The Fast Decoding of Reed-Solomon Codes Using High-Radix Fermat Theoretic Transforms

K. Y. Liu and I. S. Reed
University of Southern California

T. K. Truong
TDA Engineering Office

Fourier-like transforms over $GF(F_n)$, where $F_n = 2^{2^n} + 1$ is a Fermat prime, have found application in decoding Reed–Solomon codes. It is shown here that such transforms can be computed using high-radix fast Fourier transform (FFT) algorithms requiring considerably fewer multiplications than the more usual radix 2 FFT algorithm. A special 256-symbol, 16-symbol-error-correcting, Reed–Solomon (RS) code for space communication-link applications can be encoded and decoded using this high-radix FFT algorithm over $GF(F_3)$.

I. Introduction

Recently, Justesen (Ref. 1) and Reed, Truong, and Welch (Refs. 2, 3) proposed that transforms over $GF(F_n)$ (Refs. 4, 5) can be used to define Reed–Solomon (RS) codes (Ref. 6) and to improve the decoding efficiency of these codes. The transform over $GF(F_n)$ is of the form

$$A(f) = \sum_{t=0}^{d-1} a(t)\,\gamma^{ft} \qquad \text{for } 0 \le f \le d-1 \tag{1}$$

where $F_n = 2^{2^n} + 1$ is a Fermat prime for $n \le 4$. In (1) the transform length d divides $F_n - 1$, $a(t) \in GF(F_n)$, and γ is a primitive dth root of unity which generates the d element cyclic subgroup

$$G_d = \{\gamma, \gamma^2, \cdots, \gamma^{d-1}, 1\}$$

in the multiplicative group of $GF(F_n)$. The inverse transform of (1) is

$$a(t) = (d)^{-1} \sum_{f=0}^{d-1} A(f)\,\gamma^{-ft} \qquad \text{for } 0 \le t \le d-1 \tag{2}$$

where (d) denotes the residue of d modulo F_n and $(d)^{-1}$ is the inverse of (d) in $GF(F_n)$.

To transform longer integer sequences over $GF(F_n)$, from Ref. 5, one can use the fact that $\gamma = 3$ is a primitive element in $GF(F_n)$. Such a γ gives a maximum transform length of 2^{2^n}.

In space communication links, it was shown in Ref. 7 that in the concatenated $E = 16$-error-correcting, 255-symbol RS code, each symbol with 8 bits and a $K = 7$, rate = ½ or ⅓, Viterbi decoded convolutional code, can be used to reduce the value of E_b/N_0 required to meet a specified bit-error rate P_b, where E_b is the received energy for each bit, and N_0 is the noise power spectral density at the receiver input.

Figure 1 presents a curve of concatenated code bit probability of the error bound vs E_b/N_0 for a $K = 7$, $R = ½$, convolutional code with 8 bits per RS symbol.

Since 3 is an element of order 2^{2^n} in $GF(F_n)$ (Ref. 5), an RS code of as many as 2^8 symbols of 9 bits each can be generated in $GF(F_3)$. Hence, by Ref. 2, the Fermat theoretic transform over $GF(F_3)$ can be used to decode an RS code of 2^8 symbols. For a given 223 information symbols, each of 8 bits, as mentioned above by Ref. 2, 224 information symbols in $GF(F_3)$, i.e., $S_1 = 0, S_2, \cdots, S_{224}$, can be represented in the range from 0 to $2^{2^3} - 1$. After encoding the information symbols, the parity check symbols in the 256-symbol RS code may occur in the range between 0 and 2^{2^3}. If 2^{2^3} is observed as a parity check symbol, deliberately change this value to 0, now an error. The transform decoder will correct this error automatically. Hence, the RS code generated in $GF(F_3)$ can be used to concatenate with a $K = 7$, rate ½ or ⅓ convolutional code.

The arithmetic used to perform these transforms over $GF(F_n)$ requires integer multiplications by powers of 3 and integer additions modulo F_n. However, integer multiplications by powers of 3 modulo F_n are not as simple as multiplications by powers of $\sqrt{2}$ modulo F_n, which can be implemented by circular shifts (Ref. 5). To remedy such a problem, it is shown here that high-radix fast Fourier transform (FFT) algorithms can be used to reduce the number of multiplications required for transforming integer sequences in $GF(F_n)$.

II. High-Radix FFT Algorithms Over $GF(F_n)$, Where F_n Is a Fermat Prime

In order to develop high-radix FFT algorithms over $GF(F_n)$, it is desirable, as we shall see, that multiplications involving the 2^ith roots of unity in $GF(F_n)$ be simple operations. This is made possible from the fact that the 2^ith roots of unity over $GF(F_n)$, where $2 \leq i \leq n+1$ are plus or minus power of 2 mod F_n.

To see this, note that if $2|s$, then

$$(\pm 2^{s/2})^2 \equiv 2^s \qquad \text{mod } F_n$$

and

$$[\pm 2^{(2^n+s)/2}]^2 = 2^{2^n} \cdot 2^s \equiv -2^s \qquad \text{mod } F_n$$

Hence, by theorem 2.20 of Ref. 8, the congruences

$$x^2 \equiv 2^s \qquad \text{mod } F_n \tag{3}$$

and

$$x^2 \equiv -2^s \qquad \text{mod } F_n \tag{4}$$

have exactly two solutions given by

$$x \equiv \pm 2^{s/2} \qquad \text{mod } F_n \tag{5}$$

and

$$x \equiv \pm 2^{(2^n+s)/2} \qquad \text{mod } F_n \tag{6}$$

respectively. Now let γ be a primitive dth root of unity in $GF(F_n)$, where $d = 2^t$ with $1 \leq t \leq 2^n$. Then by theorem 1 of Ref. 9,

$$\gamma^{d/2} = (\gamma^{d/4})^2 \equiv -1 \qquad \text{mod } F_n$$

Also by (6),

$$\gamma^{d/4} = (\gamma^{d/8})^2 \equiv \pm 2^{2^{n-1}} \qquad \text{mod } F_n$$

Combining (5) and (6), one obtains

$$\gamma^{d/8} = (\gamma^{d/16})^2 \equiv \pm 2^{k\,2^{n-2}} \qquad \text{mod } F_n$$

where $k = 1, 3$. By repeatedly applying (5) and (6) in this manner, one has finally

$$\gamma^{d/2^i} \equiv \pm 2^{k\,2^{n-i+1}} \qquad \text{mod } F_n \tag{7}$$

where $2 \leq i \leq n+1$ and $k = 1, 3, 5, \cdots, 2^{i-1} - 1$.

The high-radix FFT algorithms over $GF(F_n)$ are similar to those over the field of complex numbers (Refs. 10, 11). The following example illustrates the radix 16, decimation-in-frequency, twiddle-factor FFT over $GF(F_3)$.

Example: Let $F_3 = 2^{2^3} + 1 = 257$, $d = 16^2 = 256$. The radix 16, decimation-in-frequency, twiddle factor, FFT algorithm over $GF(F_3)$ is described as follows.

Let f and t in (1) be expressed as

$$f = f_1 \cdot 16 + f_0 \tag{8}$$

$$t = t_1 \cdot 16 + t_0 \tag{9}$$

where

$$f_i, t_i = 0, 1, 2, 3, \cdots, 15$$

Substituting (8) and (9) into (1), one has

$$A(f) = \sum_{t_0=0}^{15} \sum_{t_1=0}^{15} a(t_1 \cdot 16 + t_0)\gamma^{(f_1 \cdot 16 + f_0)\,(t_1 \cdot 16 + t_0)} \tag{10}$$

Since $\gamma^d = \gamma^{16^2} \equiv 1 \bmod F_3$, (10) becomes

$$A(f) = \sum_{t_0=0}^{15} \sum_{t_1=0}^{15} a(t_1 \cdot 16 + t_0)\gamma^{f_1 t_0 \cdot 16 + f_0 t_1 \cdot 16 + f_0 t_0}$$

$$= \sum_{t_0=0}^{15} \left[\left[\sum_{t_1=0}^{15} a(t_1 \cdot 16 + t_0)\gamma^{f_0 t_1 \cdot 16} \right] \gamma^{f_0 t_0} \right] \gamma^{f_1 t_0 \cdot 16}$$

Let

$$B_1(f_0 \cdot 16 + t_0) = \left[\sum_{t_1=0}^{15} a(t_1 \cdot 16 + t_0)\gamma^{f_0 t_1 \cdot 16} \right] \gamma^{f_0 t_0}$$

$$B_2(f_0 \cdot 16 + f_1) = \sum_{t_0=0}^{15} B_1(f_0 \cdot 16 + t_0) \cdot \gamma^{f_1 t_0 \cdot 16}$$

The radix 16, 256-point, FFT algorithm over GF(F_3) is then composed of the following stages:

Stage 1:

$$B_1(f_0 \cdot 16 + t_0) = \left[\sum_{t_1=0}^{15} a(t_1 \cdot 16 + t_0)\, \gamma^{f_0 t_1 \cdot 16} \right] \gamma^{f_0 t_0} \tag{11}$$

Stage 2:

$$B_2(f_0 \cdot 16 + f_1) = \left[\sum_{t_0=0}^{15} B_1(f_0 \cdot 16 + t_0)\gamma^{f_1 t_0 \cdot 16} \right] \tag{12}$$

From (7),

$$\gamma^{16} = \gamma^{d/16} \equiv \pm 2^k \qquad \bmod F_3$$

where $k = 1, 3, 5, 7$. It is shown in Ref. 9 that if γ is a primitive element in $GF(q)$, where q is a prime, then γ^m is also a primitive element in $GF(q)$, where $m = 3, 5, \cdots, q-2$. It is well known (Ref. 5) that 3 is a primitive element in $GF(F_n)$. Thus 3^m is also a primitive element in $GF(F_n)$ for $m = 3, 5, \cdots 2^{2^n} - 1$.

Now the choice of $\gamma = 3^3$ gives

$$\gamma^{16} = (3^3)^{16} \equiv (3^{16})^3 = (-2^3)^3 \equiv 2 \qquad \bmod F_3$$

since $3^{16} \equiv -2^3 \pmod{F_3}$ and $2^9 \equiv -2 \pmod{F_3}$. Hence, $\gamma^{f_0 t_1 \cdot 16}$ in (11) can take on only the values ± 1 or a power of 2.

Since multiplications by ± 1 involve only sign change and since multiplications involving powers of 2 mod F_3 can be achieved by circular shifts, the 16-point discrete Fourier transform in the brackets of (11) can be evaluated without multiplications. These results are then referenced by multiplying by the so-called twiddle factor $\gamma^{f_0 t_0}$. Using a similar argument, (12) can also be evaluated without multiplications.

The number of operations required to perform a FFT of 256 points using the radix 2, the radix 4, and the radix 16 FFT algorithms over $GF(F_n)$ is shown in Table 1. From this table, one can see that the radix 4 and the radix 16 FFT algorithms require 30% and 70% fewer multiplications, respectively, than the more usual radix 2 FFT algorithm.

In the above example, it was shown that one can find a power of 3 for γ such that

$$\gamma^{d/2^{n+1}} \equiv 2 \qquad \bmod F_n$$

For this γ, one has

$$\gamma^{d/2^{n+2}} \equiv \pm \sqrt{2} \qquad \bmod F_n$$

From Ref. 5,

$$\sqrt{2} \equiv 2^{2^{n-2}}(2^{2^{n-1}} - 1) \qquad \bmod F_n$$

Hence multiplications involving integral powers of $\gamma^{d/2^{n+2}}$ can be accomplished either by circular shifts or a circular shift followed by a subtraction, depending on whether an even or an odd power of $\sqrt{2}$ is involved. As a consequence, a high radix FFT up to 2^{n+2} also could be developed. For example, the 256-point FFT over $GF(F_3)$ could be computed using a mixed radix FFT of radix 32 and radix 8.

In light of the above discussion, when transforming long integer sequences in $GF(F_n)$, it is desirable to perform as many high-radix FFT iterations as possible to reduce the required multiplications.

Acknowledgment

The authors wish to thank Mr. B. Mulhall and Dr. B. Benjauthrit of JPL for supporting and encouraging the research that led to this paper.

References

1. Justesen, Jorn, "On the Complexity of Decoding Reed–Solomon Codes," *IEEE Trans. Inform. Th.*, Vol. IT-22, March 1976, pp. 237–238.

2. Reed, I. S., Truong, T. K., and Welch, L. R., "The Fast Decoding of Reed–Solomon Codes Using Number Theoretic Transforms," *The Deep Space Network Progress Report 42-35*, pp. 64–78, Jet Propulsion Laboratory, Pasadena, Calif., Oct. 15, 1976.

3. Reed, I. S., Truong, T. K., and Welch, L. R., "The Fast Decoding of Reed–Solomon Codes Using Fermat Theoretic Transforms and Continued Fractions" (this volume).

4. Rader, C. M., "Discrete Convolutions via Mersenne Transforms," *IEEE Trans. Comput.*, Vol. C-21, pp. 1269–1273, Dec. 1972.

5. Agarwal, R. C., and Burrus, C. S., "Fast Convolution Using Fermat Number Transforms with Applications to Digital Filtering," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. Assp-22, No. 2, Apr., 1974, pp. 87–97.

6. Reed, I. S., and Solomon, G., "Polynomial Codes over Certain Finite Fields," *SIAM J. Appl. Math.*, Vol. 8, June 1960, pp. 300–304.

7. Odenwalder, J., et al., "Hybrid Coding Systems Study Final Report," Linkabit Corp., NASA CR 114,486, Sept. 1972.

8. Niven, I., and Zuckerman, H. S., *An Introduction to the Theory of Numbers,* New York, Wiley, 1966.

9. Reed, I. S., and Truong, T. K., "The Use of Finite Fields to Compute Convolutions," *IEEE Trans. Inform. Th.*, Vol. IT-21, No. 2, Mar. 1975, pp. 208–212.

10. Bergland, G. D., "A Fast Fourier Transform Algorithm Using Base 8 Iterations," *Math. Comput.*, Vol. 22, No. 102, Apr. 1968, pp. 275–279.

11. Singleton, R. C., "An Algorithm for Computing the Mixed Radix Fast Fourier Transform," *IEEE Trans. Audio Electroacoust.*, Vol. AU-17, pp. 93–103, June 1969.

Table 1. Number of operations required to transform $d = 256$ points FFT over $GF(F_n)$, where n = 3, 4.

Algorithm	Mod F_n multiplications	Mod F_n additions	Circular shifts
Radix 2 ($d = 2^8$)	769	2048	0
Radix 4 [$d = (2^2)^4$]	513	2048	256
Radix 16 [$d = (2^4)^2$]	225	2048	544

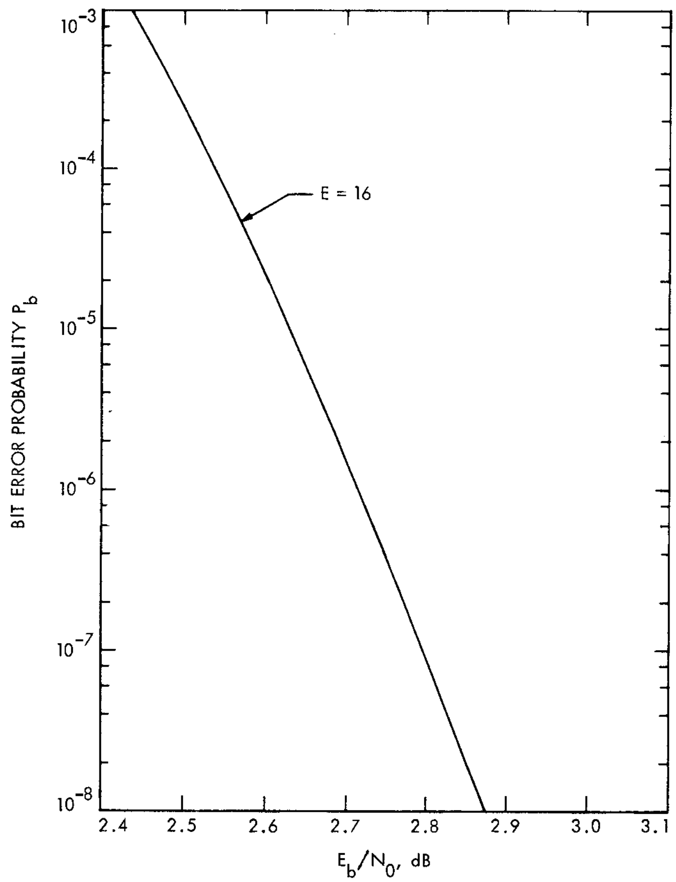

Fig. 1. Concatenated coding performance with a $K = 7$, $R = 1/2$ inner code and 8 bits/RS symbol