett	371/39
4,466,099	8/1984	Meltzer	371/37
4,473,902	9/1984	Chen	371/37
4,509,172	4/1985	Chen	371/38

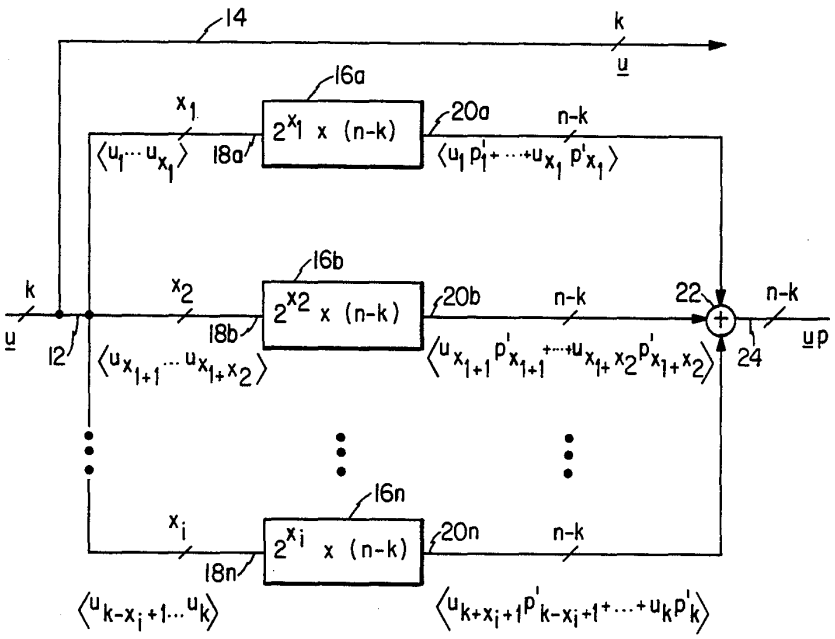
Primary Examiner—Michael R. Fleming

Attorney, Agent, or Firm—Albert M. Crowder, Jr.

[57] **ABSTRACT**

An efficient look-up table encoder for encoding k bit information words with linear error correcting block codes is provided comprising a plurality of read-only memories having 2^{x1}, 2^{x2}, . . . , and 2^{xi} address locations, respectively, where $x_1+x_2+ \ldots x_i=k$. Each of the read-only memories receives a portion of a k bit information word that serves to address a respective location therein, thereby mapping the portion of the k bit information word into an output word stored at the location. Corresponding bits of output words from the plurality of read-only memories are mod-2 summed by n−k exclusive-OR gates to generate a parity word associated with the information word. For systematic encoding, the parity word is then appended to the information word to form a codeword uniquely associated with the information word. This encoding scheme obviates look-up tables having 2^kx(n−k) storage locations.

14 Claims, 3 Drawing Figures



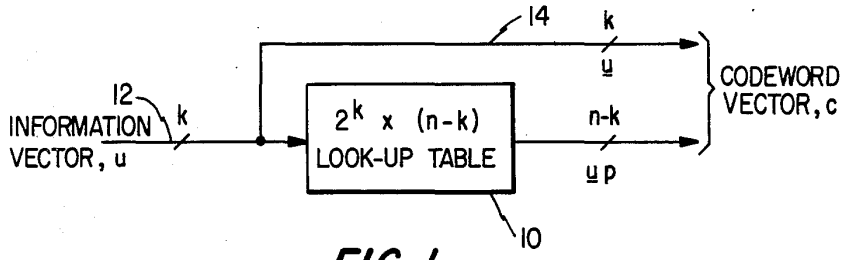


FIG. 1

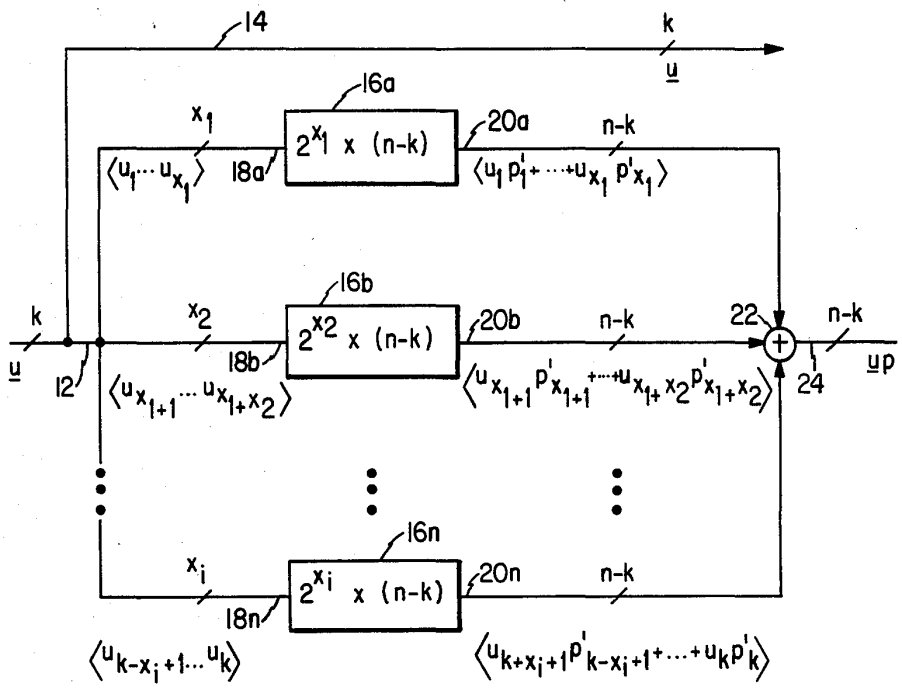


FIG. 2

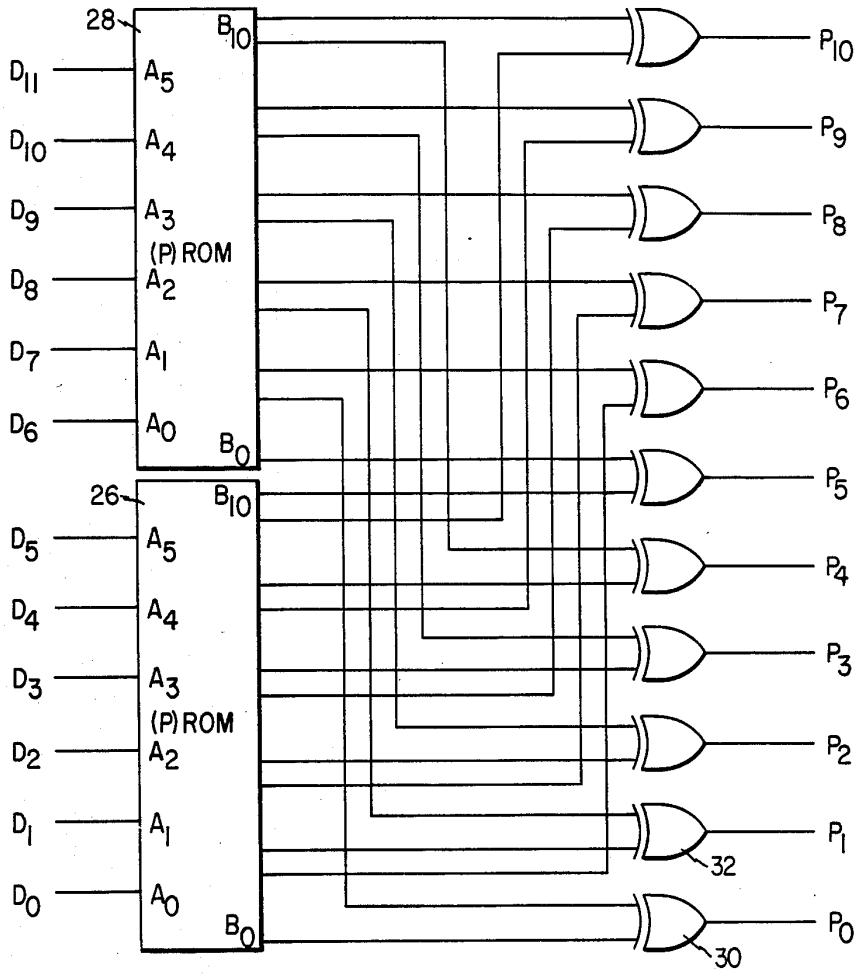


FIG. 3

LOOK-UP TABLE ENCODER FOR LINEAR BLOCK CODES

TECHNICAL FIELD

The present invention relates generally to encoding schemes and particularly to an efficient look-up table encoder for linear block codes.

BACKGROUND OF THE INVENTION

The use of linear block codes for error detection and correction in digital data transmission is well-known. Such codes act on information in blocks, each block forming an information word to be transmitted or stored. In the case of cyclic linear block codes, a codeword uniquely related to each information word is produced during the encoding scheme, this codeword being divisible by a fixed generator word. At the receiver, a received codeword is divided by the generator word. If a non-zero remainder results from this division, then errors have occurred during transmission. Moreover, this remainder, or syndrome, is uniquely related to an error pattern which is then corrected.

Suitable codewords for such schemes have been generated in a variety of ways. For systematic encoding of cyclic codes, one such method utilizes serial data input-/output wherein each information word is applied to an (n−k)-stage shift register with feedback connections based on a generator polynomial. After the information bits are shifted into the register and simultaneously into the communication channel, the n−k parity bits formed in the register are shifted into the channel, thus forming the complete codeword. This approach, however, is incompatible with parallel data input/output. To overcome this problem, suitable codewords have also been generated by table look-up encoding schemes which serve to map the k bits of each information work to n−k parity bits. These n−k parity bits are then appended to the k bits of each information word to define the codeword uniquely associated with each information word.

Standard look-up table encoding, however, requires memory devices with 2^k address locations. Such encoding schemes are thus impractical for medium length information words, and prohibitively complex and expensive when k is large. There is therefore a need for an efficient table look-up encoding scheme for linear block codes which obviates such large memory requirements.

SUMMARY OF THE INVENTION

The present invention describes an efficient look-up table encoder for encoding k bit information words with a linear error correcting block code which does not require storage devices having 2^k address locations. In the preferred embodiment, the encoder includes a plurality of storage devices having 2^{x1}, 2^{x2}, . . . , and 2^{xi} address locations, respectively, where $x_1+x_2+ \ldots +x_i=k$. Each of the storage devices receives a portion of a k bit information word that serves to address a respective location therein, thereby mapping the portion of the k bit information word into an output word stored at the location. The output words from the plurality of storage devices are then summed to generate a parity word associated with the information word. For systematic encoding of a linear block code, the parity word is appended to the information word to form a codeword uniquely associated with the information word.

Preferably, each of the storage devices is a programmable read-only memory (PROM). For systematic encoding, corresponding bits of output words from the plurality of PROM's are mod-2 summed by n−k exclusive-OR gates to generate the parity word, which is then appended to the information word to form the unique codeword.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following Description taken in conjunction with the accompanying Drawings in which:

FIG. **1** is a block diagram of a standard look-up table encoder of the prior art for encoding systematic linear block codes;

FIG. **2** is a block diagram of the look-up table encoder according to the present invention for encoding systematic linear block codes; and

FIG. **3** is a schematic diagram of a look-up table encoder for a Golay linear block code.

DETAILED DESCRIPTION

Referring now to the drawings wherein like reference characters designate like or similar parts throughout the several views, FIG. **1** is a block diagram of a standard look-up table encoder of the prior art. This encoder is utilized to generate a lengthened codeword vector c uniquely related to each information word vector, u. Referring to FIG. **1**, the information word vector u, which is a component vector of k information bits, is applied to the look-up table **10** via line **12**. The look-up table **10** has 2^k address locations each of which stores an n−k length digital word. In the case of systematic encoding of a linear block code, the information word vector u also forms part of the codeword vector c via line **14**. In operation, the information word vector u serves as an address to the look-up table **10** to select a parity word of n−k parity bits located at one of the 2^k address locations. The look-up table **10** thus maps the information word vector u to uP where P represents a parity check matrix unique to a given block code. As seen in FIG. **1**, the first k bits of the codeword vector c are the same as the k information bits. The standard look-up table encoding scheme then maps the k information bits to the n−k parity bits of the parity word given by uP. These parity bits are then appended to the information bits to define a codeword uniquely associated with the information word.

Standard implementation of the look-up table encoder **10** of FIG. **1** requires storage devices with 2^k address locations n−k bits in length. Such prior art look-up table encoders are therefore impractical even for medium length information words. Moreover, when k is large, look-up table encoders such as shown in FIG. **1** are prohibitively expensive due to these large memory requirements.

To ameliorate this problem, according to the present invention, the 2^kx (n−k) memory device **10** of FIG. **1** is replaced with $2 \leqq i < k$ smaller memory devices of 2^{x1}, 2^{x2}, . . . , and 2^{xi} address locations, respectively, where $x_1+x_2+ \ldots +x_i=k$. To appreciate the advantages of the present invention, it should first be noted that the n−k parity bits given by uP may be written in the form:

$$\underline{u}P = [u_1u_2 \ldots u_k]\begin{bmatrix} p_{11}p_{12}\cdots p_{1(n-k)} \\ \cdot \\ \cdot \\ \cdot \\ p_{k1}p_{k2}\cdots p_{k(n-k)} \end{bmatrix} \quad (1)$$

If each of the row vectors of the parity check matrix P in equation (1) is written as p'_k, then the n−k parity check bits can be designated as:

$$\underline{u}P = [u_1u_2 \ldots u_k]\begin{bmatrix} p'_1 \\ p'_2 \\ \cdot \\ \cdot \\ \cdot \\ p'_k \end{bmatrix} = u_1p'_1 + u_2p'_2 + \ldots + u_kp'_k \quad (2)$$

Now, if each of the elements of the information word vector u, which has 2^k possible values, is partitioned as follows:

$$[u_1u_2 \ldots u_k]=[u_1u_2 \ldots u_{x1}:u_{x1+1}u_{x1+2} \ldots u_{x1+x2}: \ldots :u_{k-xi+1} \ldots u_k], \quad (3)$$

then equation (2) can be written as:

$$\underline{u}P = (u_1p'_1 + u_2p'_2 + \ldots + u_{x1}p'_{x1}) + (u_{x1+1}p'_{x1+1} + \ldots + u_{x1+x2}p'_{x1+x2}) + \ldots + (u_{k-xi+1}p'_{k-xi+1} + \ldots + u_kp'_k). \quad (4)$$

As will be discussed in more detail below, the association of vectors as set forth in equation (4) allows replacement of the 2^kx(n−k) look-up table **10** as shown in FIG. **1** with $2 \leqq i < k$ memory devices of 2^{x1}, 2^{x2}, ... and 2^{xi} address locations, respectively, where $x_1+x_2+ \ldots +x_i=k$.

Referring now to FIG. **2**, a block diagram of the look-up table encoder of the present invention is shown.

In particular, the look-up table **10** of FIG. **1** is replaced by a plurality of storage devices **16***a*, **16***b*, ... **16***n*. The k bit information word vector u is supplied to the look-up table encoder via line **12**, and to the storage devices **16***a*, **16***b*, ... **16***n* via the lines **18***a*, **18***b*, ... **18***n*$_{x1}$, respectively. More specifically, storage device **16***a* has 2 address locations which are addressed by an x_1 bit-in-length portion of the information word vector u. In particular, the portion of the information word vector supplied via line **18***a* to the storage device **16***a* is represented by the vector:

$$[u_1u_2 \ldots u_{x1}] \quad (5)$$

This portion of the information word vector addresses the storage device **16***a* to select an output word of n−k bits at one of the 2^{x1} address locations. Similarly, the storage device **16***b* having 2^{x2} address locations is addressed by a x_2 bit-in-length portion of the information word vector u. As seen in FIG. **2**, this portion of the information word vector selects an output word of n−k parity bits at one of the 2^{x2} address locations and is represented by the vector:

$$[u_{x1+1}u_{x1+2} \ldots u_{x1+x2}] \quad (6)$$

Finally, the storage device **16**$_n$ having 2^{xi} address locations is addressed by an x_i bit-in-length portion of the information word vector u. This portion of the information word vector selects an output word of n−k parity bits at one of the 2^{xi} address locations and is represented by the vector:

$$[u_{k-xi+1}u_{k-xi+2} \ldots u_k]. \quad (7)$$

The output of storage device **16***a* is provided on line **20***a* and as discussed above, is an n−k bit output word. In particular, the x_1 bit vector applied to the storage device **16***a* serves to identify the output word:

$$[u_1p'_1+u_2p'_2+ \ldots +u_{x1}p'_{x1}] \quad (8)$$

Similarly, the output of storage device **16***b* is provided on line **20***b* and has the following form:

$$[u_{x1+1}p'_{x1+1}+ \ldots +u_{x1+x2}p'_{x1+x2}] \quad (9)$$

Likewise, the output of storage device **16***n* is provided on line **20***n* and is represented by::

$$[u_{k-xi+1}p'_{k-xi+1}+ \ldots +u_kp'_k] \quad (10)$$

Therefore, it can be seen that when summed, the output words provided from the storage devices **16***a*, **16***b*, ... **16***n* represent the various association of vectors set forth in equation (4).

Referring back to FIG. **2**, each of the output words or vectors on lines **20***a*, **20***b*, ... **20***n*, respectively, are then applied to n−k mod-2 adders represented generally by the symbol **22**. The n−k adders **22** sum the output words applied thereto to form the n−k parity bits uP which are then output from the look-up table encoder via line **24**. As discussed above, the n−k parity word bits uP are then appended to the k information word bits during systematic encoding of the linear block code.

Therefore, according to the present invention association of the vectors as set forth in equation (4) allows replacement of a 2^kx(n−k) storage device with $2 \leqq i < k$ smaller storage devices of 2^{x1}, 2^{x2}, ... and 2^{xi} address locations, respectively, where $x_1+x_2+ \ldots +x_i=k$. Each of these storage devices receives a portion of a k bit information word that serves to address a respective location therein, thereby mapping the portion into an output word associated with each storage device. The output words are then summed in n−k mod-2 adders to generate a parity word associated with each information word.

In the preferred embodiment of the invention, the storage devices **16***a*, **16***b*, ... **16***n* of FIG. **2** are read-only memories (ROM's), or programmable read-only memories (PROM's) where appropriate, such devices being programmed with binary information defining the parity check matrix P unique to a given block code. It should also be appreciated that the amount of storage locations in the memory devices **16***a*, **16***b*, ... **16***n* may be equal or variable. More specifically, the values of x_1, x_2, ... x_i may be identical or take on different values such that the number of address locations 2^{x1}, 2^{x2}, ... 2^{xi}, respectively, for each storage device may vary. The only constraint on the size of these storage devices is that $x_1+x_2+ \ldots +x_i=k$. By way of example only, if k=10, i.e., a 10 bit information word is desired to be transmitted, then the table look-up encoder of the present invention may utilize two ROM's having 2^8 and 2^2 address locations, respectively, or two ROM's each having 2^5 address locations, or three PROM's having 2^3,

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.