PROGRAMMABLE BIT-SERIAL REED-SOLOMON ENCODERS

S.T.J.Fenn, M. Benaissa, D.Taylor & J. Luty
School of Engineering, The University of Huddersfield, Queensgate,
Huddersfield, HD1 3DH, U.K.
Tel: +44 [1484] 472 156; Fax: +44 [1484] 451 833
e-mail: S.T.J.Fenn@hud.ac.uk

ABSTRACT

In this paper the design of programmable bit-serial Reed-Solomon encoders is considered using the traditional Berlekamp multiplier. It is suggested that there are certain advantages to be gained by deriving the generator polynomial of the code using combinational logic, or equivalently using look-up tables, rather than using an iterative LFSR based approach. The use of the recently proposed Berlekamp-like bit-serial multiplier is also considered and shown to demonstrate a number of potential advantages over the traditional Berlekamp multiplier in Reed-Solomon encoders.

1 INTRODUCTION

Reed-Solomon (RS) codes [1] are an important class of multiple bit correcting codes which are frequently used in noisy communications channels. For example, RS codes have been used in the Compact Disc system and the Hubble Space Telescope.

RS codes operate over structures called finite fields, denoted $GF(p^m)$, and because of the ease with which they can be implemented in digital hardware, these codes usually operate over $GF(2^m)$. When the RS codeword symbols are elements of $GF(2^m)$, m bits are required to represent the symbols. There are therefore essentially two types of RS encoder architecture - bit-serial and bit-parallel - in which the m bits representing field elements are represented in series or in parallel respectively.

This paper is concerned with the structure of bit-serial RS encoders, and in particular, in the design of programmable or rate-adaptive RS encoders. There are two reasons why we wish to design programmable RS encoders. Firstly, it is preferable to produce an RS encoder chip which can accommodate a range of parameters rather than to produce a range of encoder chips each implementing a fixed rate encoder. Secondly, if the encoder operates over a channel in which the bit-error rate varies, it is preferable not to have to run the encoder permanently assuming the worst case, as this reduces the data rate of the channel. Instead we look to use an RS code appropriate to the current characteristics of the channel, hence the need for rate-adaptive or programmable RS encoders and our interest in attempting to design them.

There are two related problems in the design of programmable (n, k) RS encoders, the determination of the generator polynomial g(x) for the code and the design of the constant finite field multipliers. There are a number of available bit-serial multipliers, for example dual [1], normal or triangular basis multipliers [2]. There are also two main ways of determining these generator coefficients, either calculating them on-line on the chip or through the use of ROMs (or equivalently, through combinational logic). For example in [2], a programmable architecture for an (n,k) RS encoder with variable k was presented operating over the triangular basis and generating g(x) on-line.

In this paper, we suggest using the traditional Berlekamp multiplier (TBM) in programmable bit-serial RS encoders [1]. We show that by using TBMs we can vary the value of n and not just k. This is achieved through designing a parameterised TBM. The architecture of this programmable (n, k) RS encoder is described and the method of generating the required coefficients of g(x) with combinational logic highlighted and shown to offer a number of

advantages. In addition, we show that the recently proposed Berlekamp-like multiplier (BLM) [3] demonstrates a number of advantages over the TBM in RS encoders.

2 RS ENCODERS

An (n, k) RS encoder has a very well known structure. If t = (n-k)/2 is the number of errors that can be corrected by the RS code, then the encoder circuit requires 2t constant multipliers, (2t-1) m-bit registers and 2t adders. Each adder is simply one or m XOR gates depending on whether the architecture is bit-serial or bit-parallel. The key issue in the design of an RS encoder is therefore the design of the multipliers and the generation of the coefficients of g(x) used by these multipliers.

Historically, the most appropriate multiplier for use in bit-serial RS encoders has been considered to be the TBM [1]. This is because the TBM has low hardware requirements and one TBM can carry out many constant multiplications. Hence an RS encoder utilizing the TBM requires only one such multiplier rather than 2t.

3 PROGRAMMABLE ENCODER CIRCUIT

A TBM multiplier has a structure that can be generalised and hence is appropriate for use in programmable RS codecs. A TBM for GF(2^m) comprises an m-length LFSR, m register elements and m AND gates and (m-1) XOR gates to carry out the required vector multiplication. The feedback terms for the LFSR comprise the non-zero coefficients of the defining irreducible polynomial for the field f(x). A schematic for this multiplier is shown in Fig. 1 where $f(x) = 1 + f_1x + ... + f_{m-1}x^{m-1} + x^m$.

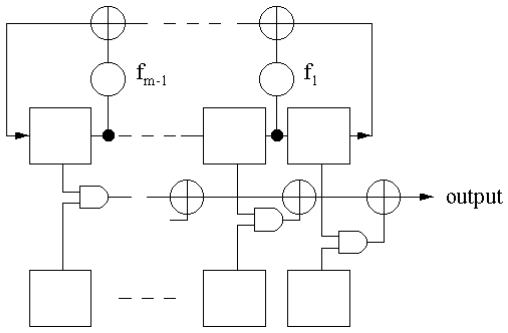

Fig. 1. Traditional Berlekamp multiplier

If the multiplier in Fig. 1 generates a = bc, where a,b,c ∈ GF(2^m) and c is the constant value, c is stored in the lower registers. But since this value is constant, it can be hardwired and the registers removed. The constant multiplication is then carried out by generating the inner product of the LFSR values with the polynomial basis representations of the constant multipliers g_i, (i=1,2, ...2t-1).

The presented RS encoder is programmable for m = 4, 5 and 8 and for various values of k. Hence the generalised constant TBM shown in Fig. 2 was designed. This circuit is essentially an 8-bit LFSR with extra combinational logic allowing it to also operate as a 5 or 4-bit LFSR. The control signals m_1 and m_0 determine whether Fig. 2 operates as a 8, 5 or 4-bit LFSR according to Table 1 below. 7 extra registers and 10 extra multiplexers are also required to allow for pipelined operation of the multiplier.

m_1	m_0	m
0	0	4
0	1	5
1	1	8

Table 1. Configuration of values for m_1 and m_0 to program the Berlekamp multiplier

Similarly, a programmable 4, 5 or 8 bit shift register is required in the encoder architecture. The design of this can be seen in Fig. 3. This circuit is programmed according to the configuration given in Table 1.

4 GENERATOR POLYNOMIAL UNIT

There are two main ways of deriving the generator polynomial for an RS code, on-line with a sequential circuit or through the use of a look-up table approach. The first of these methods has been used by Hasan in a rate-adaptive RS encoder [2]. With this technique, the generator polynomial coefficients are produced by a sequential circuit comprising a number of m-bit bidirectional LFSRs. By clocking these LFSRs the redundancy of the code is either increased or decreased by one, and so two clock cycles are required to increase or decrease the error-correcting ability of the code by one. In [2] these LFSRs are of a fixed length and so the overall encoder is an (n. k) RS encoder, for fixed n and variable k. However by utilizing programmable length LFSRs as described above, it is possible for a

more general programmable (n, k) RS architecture to be developed.

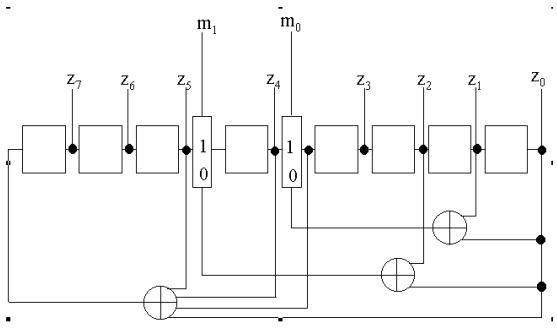

Fig. 2. Programmable constant Berlekamp multiplier

In the presented architecture this method of generating g(x) was not used in favour of a look-up table approach. This technique is considered to offer a number of advantages. Firstly, it is then straightforward to use symmetric generator polynomials which is not the case with the iterative approach given in [2]. Secondly, the technique given in [2] becomes considerably more complicated to implement if the value of m (and hence n) is also programmable. Finally, in generating g(x) by means of combinational logic instead of on-line, the error-correcting ability of successive codewords can differ by more than one. In fact using our approach, successive codewords can be encoded over different finite fields. For example with the approach of [2], raising the error-correcting ability of an RS code from 1 error to 8 errors take 14 clock cycles. Overall therefore, the complexity of this on-line generation approach was considered too complicated when n is to be varied as well as k.

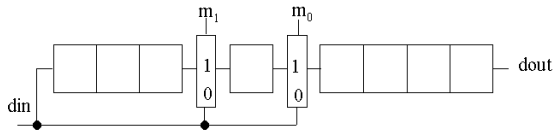

Fig. 3. Programmable 4,5 or 8-bit shift register

Instead a look-up table approach was used to generate g(x), and to construct the prototype RS encoder, combinational logic was used to implement this table. This necessitated the calculation of g(x) for 18 values of n and k. In total, the encoder can function as an (n, k) RS encoder where n=15 and (k=13,11,9,7,5,3,1), n=31 and (k=29,27,25,23,15) and n=255 and (k=253,251,249,247,239,223). 18 sets of polynomial coefficients were therefore calculated and the combinational circuits designed to generate these values. Because $n = 2^m-1$, the value of n can also be programmed by m_1 and m_0. The encoder can act as a 1,2,3,4,5,6,7,8 or 16 error correcting encoder and so four control signals T_0, T_1, T_2, T_3 were required to select the error-correcting ability of the code. Altogether therefore, six select lines program the parameters of the encoder.

As regards controlling the circuit, two control signals are required. The signal *control1* is required to generate the sequence 00..01 repeatedly, where there are (m-1) '0's. This signal controls the parallel loading of the m data bits of the finite field symbols into the Berlekamp multiplier. This signal is generated using a programmable finite state machine (FSM).

The signal *control2* is required to allow the first k information symbols to enter and leave the circuit and then for the remaining 2t check symbols to leave the circuit. To generate this signal a general counter was required that counts in binary from 0 to 2^m-1 every m clock cycles, where m = 4,5 or 8. This counter controls another programmable FSM that given this counter value and the control signals T_3, T_2, T_1, T_0, m_1 and m_0 generates a control signal that is high for m*k clock cycles and low for the following m*2t clock cycles.

5 OVERALL ENCODER STRUCTURE

The components described above were then combined to form the overall encoder shown in simplified form in Fig. 4. The encoder comprises 31 programmable shift registers, since the maximum value of 2t = 32. One programmable Berlekamp multiplier is required to produce the 8 bits to be multiplied by the constant values of the generator polynomial for the code. This is achieved with a programmable GF(2) inner product generator. This unit is constructed out of 8 AND gates, 7 XOR gates and a 3:1 multiplixer and generates either a 4, 5 or 8-bit GF(2) inner product, depending on the values of m_1 and m_0.

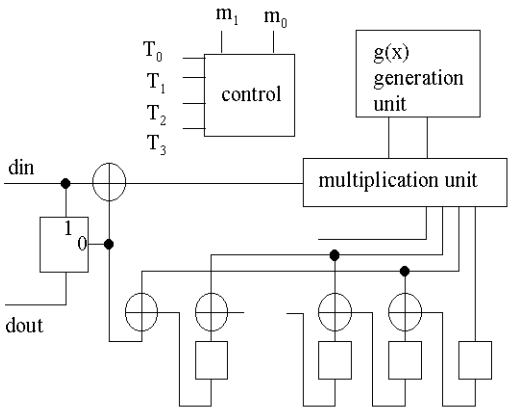

Fig. 4. Overall programmable encoder architecture

The inputs to these inner product generators are the 8 bits from Berlekamp multiplier and the constant coefficients of g(x). These coefficients are generated by combinational logic, although they could equivalently be stored in ROMs. The multiplier unit shown in Fig. 4 therefore comprises 16 of these inner product generators and the Berlekamp multiplier. The combinational logic generating g(x) aside, the overall architecture has a relatively regular structure. The complexity of this structure is further simplified by using symmetric generator polynomials. Since a Berlekamp multiplier is used in this encoder, data enters and leaves the circuit represented in the dual basis.

6 BLMS IN RS ENCODERS

The presented architecture is based around the design of a TBM, however it is also be possible to utilize the recently presented BLM in both fixed rate and programmable RS encoders [3] . This multiplier has a very similar structure to the TBM but when used in an RS encoder, has a different input/output format, since data enters the encoder represented in the polynomial basis and leaves in the dual basis. The constant values are also represented in the dual basis and since there are 2^m-1 dual bases to the polynomial basis for any GF(2^m), it is possible to choose the dual basis that reduces the complexity of the constant multiplications required. This may prove significant in applications such as space communications where reducing the complexity of the encoder is all important. In addition, the longest delay path of the LFSR required in a BLM is always one XOR gate whereas with the TBM this delay path is $\geq$ one XOR gate, depending on the Hamming weight of the irreducible polynomial for the field. Hence in those cases when f(x) is not an irreducible trinomial - GF(2^8) for example - the required LFSR can be clocked quicker than the equivalent LFSR in the TBM.

7 CONCLUSIONS

In this paper, a programmable (n, k) RS encoder has been described. This encoder is bit-serial in operation and utilizes a programmable Berlekamp multiplier. This encoder can be programmed for various values of both n and k. In contrast to a previously described rate-adaptive encoder [2], the presented architecture uses combinational logic to derive the generator polynomial for the code. This is considered to offer a number of advantages. For example successive codewords can be encoded using RS codes with widely differing parameters with no extra delay, the presented method has lower hardware requirements and also allows for the parameter n to be programmed in addition to k. It is to be noted however that by using the methods stated in Section 3, the approach described in [2] can also be modified to accommodate programmable n, albeit at the expense of a significant increase in complexity, as the required bidirectional LFSRs will have to be programmable.

To further simplify the structure of the encoder it is suggested that ROMs be used to store the values of g(x) for each n and k. Furthermore, it may not prove necessary to accommodate such a large range of choices of n and k. For greater details of the encoder architecture, the reader is encouraged to refer to the schematics given in [4]. The use of BLMs in RS encoders has also briefly been considered and shown to demonstrate a number of potential advantages over TBMs in bit-serial RS encoders.

REFERENCES

[1] E.R.Berlekamp, “Bit-serial RS Encoders”, IEEE Trans. Info. Theory, **28**, Nov. 1982, pp. 869-874.

[2] M.A.Hasan & V.K.Bhargava, “Architecture for a low complexity rate-adaptive RS encoder”, IEEE Trans. Computers, **44**, July 1995, pp. 938-942.

[3] S.T.J.Fenn, M.Benaissa & D.Taylor, “Bit-serial Berlekamp-like multipliers for GF(2^m)”, Electronics Letters, **31**, 26 Oct. 1995, pp. 1893-1894.

[4] J. Luty, “The design of finite field multipliers and their application in RS encoders”, MSc Thesis, The University of Huddersfield, U.K.