# Architecture For A Low Complexity Rate-Adaptive Reed-Solomon Encoder [†]

**M. A. Hasan**
Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario
Canada N2L 3G1

and

**V. K. Bhargava**
Department of Electrical and Computer Engineering
University of Victoria, Victoria, B.C.
Canada V8W 3P6

**Abstract**

Multiple error-correcting Reed-Solomon (RS) codes have many practical applications. The complexity of an RS encoder depends on multiplications in the finite field over which the code is defined. In this article, we consider a triangular basis for representing the field elements, and present architecture for a rate adaptive RS encoder using a triangular basis multiplication algorithm. The architecture supports pipeline and bit-serial operations, and has a low circuit complexity.

**Index Terms:** Bit-serial structure, finite field multiplication, generator polynomial, rate-adaptive Reed-Solomon encoder, triangular basis.

## I. INTRODUCTION

A Reed-Solomon (RS) code is a multiple-error-correcting code. It is also a *maximum distance* code: no code having the same block length and code rate can have a larger minimum distance than an RS code [2]. The multiple-error-correcting capability of RS codes has been used in many practical applications. Examples of important practical applications include magnetic and optical storage systems, space and mobile communications, etc.

An $(n,\ k)$ RS code with symbols from a finite field consists of $k$ data symbols and $n-k$ parity symbols, and can correct a maximum of $t = \lfloor (n-k)/2 \rfloor$ symbols in error. In this article, we consider the field GF$(2^m)$. Let the data symbols be represented by the coefficients of $D(x) \triangleq \sum_{i=0}^{k-1} D_i x^i$.

[†]Appeared in *IEEE Trans. Computers,* pp. 938-942, July 1995. This work was supported, in part, by the University of Waterloo under a start-up grant and, in part, by the Canadian Institute for Telecommunications Research under the NCE program. A preliminary version of part of this article was presented at the 16th Biennial Symposium on Communications, Kingston, Ontario, Canada [1].

Then the parity symbols are the coefficients of $x^{n-k}D(x)$ mod $G(x)$ where $G(x)$ is the generator polynomial of the code and is defined as follows:

$$G(x) \triangleq \sum_{i=0}^{n-k} G_i x^i = \prod_{i=0}^{n-k-1} \left(x - \gamma^{b+i}\right). \tag{1}$$

In (1), $\gamma$ is a primitive $n$th root of unity in GF$(2^m)$ and $b$ is an integer constant.

The calculation of the parity symbols requires multiplications in the field GF$(2^m)$. The complexity of an RS encoder depends mainly on the associated multiplication circuitry. A parallel-type multiplier usually requires $O(m^2)$ two-input AND gates and XOR gates; however, the circuit complexity of a bit-serial multiplier is only $O(m)$ [3], [4], [5]. Using Berlekamp's bit serial multiplication algorithm an RS encoder has been developed by Hsu *et al.* [6]. The encoder results in RS codes with a fixed error correcting capability. A designer usually considers the worst possible channel condition to determine this capability. However, in many applications the channel remains in its worst state for only a small fraction of the total time of use. Thus it is advantageous to have an option to vary the number of redundant symbols.

In this article, we present a rate adaptive RS encoder using a triangular basis multiplication algorithm. The encoder can provide code rates from unity to a minimum value determined by the associated hardware circuitry. The encoder is of low circuit complexity and suitable for VLSI implementation.

The organization of this article is as follows. In Section II, we consider triangular basis and multiplication in GF$(2^m)$. In Section III, a scheme for forming generator polynomials of RS codes is given and a structure for a rate-adaptive RS encoder is presented. Finally, in Section IV, concluding remarks are made.

## II. TRIANGULAR BASIS AND MULTIPLICATION IN GF$(2^m)$

### A. Triangular Basis

Let $f(x) = \sum_{i=0}^{m} f_i x^i$ be a monic irreducible polynomial of degree $m$ over GF(2) so that GF$(2^m)$=GF(2)$[x]/f(x)$.

Then GF$(2^m)$ can be viewed as a vector space of dimension $m$ over GF(2) and any ordered basis $A = (A_0, A_1, \cdots, A_{m-1})$ is a vector $\underline{A}$ over GF$(2^m)$ (or an $m \times m$ matrix over GF(2) if each $A_i$ is a column vector). $\underline{B} = \underline{A}M$ is the vector of another basis $B$ iff the basis transformation matrix $M$ is an $m \times m$ non-singular matrix over GF(2). Any $c \in$GF$(2^m)$ can be written uniquely as $c = \underline{c}_A \underline{A}^T$, with $\underline{c}_A \in$GF$(2^m)$ being the vector of *coordinates* of $c$ with respect to $A$. Clearly, $\underline{c}_A = \underline{c}_B M^T$ and $\underline{c}_B = \underline{c}_A (M^T)^{-1}$.

Let $M = [M_{i,j}]$ with

$$M_{i,j} = \begin{cases} f_{i+j+1} & 0 \le i+j \le m-1 \\ 0 & m \le i+j \le 2m-2 \end{cases} \tag{2}$$

which is a Hankel matrix with constants on its back-diagonals. Then $M^{-1}$ is a matrix with entries

$$\left(M^{-1}\right)_{i,j} = \begin{cases} 0 & 0 \le i+j \le m-2 \\ h_{i+j+1-m} & m-1 \le i+j \le 2m-2, \end{cases} \tag{3}$$

with $h_0 = 1$ and $h_k = \sum_{i=0}^{k-1} f_{m-k+i} h_i$. Given this particular choice of $M$, $B$ is referred to as the triangular basis with respect to $A$.

The advantage of the triangular basis is that the change of coordinates between $A$ and $B$ can be easily implemented by shift register techniques using:

$$(\underline{c}_A)_j = \sum_{i=0}^{m-j-1} f_{i+j+1} (\underline{c}_B)_i \quad 0 \le j \le m-1 \tag{4}$$

and

$$(\underline{c}_B)_k = \begin{cases} (\underline{c}_A)_{m-1-k} & k = m-1 \\ (\underline{c}_A)_{m-1-k} + \sum\limits_{i=0}^{k-1} f_{m-k+i} (\underline{c}_B)_i & 0 \le k \le m-2. \end{cases} \tag{5}$$

The realization of (4) requires an $m-1$ stage linear *feed-forward* shift register while that of (5) requires an $m-1$ stage linear *feed-back* shift register.

Consider the canonical ordered basis $A = \{1,\ \alpha,\ \alpha^2, \cdots, \alpha^{m-1}\}$ where $\alpha$ satisfies $f(\alpha) = 0$. Then, field element representation with respect to the corresponding triangular basis has the feature:

$\underline{\alpha}_B^j = (t_{j+1},\ t_{j+2},\ \cdots,\ t_{j+m})$ with $t_0 = 1$, $t_1 = \cdots = t_{m-1} = 0$, and $t_k = \sum\limits_{i=1}^{k-1} f_{m-i} t_{k-i}$ which can be realized using a conventional $m$ stage linear feed-back shift register (LFSR) with $f(x)$ as its *connection* polynomial.

## B. Multiplication in GF($2^m$)

**Proposition 1** Let $u,\ v\ \in$ GF($2^m$) and $U$ be an $m \times m$ matrix with row vectors $\underline{u}_B$, $\underline{u\alpha}_B$, $\cdots$, $\underline{u\alpha}_B^{m-1}$. Then $U$ is a Hankel matrix and $U\underline{v}_A^T = (\underline{uv}_B)^T$.

Again, the calculation of the Hankel matrix constants can be implemented using an $m$ stage LFSR initially loaded with $\underline{u}_B$, since

$$U_{i,j} = U_{i+j} = \begin{cases} \left(\underline{u\alpha}_B^j\right)_{i+k} & 0 \le i+k \le m-1 \\ \sum\limits_{l=1}^{m} U_{i+k-l} f_{m-l} & m \le i+k \le 2m-2. \end{cases} \tag{6}$$

Given $\underline{u}_A$ and $\underline{v}_A$, the product $uv$ can be computed using the following steps which apply triangular basis and require $O(m^2)$ arithmetic operations over the prime field GF(2).

**Step 1.** Transform $\underline{u}_A$ to $\underline{u}_B$.

**Step 2.** Generate the rows of the Hankel matrix $U$.

**Step 3.** Perform the matrix-vector multiplication $U\underline{v}_A^T$ to obtain $(\underline{uv}_B)^T$.

**Step 4.** Transform $\underline{uv}_B$ to $\underline{uv}_A$.

**Example 1** Let $f(x)$ be $1 + x^3 + x^4$, $\underline{u}_A = (1,1,1,1)$ and $\underline{v}_A = (0,1,1,1)$. Then $\underline{u}_B = (1,0,1,0)$ which is the 0th row of the corresponding Hankel matrix $U$. Using (6), $U_4 = 1$, $U_5 = 1$ and $U_6 = 0$. Then $U\underline{v}_A^T = (\underline{uv}_B)^T = (1,0,0,0)^T$ and a back transformation using (4) results in $\underline{uv}_A = (0,0,1,1)$.

Note that $\underline{u}_A$ is required at Step 1 and $\underline{v}_A$ at Step 3. This feature may reduce hardware cost in applications where several (say, $s$) multiplications are to be performed simultaneously with one factor common to all the multiplications; because $s$ of each of Steps 3 and 4, but only one of each of Steps 1 and 2 are to be implemented. Furthermore, at the output of Step 3 each product is available with respect to $B$. If these $s$ products are to be added to obtain a single result, the addition operation can be performed with respect to $B$ at Step 3 and then the resultant sum can be transformed to the $A$ basis coordinates at Step 4. Consequently, the total number of Step 4 reduces from $s$ to 1. This idea can be applied to develop area efficient architecture for VLSI design of RS encoder.

## C. Triangular and Dual Bases

The following theorem gives a condition under which the canonical and its triangular bases are dual to each other. The concept of dual basis has been efficiently used to obtain finite field multipliers [3].

**Theorem 1** Let $A$ be the canonical basis, and $B$, its triangular basis as defined above. Then $A$ and $B$ are dual bases iff $m$ is even and

$$f_i = \begin{cases} 1 & i = 1 \\ 0 & \text{odd } i \text{ and } 1 < i < m. \end{cases} \tag{7}$$

Proof: $A$ and $B$ are dual bases iff the matrix $[\mathrm{Tr}(A_i B_j)] = I$ where Tr is the trace function and $I$ is the $m \times m$ identity matrix. Since $A$ is the canonical basis and $\underline{B} = \underline{A}M$, $\mathrm{Tr}(\alpha^i B_j) = \mathrm{Tr}(\alpha^i \sum_k \alpha^k M_{k,j}) = \sum_k M_{k,j}\mathrm{Tr}(\alpha^{i+k})$, so $I = [\mathrm{Tr}(A_i B_j)] = [\mathrm{Tr}(A_l)]M$, which means that $[\mathrm{Tr}(A_l)] = M^{-1}$. This in turn results in:

$$\mathrm{Tr}(\alpha^l) = \begin{cases} 0 & 0 \le l \le m-2 \\ h_{l+1-m} & m-1 \le l \le 2m-2. \end{cases}$$

$m \equiv \mathrm{Tr}(\alpha^0) = 0$ means $m$ is even. $\mathrm{Tr}(\alpha^{2l}) = \mathrm{Tr}(\alpha^l) = 0$ for $0 \le l \le m-2$ gives $h_{2t+1-m} = 0$ for $m \le 2t < 2m-2$, and $h_{m-1} = h_0 = 1$ which lead to Equation (7).

Examples of irreducible polynomials, for which the canonical and its triangular bases are dual, are $1+x+x^4$, $1+x+x^6$, $1+x+x^2+x^6+x^{26}$, etc. To obtain a low complexity basis transformation circuitry, if an irreducible *trinomial* is chosen then the type of trinomial which results such canonical and triangular bases is $1+x+x^m$. This type of irreducible trinomial does exist for certain $m$ [7] and the corresponding triangular basis is $\{\alpha^{-1},\ \alpha^{m-2},\ \alpha^{m-3},\ \cdots,\ 1\}$.

## III. A RATE-ADAPTIVE RS ENCODER

### A. Basic Structure

A rate-adaptive RS encoder can be seen as a fixed-rate RS encoder capable of changing generator polynomials to provide variable redundancy in codewords. A structure for obtaining different generator polynomials will be discussed in the next subsection. Here we consider a fixed-rate RS encoder which supports pipeline and bit-serial operations. Fig. 1 shows such an encoder using the triangular basis multiplication algorithm. More on its structure, complexity and comparison can be found in [1]. As an informal description of how does it work, note that the data sequence $\{D_0,\ D_1,\ \cdots, D_{k-1}\}$ is shifted bit-by-bit into the encoder circuit and simultaneously into the communication channel with the switches S1 and S2 at position P1. The incoming bit stream is first buffered into the $m-1$ stage shift register before a block of $m$ bits is loaded into the LFSR. The loading operation corresponds to the formation of the 0th row of the Hankel matrix in the multiplication algorithm. The other rows are generated by the LFSR in the subsequent clock cycles. Module $M_i$ $(0 \leq i \leq 2t-1)$ performs an inner product of two input vectors to implement Step 3 of the algorithm. As soon as the complete data sequence has entered the encoder, both switches are placed at position P2. Over the next $2tm$ clock cycles, the parity check symbols are transmitted by shifting the contents of the registers.

The above encoder has a fixed number of redundant symbols $r = n-k$ which is usually sufficient to meet the worst possible channel condition. In many applications, the channel, however, remains in its worst state for only a small fraction of the total time of use. So an option to vary the number of redundant symbols may achieve a higher code rate during other times. In the following discussion, we do not address the issue of how the system detects the need for a change in redundancy, but we do present an encoder architecture for which such a change can be efficiently implemented.

## B. Formation of Generator Polynomials

**Algorithm**

Denote $r_i \in \{0, 1, 2, \cdots, r\}$ and $G(i, x)$ as the redundancy and generator polynomial of the codeword $C^{(i)}$ $(i = 0,\ 1,\ \cdots)$. Let the encoder start *forming* $G(i, x)$ and then *transmitting* $C^{(i)}$ at the time instances $F_i$ and $T_i$, respectively. If the encoder requires $w_i$ clock cycles to form $G(i, x)$, then for a pipeline bit-serial operation $T_{i+1} - T_i = mn$ clock cycles and $mn \geq T_{i+1} - F_{i+1} \geq w_{i+1}$. The value of $w_{i+1}$ depends on how the encoder forms $G(i+1, x)$. The latter, in this article, is formed from $G(i, x)$ with the assumption that $\Delta r_i \overset{\Delta}{=} r_{i+1} - r_i$ be made available to the encoder prior to $F_{i+1}$.

To generate $G(i+1, x)$, first consider the case where the maximum difference between the number of redundant symbols of two successive codewords is unity, i.e.,

$$|\Delta r_i|_{max} = |r_{i+1} - r_i|_{max} = 1. \tag{8}$$

For the sake of simplicity we take $b = 1$ and $\gamma = \alpha$ in (1), where $\alpha$ is a root of the polynomial defining the field. Because $G(i, x) \overset{\Delta}{=} \sum_{j=0}^{r_i} G_j(i) x^j = \prod_{j=0}^{r_i - 1} (x + \alpha^{j+1})$ where $G_j(i)$ is the $j$th order coefficient of $G(i, x)$,

$$G(i+1, x) \overset{\Delta}{=} \sum_{j=0}^{r_{i+1}} G_j(i+1) x^j = \begin{cases} (x + \alpha^{r_i+1})\, G(i, x) & \text{if } \Delta r_i = 1 \\ G(i, x) & \text{if } \Delta r_i = 0 \\ (x + \alpha^{r_i})^{-1}\, G(i, x) & \text{if } \Delta r_i = -1. \end{cases} \tag{9}$$

For $\Delta r_i = 1$,

$$G_j(i+1) = \begin{cases} G_{r_i}(i) & j = r_{i+1} \\ G_{j-1}(i) + G_j(i) \alpha^{r_i+1} & r_{i+1} > j > 0 \\ G_0(i) \alpha^{r_i+1} & j = 0, \end{cases} \tag{10}$$

and for $\Delta r_i = -1$,

$$G_j(i+1) = \begin{cases} G_{r_i}(i) & j = r_{i+1} \\ G_{j+1}(i) + G_{j+1}(i+1)\alpha^{r_i} & r_{i+1} > j \geq 0. \end{cases} \tag{11}$$

Equations (10) and (11) give a sequential scheme to obtain $G(i+1,x)$. When $\Delta r_i = 1$, the coefficients of $G(i+1,x)$ are obtained using only the coefficients of $G(i,x)$. On the other hand, when $\Delta r_i = -1$, the coefficients of $G(i+1,x)$ are obtained using the next higher order coefficients of both $G(i+1,x)$ and $G(i,x)$. Whether $\Delta r_i = 1$ or $-1$, a maximum of one finite field multiplication and one addition is involved in obtaining a coefficient of $G(i+1,x)$.

**Structure**
A structure for obtaining $G(i+1,x)$ is shown in Fig. 2. It is divided into three main units- Root Generation Unit, Coefficient Updating Unit and Coefficient Waiting Unit which are briefly described below.

Root Generation Unit: It consists of a bidirectional LFSR as shown in Fig. 3. The LFSR contents are the canonical basis coordinates of one element of the set $\{\alpha^0, \alpha^1, \cdots, \alpha^r\}$. At $i = 0$, the LFSR is initialized to $\underline{\alpha}_A^{r_0}$ where $r_0$ is the redundancy initially set to the encoder. If the LFSR contains $\underline{v}_A$, then after one shift with $L = 1$ (resp. 0), it contains $\underline{v\alpha}_A$ (resp. $\underline{u\alpha}_A^{-1}$). The value of $L$ is determined as follows:

$$L = \begin{cases} 1 & \text{if } \Delta r_i > 0 \\ 0 & \text{if } \Delta r_i < 0. \end{cases}$$

If $\Delta r_i = 0$, the LFSR is not clocked. Also if $\Delta r_i > 0$ (resp. $\Delta r_i < 0$) the LFSR is clocked at $F_{i+1}$ [resp. $m(r+1)$ cycles after $F_{i+1}$].

Coefficient Updating Unit: Depending on the position of the switch, this unit performs the multiplication and addition operations of (10) and (11) in bit-serial fashion. It consists of an adder and one pipeline bit-serial triangular basis multiplier with a *latency* of $m$ clock cycles. The inputs to the multiplier are the canonical basis coordinates of the generator polynomial coefficients and $\underline{\alpha}_A^{r_{i+1}}$ generated by the Root Generation Unit.

Coefficient Waiting Unit: A copy of the currently used generator polynomial coefficients waits in this unit to be updated, if required. At the time instant $F_{i+1}$, the contents of $\mathrm{R}_{r-j}$ are the canonical basis coordinates of $G_{r_i-j}(i)$ for $j = 0, 1, \cdots, r_i$ and $i = 0, 1, 2, \cdots$. Registers $\mathrm{R}_0$, $\mathrm{R}_1$, $\cdots$, $\mathrm{R}_r$ and the Coefficient Updating Unit form a unidirectional ring. To obtain $G(i+1,x)$, the register contents are circulated once along the ring in a bit-serial manner starting one clock cycle after $F_{i+1}$. The circulation takes $m(r+1)$ clock cycles. The new coefficients wait in this unit until a next update is required (the earliest update can start at $F_{i+2}$ if $\Delta r_{i+1} \neq 0$), and prior to the time instant $T_{i+1}$ the outputs of $\mathrm{R}_0$, $\mathrm{R}_1$, $\cdots$, $\mathrm{R}_{r-1}$ are down loaded to the buffers which hold the generator polynomial coefficients.

Depending on the value of $\Delta r_i$, the structure operates in one of the following three modes.

**Mode 1.** ($\Delta r_i > 0$) At the instant $F_{i+1}$, the switch is placed at position P3 and the output of the Root Generation Unit is updated from $\alpha^{r_i}$ to $\alpha^{r_{i+1}} = \alpha^{r_i} \cdot \alpha$ which is done in one clock cycle. From the next cycle, the contents of the coefficient registers $\mathrm{R}_0$, $\mathrm{R}_1$, $\cdots$, $\mathrm{R}_r$ are shifted $m(r+1)$ times to obtain $G(i+1,x) = (x + \alpha^{r_{i+1}})G(i,x)$.

**Mode 2.** ($\Delta r_i = 0$) The coefficient registers are not shifted and the output of the Root Generation Unit is left unchanged.

**Mode 3.** ($\Delta r_i < 0$) At the instant $F_{i+1}$, the switch is placed at position P4. From the next cycle, the contents of the coefficient registers are shifted $m(r+1)$ times to obtain $G(i+1,x) = (x+\alpha^{r_i})^{-1}G(i,x)$. The output of the Root Generation Unit is then updated from $\alpha^{r_i}$ to $\alpha^{r_{i+1}} = \alpha^{r_i} \cdot \alpha^{-1}$.

Among the three modes, Mode 3 requires the maximum number of clock cycles for its execution and determines $w_{i+1}$ which is $m(r+1)+1$ for $i = 0,\ 1,\ \cdots$.

## C. Comments

Although only $|\Delta r_i|_{max} = 1$ has been considered so far, the structure can also be used for higher values of $|\Delta r_i|_{max}$, in which case if $\Delta r_i$ is greater (resp. less) than zero, then the structure of Fig. 2 operates in Mode 1 (resp. Mode 3) for $\Delta r_i$ times, and $w_{i+1} = (m(r+1)+1)|\Delta r_i|_{max}$. Because of the constraint from pipeline operation, the value of $|\Delta r_i|_{max}$ is, however, bounded as follows:

$$1 \le |\Delta r_i|_{max} \le \min \left\{ \left\lfloor \frac{mn}{m(r+1)+1} \right\rfloor, r \right\}. \tag{12}$$

Given that the redundancy in a codeword does not need to be more than $r$, (12) enables the encoder to add/remove the required number of redundant symbols without sacrificing its pipeline operation.

Instead of the structure presented here, a straightforward approach to obtain the coefficients of all possible generator polynomials[1] would be to store them in a ROM and down load the appropriate coefficients into the buffers holding the generator polynomial coefficients. This approach requires a total ROM size of about $\frac{1}{2}mr^2$ bits. On the other hand, the structure presented here mainly requires $m(r+5)-2$ flip flops, $m+4(W_f-2)$ two-input XOR gates and $m$ two-input AND gates. Since $m$ is usually less than 10 and the circuit complexity of the proposed structure is only linearly proportional to $r$, for applications where the value of $r$ is even moderately large (say 10), the use of the structure for the generator polynomial formation, in conjunction with the fixed-rate RS encoder as shown in Fig. 1, would result in an area efficient rate-adaptive RS encoder.

## IV. CONCLUSIONS

In this article, a triangular basis has been considered. Transformations from a primal basis to its triangular basis and vise versa can be implemented using shift registers. A class of irreducible polynomials has been given, for which the canonical and its triangular bases are dual. A structure for rate-adaptive RS encoding over GF($2^m$) is presented. The structure can be easily extended for RS codes with different values of $m$ and $t$. Also, if a different primitive polynomial is chosen to define the field GF($2^m$), then the proposed structure can be readily modified to incorporate this change without any pre-algebraic manipulation. Moreover, the encoder is of low circuit complexity; thus it is well suited for use in applications where silicon area is a prime concern such as satellites and pocket size wireless terminals.

## Acknowledgment

The authors would like to thank Reviewer G for improving the article, and also Profs. I. F. Blake and M. Z. Wang for their useful comments during the manuscript preparation.

[1]For a fixed pair of $b$ and $\gamma$ in (1) there are $r+1$ generator polynomials.

## References

[1] M. A. Hasan and V. K. Bhargava, "A VLSI Architecture for A Low Complexity Rate-Adaptive Reed-Solomon Encoder," *Proc. 16th Biennial Symposium on Communications, Kingston, Ontario.*, pp. 331–334, May 1992.

[2] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications.* Cambridge: Cambridge University Press, 1986.

[3] E. R. Berlekamp, "Bit-Serial Reed-Solomon Encoder," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 869–874, Nov. 1982.

[4] M. Z. Wang and I. F. Blake, "Bit Serial Multiplication in Finite Fields," *SIAM J. Disc. Math.*, vol. 3, pp. 140–148, Feb. 1990.

[5] M. A. Hasan and V. K. Bhargava, "Division and Bit-Serial Multiplication over GF($q^m$)," *IEE Proc. Part E*, vol. 139, pp. 230–236, May 1992.

[6] I.-S. Hsu, I. S. Reed, T. K. Truong, K. Wang, C. S. Yeh, and L. J. Deutsch, "The VLSI Implementation of a Reed-Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm," *IEEE Trans. Comput.*, vol. C-33, pp. 906–911, Oct. 1984.

[7] N. Zierler and J. Brilluart, "On Primitive Trinomials (Mod 2)," *Inform. and Contr.*, vol. 13, pp. 541–554, 1968.

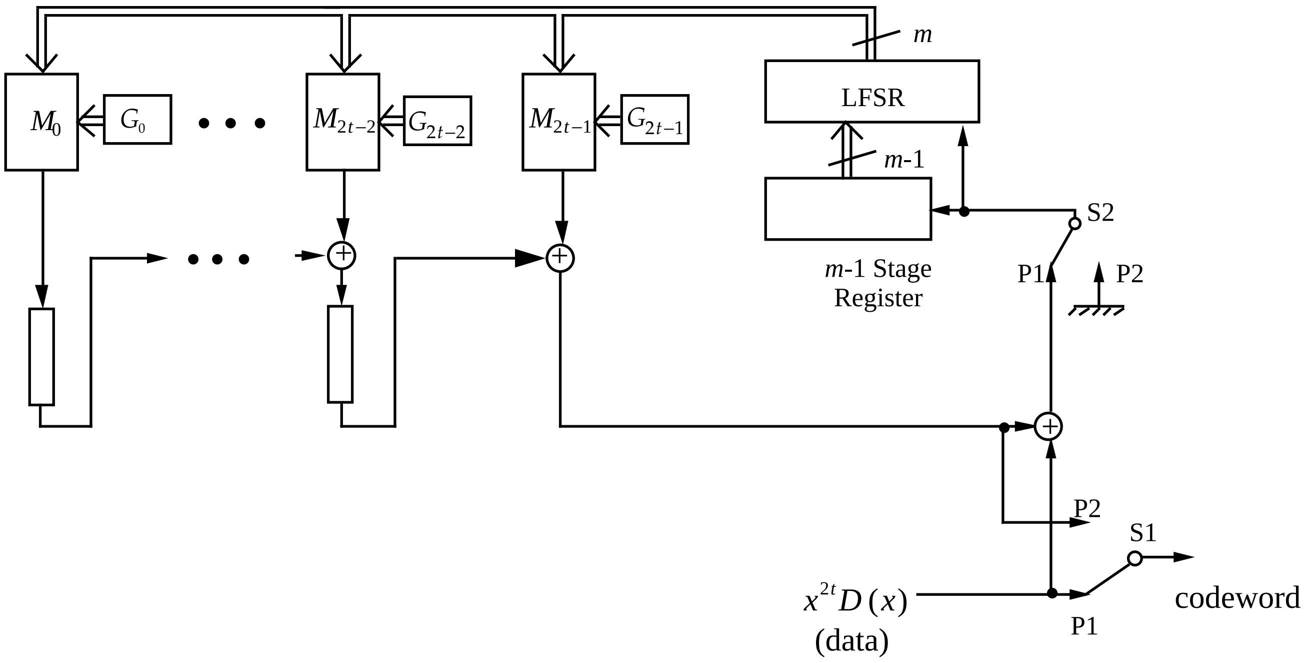


Fig. 1. A fixed-rate pipeline bit-serial RS encoder.

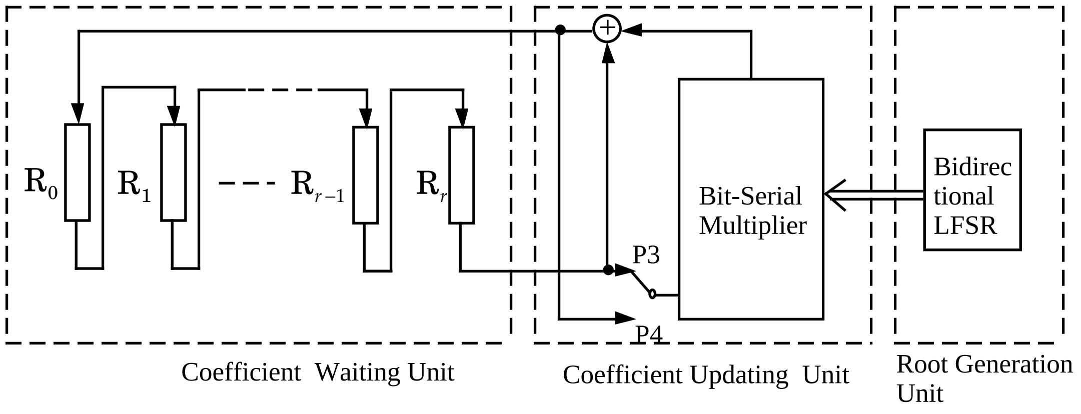
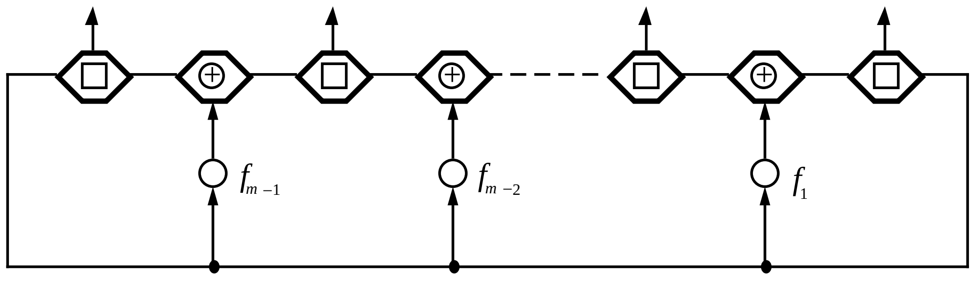
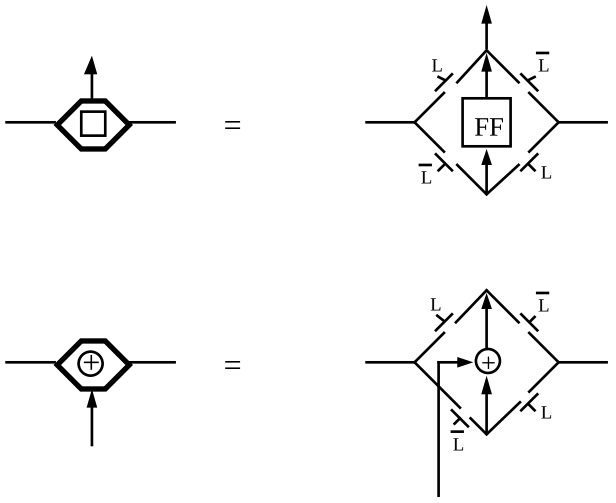


Fig. 2. Structure for the formation of generator polynomials.





(An '1' at the pass gate's input closes the circuit.)

Fig. 3. Root Generation unit using bidirectional LFSR.