$GF(2^m)$ Multiplication and Division Over the Dual Basis

Sebastian T.J. Fenn, Mohammed Benaissa, and David Taylor

Abstract—In this paper an algorithm for $GF(2^m)$ multiplication/division is presented and a new, more generalized definition of duality is proposed. From these the bit-serial Berlekamp multiplier is derived and shown to be a specific case of a more general class of multipliers. Furthermore, it is shown that hardware efficient, bit-parallel dual basis multipliers can also be designed. These multipliers have a regular structure, are easily extended to different $GF(2^m)$ and hence suitable for VLSI implementations. As in the bit-serial case these bit-parallel multipliers can also be hardwired to carry out constant multiplication. These constant multipliers have reduced hardware requirements and are also simple to design. In addition, the multiplication/division algorithm also allows a bit-serial systolic finite field divider to be designed. This divider is modular, independent of the defining irreducible polynomial for the field, easily expanded to different $GF(2^m)$ and its longest delay path is independent of m.

Index Terms—Dual basis, finite field division, finite field multiplication, irreducible polynomials, Reed-Solomon codecs, systolic arrays, VLSI.

1 Introduction

Finite field arithmetic is fundamental to the implementation of Reed-Solomon (RS) codes [1] and certain cryptographic systems [2]. The most important finite field arithmetic operation is multiplication, and a considerable amount of work has been carried out defining finite field multipliers suitable for implementation in VLSI [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. Of these the most suitable for use in RS codecs appears to be the Berlekamp [3] and Massey-Omura bit-serial multipliers [9], and the Mastrovito bit-parallel multiplier [6], [7].

The bit-parallel Mastrovito multiplier for $GF(2^m)$ operates over the polynomial basis and comprises m identical inner product modules and one further module, the design of which is dependent on the defining irreducible polynomial for the field [6]. The bit-parallel Massey-Omura multiplier had previously been regarded as the most suitable bit-parallel multiplier for implementation in VLSI because it comprises m identical modules. However, it has been shown that the Mastrovito multiplier requires only around half the gates the Massey-Omura multiplier needs to implement the necessary combinational logic [6]. Given the multiplier's high regularity and low hardware requirements it is therefore considered highly suited to implementation in RS codecs.

The bit-serial Massey-Omura multiplier for $GF(2^m)$ operates over the normal basis and consists of $2m$ register elements and at least $(2m - 1)$ AND gates and $(2m - 2)$ XOR gates [13]. The bit-serial Berlekamp multiplier, however, operates over both the polynomial basis and the dual basis and has lower hardware requirements than the Massey-Omura multiplier [8]. That is, a bit-serial Berlekamp multiplier for $GF(2^m)$ requires $2m$ register elements, m AND gates, and $(m + H(pp) - 3)$ XOR gates where $H(pp)$ denotes the Hamming weight of the defining irreducible polynomial for the field. The bit-serial Berlekamp multiplier also has the advantage that it can be hardwired to carry out constant multiplication, further reducing its hardware requirements and allowing one such multiplier to yield many products [15].

Crucial to the operation of the Berlekamp multiplier is the trace function and the concept of duality. The trace function is a linear function from $GF(p^m)$ to $GF(p)$, where the trace of $\beta \in GF(p^m)$, $tr(\beta)$, is defined to be

$$tr(\beta) = \sum_{i=0}^{m-1} \beta^{p^i}.$$

Two bases are then said to be dual to one another if a set of conditions are satisfied by the trace values of the basis elements [3].

In this paper we extend the idea of the trace function to encompass any general linear function. Using a linear function and a new definition of duality, a finite field multiplication/division algorithm is presented and from this the bit-serial Berlekamp multiplier is derived. This algorithm also allows us to derive a bit-parallel dual basis multiplier which has a number of appealing characteristics. For example the bit-parallel multiplier is regular, easily expandible to different $GF(2^m)$ and hardware efficient. It can also be easily hardwired to perform constant multiplication.

This same algorithm also permits a bit-serial systolic divider to be designed. Finite field division is required in a number of RS decoding techniques such as time domain decoding [16] and the standard Berlekamp-Massey algorithm for solving the key equation [17]. The presented divider is modular, easily expanded to different $GF(2^m)$ and

• *The authors are with the Department of Electrical and Electronic Engineering, the University of Huddersfield, Queensgate, Huddersfield, HD1 3DH, U.K. E-mail: s-fenn@eng.hud.ac.uk.*

Manuscript revised Feb. 5, 1995.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number C96007.

the longest delay path is independent of m. These characteristics are in contrast to those of the inverters presented in [18], [19], [20]. Hence this divider is highly suited to VLSI implementation.

By selecting suitable dual bases, the problem of these operators functioning over two different bases can be largely circumvented. Dual bases can be chosen which require little or no extra hardware to convert them to or from the polynomial basis. These operators can, therefore, be utilized throughout RS encoding and decoding circuits and since dual basis multipliers have particularly low hardware requirements, this results in the overall RS codec being particularly hardware efficient. This, in combination with the fact that there also exist efficient dual basis inverters/dividers, means that the dual basis is an appropriate basis for RS codecs to operate over.

2 Mathematical Background

It is assumed throughout this paper that the reader is familiar with the basic theory of finite fields, for more details see for example [21].

Let F_{p^m} denote the set of all linear functions $f: GF(p^m) \rightarrow GF(p)$. The trace function is a well known example of such a linear function and has previously been exploited to produce finite field multipliers and to give results relating to the solution of quadratic equations over a finite field [3], [17]. However, there are a number of other available linear functions and it is frequently more convenient to employ one of these rather than the trace function.

THEOREM 1. *Let $\{\lambda_i\}$ be a basis for $GF(p^m)$ and let $z \in GF(p^m)$ be represented by*

$$z = \sum_{i=0}^{m-1} z_i \lambda_i,$$

for $z_i \in GF(p)$. Then there are p^m linear functions $f \in F_{p^m}$ and furthermore, these functions are of the form

$$f(z) = \sum_{i=0}^{m-1} x_i z_i \quad \forall z \in GF(p^m) \tag{1}$$

where $x_i \in GF(p)$ and the addition is modulo p.

To prove Theorem 1 the following lemma is required.

LEMMA 1. ([21]) *Any $f \in F_{p^m}$ can be considered to be of the form*

$$f(z) = tr(\beta z) \quad \forall z \in GF(p^m) \tag{2}$$

for some $\beta \in GF(2^m)$. Consequently, there are p^m linear functions $f \in F_{p^m}$. (Although the all zero function is of no practical interest).

PROOF OF THEOREM 1. The functions defined in (1) are obviously linear and there will certainly be p^m of them. But from Lemma 1, there are only p^m linear functions $f \in F_{p^m}$ and so the functions f defined in (1) must be all the linear functions $f \in F_{p^m}$, as required. □

It is often convenient to take the basis $\{\lambda_i\}$ to be the polynomial basis $\{1, \alpha, \ldots, \alpha^{m-1}\}$ where α is a root of the defining polynomial for $GF(p^m)$. So, for example, consider $GF(2^4)$ with $p(x) = x^4 + x + 1$ as the defining irreducible polynomial for the field. Taking α to be a root of $p(x)$, any $z \in GF(2^4)$ can be represented by

$$z = \sum_{i=0}^{3} z_i \alpha^i$$

Then from Theorem 1, all the linear functions $f \in F_{2^4}$ will be of the form

$$f(z) = \sum_{i=0}^{3} z_i x_i$$

where $x_i \in GF(2)$ and addition is modulo 2. Hence the functions $f(z) = z_0$ and $f(z) = z_0 + z_2 + z_3$ are valid functions in $f \in F_{2^4}$.

When $\{\lambda_i\}$ is taken to be the polynomial basis, functions of the form

$$f(z) = z_i \quad (i \in 0, 1 \ldots m-1) \tag{3}$$

are particularly useful because the values of $f(z)$ $\forall z \in GF(p^m)$ can be obtained directly from the polynomial basis representation of the field without any further calculation. A new, more general definition of duality is now introduced.

DEFINITION 1. *Let $\{\lambda_i\}$ and $\{\mu_i\}$ be bases for $GF(p^m)$, $f \in F_{p^m}$, and $\beta \in GF(p^m)$, $\beta \neq 0$. Then the bases are said to be* dual *with respect to f and β if*

$$f(\beta \lambda_i \mu_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

In this case, $\{\lambda_i\}$ is the standard basis and $\{\mu_i\}$ is the dual basis.

THEOREM 2. *Every basis has a dual basis with respect to any nonzero $f \in F_{p^m}$ and any nonzero $\beta \in GF(p^m)$.*

PROOF. It is well known that every basis has a dual basis when f in Definition 1 is taken to be the trace function [1]. However, from Lemma 1, every $f(z)$ can be considered to be of the form $tr(\beta z)$ for some $\beta \in GF(p^m)$ and so every basis will also have a dual basis when the definition of duality is as given in Definition 1.

THEOREM 3. *Let $\{\lambda_i\}$ and $\{\mu_i\}$ be dual bases for $GF(p^m)$ with respect to f and β. Then any $z \in GF(p^m)$ can be represented in the dual basis as*

$$z = \sum_{i=0}^{m-1} f(z\beta\lambda_i)\mu_i$$

PROOF. The proof of Theorem 3 is easily obtained by exploiting the linearity of f, see Corollary 1 in [15] for example, exchanging the trace function with f.

2.1 Calculating Optimal Dual Bases

By allowing variable values of f and β in the definition of duality, instead of there being one dual basis to any given basis there are now $(p^m - 1)$ dual bases. The most

"convenient" of these dual bases can then be selected. This idea of a convenient dual basis was first introduced by Morii et al. [10] and concerns the complexity of the dual to polynomial basis conversion. It was shown in [10] that with certain $GF(2^m)$, β can be chosen so that the dual basis is merely a reordering of the polynomial basis [10]. In fact, in [10] only the case where f is the trace function was considered, however these results also hold for any general f [22]. These results are reviewed and extended in Appendix A to consider the general case of f being any suitable linear function. In addition, the optimal dual bases for $GF(2^m)$ ($m = 2, 3, \ldots, 10$) are also presented.

3 Generalized Expression for Finite Field Multiplication and Division

A general result, considered only in the context of division was first presented in [23]. From this expression we are able to derive the Berlekamp bit-serial multiplier and also to present a class of bit-parallel dual basis multipliers. Furthermore, this theorem also forms the kernel of the bit-serial systolic divider described in Section 6.

THEOREM 4 [22], [23]. *Let $a, b, c \in GF(p^m)$ such that $a = bc$. Further, let α be a root of the defining irreducible polynomial for the field, let $f \in F_{p^m}$, $\beta \in GF(p^m)$ and represent c over the polynomial basis by*

$$c = \sum_{i=0}^{m-1} c_i \alpha^i .$$

Then the following relation holds.

$$\begin{bmatrix} f(b\beta) & f(b\beta\alpha) & \cdots & f(b\beta\alpha^{m-1}) \\ f(b\beta\alpha) & f(b\beta\alpha^2) & \cdots & f(b\beta\alpha^{m}) \\ \cdots & \cdots & \cdots & \cdots \\ f(b\beta\alpha^{m-1}) & f(b\beta\alpha^{m}) & \cdots & f(b\beta\alpha^{2m-2}) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \cdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} f(a\beta) \\ f(a\beta\alpha) \\ \cdots \\ f(a\beta\alpha^{m-1}) \end{bmatrix} \quad (4)$$

4 Bit-Serial Berlekamp Multipliers

The Berlekamp multiplier [3] is known to have the lowest hardware requirements of all the available bit-serial finite field multipliers [8]. Furthermore, the Berlekamp multiplier is particularly suited to carrying out constant multiplication because one multiplier can be hardwired to generate many products. For this reason the Berlekamp multiplier has been utilized in RS encoders [3], [15]. However by utilizing the techniques first adopted in [10] it is also possible to use these multipliers throughout RS decoding circuits.

We now derive the bit-serial Berlekamp multiplier. If Theorem 4 is restricted to operation over $GF(2^m)$ and we let $b_k = f(b\beta\alpha^k)$ ($k = 0, 1, \ldots, 2m-2$) and $a_k = f(a\beta\alpha^k)$ ($k = 0, 1, \ldots, m-1$), then Theorem 4 can be restated in the form

$$\begin{bmatrix} b_0 & b_1 & \cdots & b_{m-1} \\ b_1 & b_2 & \cdots & b_m \\ \cdots & \cdots & \cdots & \cdots \\ b_{m-1} & b_m & \cdots & b_{2m-2} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \cdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ \cdots \\ a_{m-1} \end{bmatrix}. \quad (5)$$

However, taking f and β as in Definition 1, then a_k and b_k ($k = 0, 1, \ldots, m-1$) are the dual basis coefficients of a and b, respectively. Hence, if we can generate the values of b_k for ($k = m, m+1, \ldots, 2m-2$), (5) represents a dual basis multiplication algorithm. However, as is well known, these values can be obtained from a linear feedback shift register (LFSR) which is initialized with the coefficients b_k ($k = 0, 1, \ldots, m-1$) where the feedback terms correspond to the nonzero terms of the defining irreducible polynomial for the field. For example, consider b_m and let

$$p(x) = \sum_{i=0}^{m-1} p_i x^i + x^m$$

be the defining irreducible polynomial for the field. Then

$$b_m = f(b\beta\alpha^m) = f\left(b\left(\beta\sum_{j=0}^{m-1} p_j\alpha^j\right)\right) = \sum_{j=0}^{m-1} p_j f(b\beta\alpha^j) = \sum_{j=0}^{m-1} p_j b_j$$

In general, therefore, it can be seen that

$$b_{m+k} = \sum_{j=0}^{m-1} p_j b_{j+k} \quad (6)$$

where b_k ($k = 0, 1, \ldots, m-1$) are the dual basis coefficients of b and α is a root of $p(x)$.

This immediately gives rise to the bit-serial multiplication scheme originally proposed by Berlekamp. Consider Fig. 1 and (5). If the registers in Fig. 1 are initialized by $R_i = b_i$ and $S = c_i$ ($i = 0, 1, \ldots, m-1$) then from (5) the first product bit a_0 will be available on the output line immediately. The remaining values of a_i ($i = 1, 2, \ldots, m-1$) are produced by clocking the upper LFSR a further $(m-1)$ times since the values of b_{m+i} ($i = 0, 1, \ldots, m-2$) are generated by the upper LFSR.

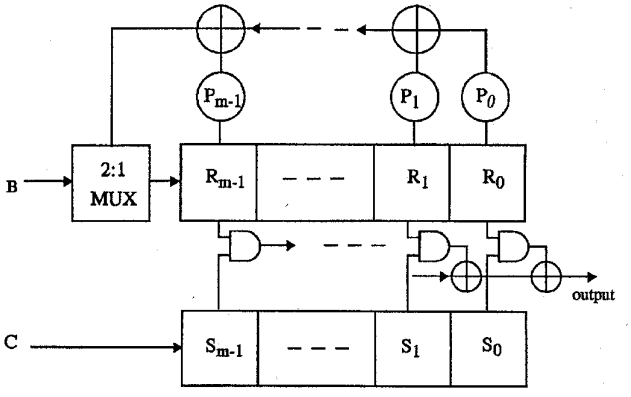


Fig. 1. Standard bit-serial Berlekamp multiplier for $GF(2^m)$.

With this multiplication scheme, both b and a are represented in the dual basis and c is represented in the polynomial basis. However, it is frequently required to enter both b and c in the dual basis, and so a dual to polynomial basis converter must be incorporated within the multiplier circuitry. As illustrated in Appendix A, the hardware required to carry out this transformation often proves trivial. For example, with $GF(2^m)$ multipliers for ($m = 2, 3, 4, 5, 6, 7, 9, 10$) for which irreducible trinomials exist, no extra hardware is required to carry out this basis conversion only a reordering of basis coefficients. With $GF(2^8)$ for which no

irreducible trinomial exists, the dual to polynomial basis transformation circuit only requires an extra two XOR gates. For a more extensive survey and discussion of bit-serial multipliers see for example, Chapter 3 in [7].

5 Bit-Parallel Dual Basis Multipliers

In some applications, it is required to adopt bit-parallel architectures rather than bit-serial ones to achieve the required performance. Although bit-serial dual basis multipliers based on the trace function have been widely employed in applications such as RS encoders [3], [15], it will often prove advantageous to employ bit-parallel dual basis multipliers, particularly in more complex circuits such as RS decoders. To this end, we now consider the design of such multipliers.

Let $a, b, c \in GF(2^m)$ such that $a = bc$ and let $\{\mu_i\}$ be the dual basis to the polynomial basis for $\beta \in GF(2^m)$ and $f \in F_{2^m}$. Representing b over the dual basis by

$$b = \sum_{i=0}^{m-1} b_i \mu_i$$

and c over the polynomial basis by

$$c = \sum_{i=0}^{m-1} c_i \alpha^i,$$

then from (5)

$$\begin{aligned} a_0 &= b_0c_0 + b_1c_1 + \ldots + b_{m-1}c_{m-1} \\ a_1 &= b_1c_0 + b_2c_1 + \ldots + b_mc_{m-1} \\ &\ldots\ \ldots\ \ldots\ \ldots \\ a_{m-1} &= b_{m-1}c_0 + b_mc_1 + \ldots + b_{2m-2}c_{m-1} \end{aligned}$$

where b_{m+k} $(k \geq 0)$ are given by (6). From these equations it can be seen that the m product bits are generated by m identical functions of the form

$$h(b, c) = b_kc_0 + b_{k+1}c_1 + \ldots + b_{k+m-1}c_{m-1} \quad (7)$$

all that changes in these functions is the value of k.

A bit-parallel dual basis multiplier for $GF(2^m)$ can, therefore, be constructed out of m $GF(2)$ inner product modules (of type A, say) that implement (7) and one module (of type B, say) that generates the b_k $(k = m, m + 1, \ldots, 2m - 2)$ from (6). An example of such a multiplier for $GF(2^4)$ is given below.

5.1 Bit-Parallel Dual Basis Multiplier for $GF(2^4)$

Let $p(x) = x^4 + x + 1$ be the defining irreducible polynomial for the field and let α be a root of $p(x)$. From (7), four type A modules are required each implementing the function

$$h(b, c) = b_kc_0 + b_{k+1}c_1 + b_{k+2}c_2 + b_{k+3}c_3.$$

This equation can be implemented by the circuit shown in Fig. 2. From $p(x) = x^4 + x + 1$ and (7) it is observed that

$$\begin{aligned} b_4 &= b_0 + b_1 \\ b_5 &= b_1 + b_2 \\ b_6 &= b_2 + b_3 \end{aligned}$$

and so the type B module can be implemented by the circuit shown in Fig. 3. The four type A modules and the type B module are then combined to form the full bit-parallel dual basis multiplier for $GF(2^4)$ shown in Fig. 4. If

$$b = \sum_{i=0}^{3} b_i \mu_i$$

is the dual basis representation of b and

$$c = \sum_{i=0}^{3} c_i \alpha^i$$

is the polynomial basis representation of c, the registers in Fig. 4 are loaded with the values $R_i = b_i$ and $S_i = c_i$ $(i = 0, 1, 2, 3)$. Once these values have been loaded into the registers, the product bits a_i $(i = 0, 1, 2, 3)$ become immediately available on the output lines. Note that in Fig. 4 both b and c enter the multiplier as represented in the dual basis and that by reordering the coefficients of c, the required dual to polynomial basis conversion is carried out. This is because as shown in Table 2 in Appendix A, if $\{1, \alpha, \alpha^2, \alpha^3\}$ is the polynomial basis then $\{1, \alpha^3, \alpha^2, \alpha\}$ forms the dual basis.

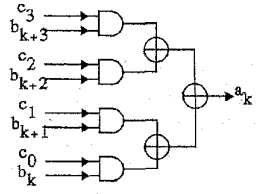


Fig. 2. Type A module for bit-parallel dual basis multiplier for $GF(2^4)$.

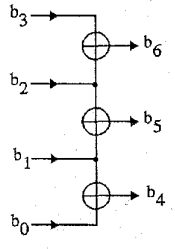


Fig. 3. Type B module for bit-parallel dual basis multiplier for $GF(2^4)$.

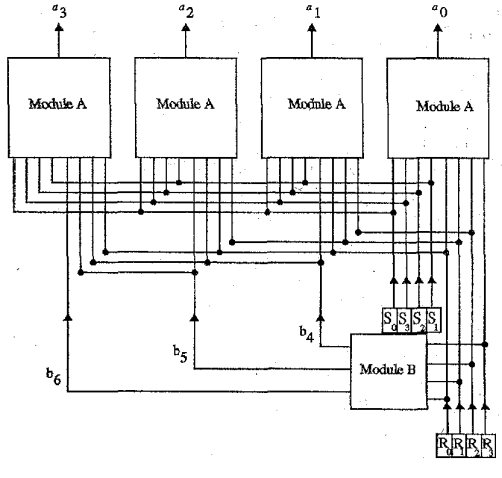


Fig. 4. Bit-parallel dual basis multiplier for $GF(2^4)$.

5.2 Complexity Analysis

It has been seen that a bit-parallel dual basis multiplier (PDBM) for $GF(2^m)$ comprises m type A modules and one type B module. The structure of the A modules is independent of $p(x)$ and depends only on m. Consequently, each type A module will consist of m AND gates and $(m-1)$ XOR gates. From (6), it can be seen that the complexity of the B type module is dependent upon m. If $H(pp)$ is the Hamming weight of $p(x)$ then each b_{m+k} in (6) can be generated by $(H(pp)-2)$ XOR gates. In total, therefore, a PDBM for $GF(2^m)$ requires m^2 AND gates and $(m-1)(H(pp)-2+m)$ XOR gates.

A bit-parallel Mastrovito multiplier (PMM) [6] operates over the polynomial basis and is implemented by m identical functions each comprising m AND gates and $(m-1)$ XOR gates (identical to the type A module in the PDBM) and one further module. It is not possible to generalize the hardware requirements of this extra module, although the lowest complexity modules for $GF(2^m)$ for a range of m have been found [7].

Both the PDBM and the PMM also require $2m$ registers elements, and so these components are not considered in the comparison.

The delay of the PDBM is now considered. Let D_A be the delay through a 2-input AND gate and let D_X be the delay through a 2-input XOR gate. Then the delay through the type A module is $D_A + D_X\lceil \log_2 m \rceil$. Now let

$$p(x) = x^m + x^k + \sum_{i=0}^{k-1} p_i x^i$$

be the defining irreducible polynomial for $GF(2^m)$. The delay through the type B module will be $t \times D_X\lceil \log_2(H(pp)-1)\rceil$ where

$$t = \begin{cases} 1 & k = 1 \\ 1 + \left\lfloor \dfrac{m-2}{m-k} \right\rfloor & k \geq 2. \end{cases} \quad (8)$$

(For an explanation of (8), see Appendix B.) Thus the delay through the PDBM in total is

$$D_A + D_X\left(t \times \lceil \log_2(H(pp)-1)\rceil + \lceil \log_2 m \rceil\right).$$

It can be seen that the hardware complexity of the PDBM is at its minimum when $p(x)$ is a trinomial. Furthermore, the delay through the PDBM is at its minimum when $p(x)$ is a trinomial of the form $p(x) = x^m + x + 1$. In Table 1, the hardware complexity and delays of the PDBM and the PMM are given for $GF(2^m)$ for $(m = 2, 3, \ldots, 10)$. The overall hardware levels and delays of the PMM are taken from [7]. The PDBM makes use the values of $p(x)$ given in Table 2 in Appendix A. Because these polynomials have the lowest Hamming weights of the available defining polynomials and the lowest values of k for each m, these choices of $p(x)$ yield the fastest and least hardware intensive bit-parallel dual basis multipliers. For $GF(2^8)$, the extra two XOR gates required to carry out the dual to polynomial basis conversion have been included. This allows both the multiplier and the multiplicand to enter the multiplier represented in the dual basis.

From Table 1, it can be seen that the PDBM and the PMM have very similar hardware requirements and longest delay paths. This is to be expected given the architectural similarities of the two multipliers, although the underlying algorithms are entirely different. The only significant difference is to be found with $GF(2^8)$, for when $m = 8$, the PBM requires 11 fewer XOR gates but has a longest delay path $3D_X$ more than the PMM. However, in [6], [7], it was noted that with some values of m it is possible to reduce the required number of XOR gates by reusing partial sums. This approach can also be adopted with the PBM, and these modified values are shown in brackets in Table 1.

TABLE 1
HARDWARE REQUIREMENTS AND DELAYS OF DUAL BASIS AND POLYNOMIAL BASIS BIT-PARALLEL MULTIPLIERS

	PDBM			PMM		
m	ANDs	XORs	delay	ANDs	XORs	delay
2	4	3	D_A+2D_X	4	3	D_A+2D_X
3	9	8	D_A+3D_X	9	8	D_A+3D_X
4	16	15	D_A+3D_X	16	15	D_A+3D_X
5	25	24	D_A+5D_X	25	25 (24)	D_A+5D_X
6	36	35	D_A+4D_X	36	33	D_A+4D_X
7	49	48	D_A+4D_X	49	48	D_A+4D_X
8	64	79(74)	$D_A+8D_X(7D_X)$	64	90 (76)	D_A+5D_X
9	81	80	D_A+5D_X	81	80	D_A+5D_X
10	100	99	D_A+6D_X	100	101 (99)	D_A+6D_X

5.3 Constant Bit-Parallel Dual Basis Multipliers

Just as in the bit-serial case, bit-parallel constant dual basis multipliers can be hardwired to produce hardware efficient constant multipliers, which are required in a number of applications such as RS encoders and syndrome calculators. To achieve this, the type A modules are replaced with modules that correspond to the polynomial basis representation of the constant multiplier c. For example, if a constant bit-parallel dual basis multiplier for $GF(2^4)$ is required to multiply field elements by α^{14}, each of the type A modules in Fig. 4 are replaced with the module implementing the equation $a_k = b_{k+3} + b_k$ (that is, an XOR gate), since the polynomial basis representation of α^{14} with $p(x) = x^4 + x + 1$ is (1001). The S_i registers $(i = 0, 1, 2, 3)$ shown in Fig. 4 are also not required in a constant multiplier.

Bit-parallel dual basis multipliers therefore allow for reduced complexity constant multipliers. However, it should be noted that unlike the bit-serial case, bit-parallel constant dual basis multipliers can only generate one product at a time. The idea of using bit-parallel constant Berlerkamp multipliers in RS encoders operating over $GF(2^8)$ was suggested in [24], however the case of general, 2-variable input multipliers was not considered and no hardware figures or delays were given.

6 Bit-Serial Systolic Divider for $GF(2^m)$

We now present a dual basis, systolic bit-serial divider which is also based upon Theorem 4. If we take $c = a/b$ and enforce the condition $b \neq 0$ in Theorem 4, we arrive at the following linear functions division algorithm (LFDA):

1) Construct (5).
2) Solve (5) to obtain the desired value of c.

The implementation of the LFDA can, therefore, be broken down into two sections, the section that generates the system represented in (5) and the section that solves these equations. The first of these problems is now addressed and as an example, an implementation of the LFDA for $GF(2^4)$ is presented.

6.1 Generating the System of Equations

In order to generate the values of b_i ($i = 0, 1, \ldots, 2m-2$) we again note that for ($i = 0, 1, \ldots, m-1$) these values of b_i correspond to the dual basis coefficients of b. Furthermore, the values of c_i ($i = 0, 1, \ldots, m-1$) also correspond to the dual basis coefficients of c. It only, therefore, remains to generate b_i ($i = m, m+1, \ldots, 2m-2$). As in Section 4, these values could be generated by an LFSR. However, because systolic architectures exist for solving the resulting system of equations, it would seen sensible to keep the architecture fully systolic and develop a systolic architecture to generate the components of the matrix in (5). To this end the matrix formulator (MF) cell shown in Fig. 5 was developed. This cell takes the dual basis coefficients b_i ($i = 0, 1, \ldots, m-1$) as input and generates the value of

$$b_m = \sum_{i=0}^{m-1} b_i p_i .$$

Consequently $(m-1)$ of these cells are needed to produce the values b_i ($i = m, m+1, \ldots, 2m-2$) in (5). For example, consider again $GF(2^4)$ with $p(x) = x^4 + x + 1$. Fig. 6 shows the arrangement of MF cells for $GF(2^4)$ and the inputs to and the outputs from this one-dimensional array, where b_i ($i = 0, 1, 2, 3$) are the dual basis coefficients of b and the p_{in} input corresponds to the coefficients of $p(x)$. Having generated the system of equations shown in (5), we now consider how to solve them.

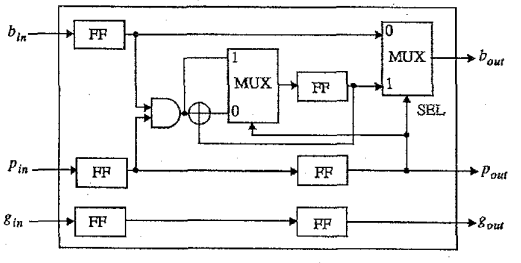


Fig. 5. MF cell for LFDA.

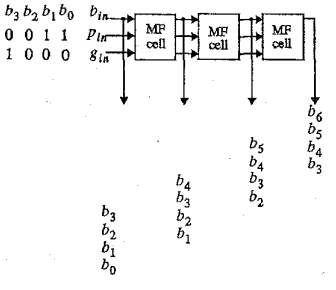


Fig. 6. MF cells for $GF(2^4)$ LFDA.

6.2 Solving Systems of Simultaneous Linear Equations over $GF(2)$

Simultaneous linear equations can be solved by Gauss-Jordan elimination (GJE), Gaussian elimination with back substitution, LU decomposition, or matrix inversion and multiplication. Solving systems of m simultaneous linear equations is of $O(m^3)$ complexity and so a degree of parallelism is required if these techniques are to be implemented in real time. Systolic arrays for implementing GJE over $GF(2)$ have been developed in [5], [25]. In [25], a systolic array for carrying out upper matrix triangularization was presented and in [5] this array was modified to carry out full GJE. This is achieved by clocking the array a further m clock cycles. Hence, to implement the LFDA, all that is required is to combine this GJE systolic array with an array of MF cells.

The GJE array given in [5] consists of two distinct processors, conventionally called "round" and "square" processors. The round processors determine whether a particular row is to act as a pivot and the square processors carry out the $GF(2)$ arithmetic specified by the round processor at the beginning of the row. For a detailed description of these cells, see [5]. The full arrangement of round and square processors for carrying out GJE over $GF(2^4)$ is shown in Fig. 7. Fig. 7 also shows the required control signal g_{in} and the ordering of the inputs to this array. Between the round processors on the leading diagonal of the array are two delay registers, these are required to synchronize g_{in}.

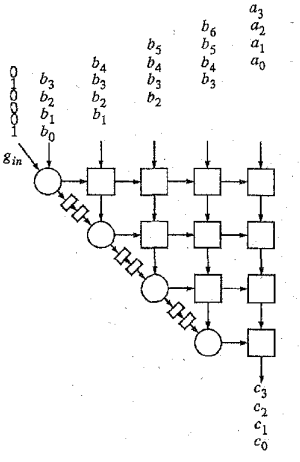


Fig. 7. Arrangement of round and square processor cells for GJE in $GF(2^4)$.

6.3 Complete Systolic Implementation of the LFDA

These architectures are now combined to produce the complete systolic implementation of the LFDA. An example of the systolic implementation of the LFDA for $GF(2^4)$ is shown in Fig. 8. The square boxes D_i represent delay units of i clock cycles. These are required because from Figs. 6 and 7 it can be seen that the output from the first MF cell is available in a different format to that required by the GJE array.

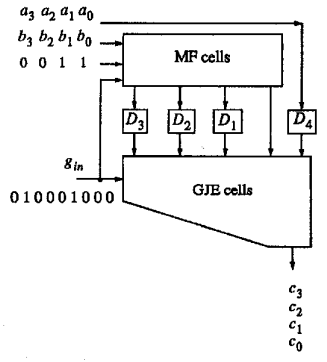


Fig. 8. Full systolic implementation of the LFDA for $GF(2^4)$.

There is a $3m$ clock cycle delay between the first coefficient of b entering the GJE array and the first coefficient of the vector becoming available at the bottom. Because an $(m-1)$ delay is required to convert the order of the b_i values from that produced by the MF cells to that required by the GJE array, the LFDA has a delay of $(4m-1)$ clock cycles between the first basis coefficients of a and b entering the circuit and the first basis coefficient of c becoming available on the output. The LFDA therefore has a total computation time of $(5m-1)$ clock cycles. This implementation of the LFDA also supports pipelining. As regards the hardware requirements of this implementation of the LFDA, each of the $(m-1)$ MF cells requires six registers and the delay units D_i consist of $m(m+1)/2$ registers. The GJE array consists of $2(m^2+4m-1)$ registers and so the divider requires $(2.5m^2+14.5m-8)$ registers in total.

6.4 Comparisons with Other Dividers

A similar systolic divider has been presented by Hasan and Bhargava [5]. Both dividers have the same hardware requirements and operation time and both can also be pipelined. The main difference between these two dividers—apart from the underlying algorithms—is the way in which they require the finite field elements to be represented. The divider presented in [5] operates over the polynomial basis and whereas the coefficients of a and b enter the circuit most significant coefficient first, the coefficients of c leave the circuit least significant coefficient first. In some situations this would prove unacceptable, and extra hardware would have to be included to reorder these coefficients. In the LFDA implementation given here, all coefficients enter and leave the circuit least significant coefficient first. However, it is to be remembered that a and b are represented in the dual basis while c is represented in the polynomial basis. Although as has already been pointed out, any required basis conversions can often be carried out with little or no extra hardware.

In fact, it can be regarded as an advantage that the LFDA operates over the dual basis. This is because multiplication is the most frequently used operation in the implementation of RS codecs and it is important to be able to utilize the most hardware efficient multipliers available. As has been seen earlier, hardware efficient dual basis multipliers exist in both bit-serial and bit-parallel forms and if the operating basis of a circuit is taken to be the dual basis, it is desirable that all the arithmetic operators should function over this same basis to avoid unnecessary basis conversions. And so in deciding whether to utilize the dividers presented here or in [5], the main criteria will be the required operating basis.

The divider presented here also shares a number of advantageous characteristics with the divider described in [5]. For example both dividers can operate with any irreducible polynomial, there are no global communications, the divider is easily extended to different $GF(2^m)$ and the longest delay path is independent of m. Compare this with the inverters presented in [18], [19], [20] for example. Firstly and most obviously, it should also be noted that for an inverter to carry out division, an extra finite field multiplier is required. (However, it is to be noted that by appropriately initializing the inverter in [20], this circuit can in fact carry out division). These inverters all have structures dependent upon the choice of $p(x)$ and they are also nonsystolic. Furthermore, the inverters described in [19] and [20] require two control lines whilst the inverter presented in [18] requires four control lines. For $GF(2^m)$ with large values of m, therefore, the dividers presented here and in [5] are more suitable for implementation in VLSI.

7 Conclusions

In this paper we have suggested an alternative definition of duality based not on the trace function but on the idea of a general linear function f. This has the advantage of simplifying the selection of dual bases since $tr(z)\ \forall z \in GF(2^m)$ does not have to calculated and f can be taken to be a single polynomial basis coefficient. The advantages to taking f to be the least significant polynomial basis coefficient, say, as opposed to the trace function will therefore be greatest when m is large. A general algorithm for carrying out finite field multiplication/division has been presented from which the Berlekamp bit-serial multiplier has been derived. This approach further allows for a class of bit-parallel dual basis multipliers to be designed. These multipliers have as low hardware requirements as any other bit-parallel multiplier, are easily extended to different $GF(2^m)$ and have a regular structure. Furthermore it is straightforward to design these multipliers given the irreducible polynomial for the field. It has also been demonstrated that these multipliers can be easily hardwired to carry out constant multiplication. Through use of the general multiplication/division algorithm a systolic bit-serial divider has been derived. Being systolic, this divider is modular, easily expandable to different $GF(2^m)$ and has a longest delay path independent of m. Consequently, it is anticipated this divider is most suited to applications where m is large or where high clock speeds are required. Furthermore, this divider also operates over the dual basis and so can be incorporated in circuits using hardware efficient, dual basis multipliers.

By using the techniques first presented in [10] it is possible to select dual bases which are permutations or which are close to being permutations of the polynomial basis. This results in little or no extra hardware being required to implement basis conversions and so dual basis operators can be utilized throughout RS codecs. Thus the problems traditionally associated with dual basis operators—that of the circuits

functioning over two different bases—can be largely overcome. Given that dual basis multipliers have low hardware requirements and there also exist efficient dual basis inverters and dividers ([23], [26] and Section 6 above), it is therefore suggested that the dual basis is an appropriate basis for RS codecs to operate over, whether the basis coefficients are represented bit-serially or in parallel.

Appendix A
Calculating Optimal Dual Bases

In [10], Morii et al. demonstrated that when the defining irreducible polynomial for $GF(2^m)$ is a trinomial of the form $p(x) = x^m + x^k + 1$ $(m > k)$ or a pentanomial of the form $p(x) = x^m + x^{k+2} + x^{k+1} + x^k + 1$ $(m > k + 2)$, convenient dual bases can be found. (Defining irreducible polynomials exist in this form for $GF(2^m)$ with $(m = 2, 3, \ldots, 11)$). It was then further shown that these results hold in the general case where f is any linear function [22].

A.1 Irreducible Trinomials

When the defining irreducible polynomial for $GF(2^m)$ is a trinomial of the form $p(x) = x^m + x^k + 1$ by selecting β in Definition 1 so that

$$f(\beta\alpha^i) = \begin{cases} 1 & i = k-1 \\ 0 & i = 0, 1, \ldots, m-1 \quad (i \neq k-1) \end{cases}$$

it can be shown [10], [22] that the dual basis to the polynomial basis is

$$\{\alpha^{k-1}, \alpha^{k-2}, \ldots, \alpha, 1, \alpha^{m-1}, \alpha^{m-2}, \ldots, \alpha^k\}. \tag{9}$$

Hence in this instance the dual basis is merely a permutation of the polynomial basis.

A.2 Irreducible Pentanomials of the Form $p(x) = x^m + x^{k+2} + x^{k+1} + x^k + 1$

When the defining irreducible polynomial for $GF(2^m)$ is a pentanomial of the form $p(x) = x^m + x^{k+2} + x^{k+1} + x^k + 1$ $(m > k + 2)$ by selecting β in Definition 1 so that

$$f(\beta\alpha^i) = \begin{cases} 1 & i = 0, k \\ 0 & i = 0, 1, \ldots, m-1 \quad (i \neq 0, k) \end{cases}$$

it can be shown that the dual basis to the polynomial basis is

$$\{\alpha^k, \alpha^{k-1}, \alpha^{k-2}, \ldots, \alpha, 1 + \alpha^k, \alpha^{k+1} + \alpha^{m-1}, \alpha^{m-1}, \alpha^{m-2}, \alpha^{m-3}, \ldots, \alpha^{k+2}, \alpha^{k+1}\}. \tag{10}$$

Hence, in this instance, the dual basis can be obtained from the polynomial basis with two additions and a reordering of basis coefficients. We now apply these results to produce the optimal dual bases for $GF(2^m)$ $(m = 2, 3, \ldots, 10)$.

A.3 Optimal Dual Bases for Selected $GF(2^m)$

There are irreducible trinomials (listed in Table 2) for all the fields $GF(2^m)$ $(m = 2, 3, \ldots, 10)$ except for $GF(2^8)$, for which there exists an irreducible pentanomial of the required form. Table 2 also gives the appropriate value of β and the resulting dual basis. For the sake of simplicity, the linear function f for all these fields is taken to be the least significant polynomial basis coefficient. Note that for $GF(2^8)$, the dual basis is a reordering of the polynomial basis coefficients plus two additions, whilst the remaining dual bases are permutations of the polynomial bases.

TABLE 2
Optimal Dual Bases for Certain Finite Fields
(α Is a Root of $p(x)$)

m	$p(x)$	β	dual basis
2	x^2+x+1	1	$\{1,\alpha\}$
3	x^3+x+1	1	$\{1, \alpha^2, \alpha\}$
4	x^4+x+1	1	$\{1, \alpha^3, \alpha^2, \alpha\}$
5	x^5+x^2+1	α^{30}	$\{\alpha, 1, \alpha^4, \alpha^3, \alpha^2\}$
6	x^6+x+1	1	$\{1, \alpha^5, \alpha^4, \alpha^3, \alpha^2, \alpha\}$
7	x^7+x+1	1	$\{1, \alpha^6, \alpha^5, \alpha^4, \alpha^3, \alpha^2, \alpha\}$
8	$x^8+x^4+x^3+x^2+1$	α^{253}	$\{\alpha^2, \alpha, 1+\alpha^2, \alpha^3+\alpha^7, \alpha^6, \alpha^5, \alpha^4, \alpha^3\}$
9	x^9+x+1	1	$\{1, \alpha^8, \alpha^7, \alpha^6, \alpha^5, \alpha^4, \alpha^3, \alpha^2, \alpha\}$
10	$x^{10}+x^3+1$	α^{1021}	$\{\alpha^2, \alpha, 1, \alpha^9, \alpha^8, \alpha^7, \alpha^6, \alpha^5, \alpha^4, \alpha^3\}$

By taking f to be the least significant polynomial basis coefficient rather than trace function, finding the appropriate value of β is simplified because the values of $tr(z)$ $\forall z \in GF(2^m)$ do not have to be calculated. In fact, these values of β do not have to be calculated at all since all we are ultimately interested in are the dual basis elements and these can be obtained directly from (9) and (10).

It is to be noted that the dual bases for $GF(2^m)$ with $m \leq 10$ have only been considered in this paper because we have considered dual basis operators with implementation in RS codecs in mind. Were it required for RS codecs to be implemented with $m > 8$ or with cryptographic applications with very large values of m, the techniques used in [10], [22] and reviewed above may have to be extended to cover irreducible polynomials which are not of the above form.

Appendix B
Delay through the B Type Module in the PDBM

To calculate the delay through the B type module of the PDBM, we must consider the generation of b_{2m-2}. In particular, we must establish how many of the b_i $(i = m, m + 1, \ldots, 2m - 3)$ contribute to the generation of b_{2m-2}. If $k = 1$ then $b_{2m-2} = b_{m-1} + b_{m-2}$ and so $t = 1$. More generally,

$$b_{2m-2} = b_{m+k-2} + \sum_{i=0}^{k-1} p_i b_{i+m-2}$$

and so we must find the greatest integer w such that

$$(m + k - 2) - (w - 1)(m - k) \geq m.$$

That is, we must find the greatest integer such that

$$w \leq \frac{m-2}{m-k}$$

and so

$$w = \left\lfloor \frac{m-2}{m-k} \right\rfloor.$$

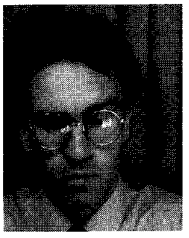
We must also include the delay involved in generating b_{2m-2} itself and hence we arrive at (8), as required.

ACKNOWLEDGMENTS

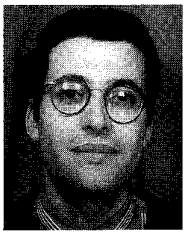
The authors would like to thank the anonymous referees for their suggestions on how to improve the quality of the manuscript.

REFERENCES

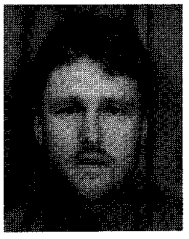
[1] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.
[2] R.L. Rivest, A. Shamir, and L.A. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Comm. ACM*, vol. 21, pp. 120-126, 1978.
[3] E.R. Berlekamp, "Bit-Serial Reed-Solomon Encoders," *IEEE Trans. Information Theory*, vol. 28, pp. 869-874, Nov. 1982.
[4] M.A. Hasan and V.K. Bhargava, "Division and Bit-Serial Multiplication over $GF(q^m)$," *IEE Proc. E.*, vol. 139, pp. 230-236, May 1992.
[5] M.A. Hasan and V.K. Bhargava, "Bit-Serial Systolic Divider and Multiplier for Finite Fields $GF(2^m)$," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 972-980, Aug. 1992.
[6] E.D. Mastrovito, "VLSI Design for Multiplication over Finite Fields," *LNCS-357, Proc. AAECC-6*, pp. 297-309, Rome, July 1988, Springer-Verlag.
[7] E.D. Mastrovito, "VLSI Architectures for Computations in Galois Fields," PhD thesis, Linkoping Univ., Linkoping, Sweden, 1991.
[8] I.S. Hsu, T.K. Truong, L.J. Deutsch, and I.S. Reed, "A Comparison of VLSI Architectures of Finite Field Multipliers Using Dual, Normal or Standard Bases," *IEEE Trans. Computers*, vol. 37, no. 6, pp. 735-737, June 1988.
[9] J.L. Massey and J.K. Omura, "Computational Method and Apparatus for Finite Field Arithmetic," U.S. Patent Application, Submitted 1981.
[10] M. Morii, M. Kasahara, and D.L. Whiting, "Efficient Bit-Serial Multiplication and the Discrete-Time Wiener-Hopft Equation over Finite Fields," *IEEE Trans. Information Theory*, vol. 35, pp. 1,177-1,183, Nov. 1989.
[11] P.A. Scott, S.E. Tavares, and L.E. Peppard, "A Fast VLSI Multiplier for $GF(2^m)$," *IEEE J. Selected Areas of Comm.*, vol. 4, pp. 62-66, Jan. 1986.
[12] M. Wang and I.F.Blake, "Bit-Serial Multiplication in Finite Fields," *SIAM J. Discrete Maths.*, vol. 3, pp. 140-148, Feb. 1990.
[13] R.C. Mullin, I.M. Onyszchuk, S.A. Vanstone, and R.M. Wilson, "Optimal Normal Bases in $GF(p^n)$," *Discrete Applied Maths.*, pp. 142-169, 1988/89.
[14] M. Kovac, N. Ranganathan, and M. Varanasi, "SIGMA: A VLSI Systolic Array Implementation of a Galois Field $GF(2^m)$ Based Multiplication and Division Algorithm," *IEEE Trans. VLSI Systems*, vol. 1, pp. 22-30, Mar. 1993.
[15] 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. Computers*, vol. 33, no. 10, pp. 906-911, Oct. 1984.
[16] R.E. Blahut, *Theory and Practice of Error Control Codes*. Reading, Mass.: Addison-Wesley, 1983.
[17] E.R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[18] K. Araki, I. Fujita, and M. Morisue, "Fast Inverter over Finite Field Based on Euclid's Algorithm," *Trans. IEICE E-72*, pp. 1,230-1,234, 1989.
[19] G-L. Feng, "A VLSI Architecture for Fast Iinversion in $GF(2^m)$," *IEEE Trans. Computers*, vol. 38, no. 10, pp. 1,383-1,386, Oct. 1989.
[20] C.C. Wang, T.K. Truong, H.M. Shao, L.J. Deutsch, J.K. Omura, and I.S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in $GF(2^m)$," *IEEE Trans. Computers*, vol. 34, no. 8, pp. 709-716, Aug. 1985.
[21] R. Lidl and H. Niederreiter, *An Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge Univ. Press, 1986.
[22] S.T.J. Fenn, "Optimised Algorithms and Circuit Architectures for Performing Finite Field Arithmetic in Reed-Solomon Codecs," PhD thesis, Univ. of Huddersfield, 1993.
[23] S.T.J. Fenn, M. Benaissa, and D. Taylor, "Division in $GF(2^m)$," *Electonic Letters*, vol. 28, pp. 2,259-2,261, Nov. 19, 1992.
[24] G.K. Maki and P.A. Owlsey, "Parallel Berlekamp vs. Conventional VLSI Architecture," *Gov. Microcircuit Applics. Conf. Rec.*, pp. 5-7, Nov. 1986.
[25] B. Hochet, P. Quinton, and Y. Robert, "Systolic Gaussian Elimination over $GF(p)$ with Partial Pivoting," *IEEE Trans. Computers*, vol. 38, no. 9, pp. 1,321-1,324, Sept. 1989.
[26] S.T.J. Fenn, M. Benaissa, and D. Taylor, "Improved Algorithm for Division over $GF(2^m)$," *Electronic Letters*, vol. 29, pp. 469-470, Mar. 4, 1993.



Sebastian T.J. Fenn received a BSc (Hons) degree in pure mathematics from the University College of Wales, Aberystwyth, an MSc in artificial intelligence from Kingston Polytechnic, and a PhD in electronic engineering from the University of Huddersfield in 1989, 1990, and 1993, respectively. He is currently a lecturer in the Department of Electrical and Electronic Engineering at the University of Huddersfield, prior to which he worked as a post-doctoral research fellow there for a year. His research interests are in finite field arithmetic, Reed-Solomon codec design, and the synthesis of digital architectures from HDLs.



Mohammed Benaissa joined the University of Huddersfield as a lecturer in 1990 after gaining his PhD from the University of Newcastle Upon Tyne. He is currently a senior lecturer in the Department of Electrical and Electronic Engineering at the University of Huddersfield. Dr. Benaissa's research interests include the areas of VLSI signal processing, number theoretic transforms, and coding. He has published widely in these areas.



David Taylor received the BSc (Hons) and PhD degrees in electronic engineering from the University of Huddersfield in 1983 and 1989, respectively. After graduating, he worked for two years in the mining industry as a systems designer and returned to academia in 1985. He has been a lecturer at the University of Huddersfield since 1985 and is currently reader in Electronics and Communications. His research interests include data coding, data coding architectures, and mixed-signal ASIC design and test. Dr Taylor was awarded an IEE prize on graduation in 1983. In 1987, he became a full member of the IEE and was awarded Chartered Engineer status. He also serves on the IEE Professional Group E3 (Microelectronics and Semiconductor Devices). Dr. Taylor is currently the leader of the Design Automation Research Group at the University of Huddersfield.