H. J. Nussbaumer

Complex Convolutions via Fermat Number Transforms

Abstract: An approach is described for computing complex convolutions modulo a Fermat number. It is shown that this technique is particularly efficient when the complex convolution is computed by means of Fermat Number Transforms and leads to improved implementation of complex digital filters.

Introduction

In most applications that involve the processing of digital signals, the bulk of the processing workload corresponds generally to digital filter functions. Among the various techniques that have been proposed for the efficient implementation of digital filters, those using finite field transforms [1, 2] are particularly promising. In such approaches, the continuous convolution corresponding to the digital filtering process is divided into a series of circular convolutions by the conventional overlap-add, overlap-save methods [3] and the various circular convolutions are computed by means of finite field transforms having the circular convolution property. The advantages of these transforms are the elimination of roundoff errors and the possibility of computation without multiplications. Additional computational savings can be achieved by using Fermat Number Transforms [4, 5] which are finite-field or ring transforms amenable to fast transform algorithms.

In this communication we consider the case of filtering complex signals. This case is important in many applications such as radar, sonar, and modem equalizers [6]. We show that, owing to the special representation of complex numbers in a Fermat number ring, it permits more efficient computation of complex convolutions than does the conventional complex number field. We then extend these results to the case of complex convolutions computed with Fermat transforms and show that the number of multiplications can be reduced by a factor of two when compared to the conventional Fermat transform approach.

Complex convolutions in a Fermat field

Consider a complex integer sequence $\{y_n\}$ to be filtered by a complex sequence having N terms $\{b_n\}$, in which $\{u_n\}$ is the filtered output sequence.

$\{u_n\}$ is defined by the convolution

$$u_m = \sum_{n=0}^{N-1} b_n\, y_{(m-n)}. \tag{1}$$

Assuming $\{x_n\}$, $\{a_n\}$, $\{z_n\}$ and $\{\hat{x}_n\}$, $\{\hat{a}_n\}$, $\{\hat{z}_n\}$ are respectively the in-phase and quadrature signal components of $\{y_n\}$, $\{b_n\}$, $\{u_n\}$, we have:

$$y_n = x_n + j\,\hat{x}_n; \tag{2}$$

$$b_n = a_n + j\,\hat{a}_n; \tag{3}$$

$$u_n = z_n + j\,\hat{z}_n, \qquad j = \sqrt{-1}. \tag{4}$$

Under these conditions, the in-phase and quadrature components of the output sequence become:

$$z_m = \sum_{n=0}^{N-1} \left(a_n x_{(m-n)} - \hat{a}_n \hat{x}_{(m-n)}\right), \tag{5}$$

$$\hat{z}_m = \sum_{n=0}^{N-1} \left(\hat{a}_n x_{(m-n)} + a_n \hat{x}_{(m-n)}\right). \tag{6}$$

It can be seen that direct computation of each complex output sample W_m requires $4N$ multiplications and $4N-2$ additions. These figures can be lowered to $3N$ multiplications and $3N+2$ additions by computing z_m with Golub's algorithm [7]

$$z_m = \sum_{n=0}^{N-1} \Big((a_n - \hat{a}_n)(x_{(m-n)} + \hat{x}_{(m-n)}) - a_n \hat{x}_{(m-n)} + \hat{a}_n x_{(m-n)}\Big). \tag{7}$$

Now consider the case of a convolution computed modulo a Fermat number $p = 2^q + 1$ with $q = 2^r$, as $2^q \equiv -1$, and $\frac{q}{2} = 2^{r-1}$, $j = \sqrt{-1}$ can be represented in this ring by $2^{q/2}$. It is therefore possible to compute directly a complex convolution u_m in a Fermat number system by

$$u_m = \left(\left(\sum_{n=0}^{N-1} (a_n + 2^{q/2}\hat{a}_n)(x_{(m-n)} + 2^{q/2}\hat{x}_{(m-n)})\right)\right), \quad (8)$$

where any quantity enclosed by superfluous double parentheses is to be replaced by its value modulo p.

Because $2^q \equiv -1$, Eq. (8) becomes

$$u_m = ((z_m + 2^{q/2}\hat{z}_m)). \quad (9)$$

The in-phase and quadrature components z_m and $\hat{z}_m$ of the output sample can be separated by considering the auxiliary convolution

$$v_m = \left(\left(\sum_{n=0}^{N-1} (a_n - 2^{q/2}\hat{a}_n)(x_{m-n} - 2^{q/2}\hat{x}_{m-n})\right)\right), \quad (10)$$

$$v_m = ((z_m - 2^{q/2}\hat{z}_m)). \quad (11)$$

Combining (9) and (11) yields

$$z_m = ((-2^{q-1}(u_m + v_m))); \quad (12)$$

$$\hat{z}_m = ((-2^{q-2/2}(u_m - v_m))), \quad (13)$$

which shows that computing a complex output sample requires only $2N$ multiplications and $2N + 4$ additions, that is to say half as many multiplications as with the conventional approach.

With this method, it is therefore possible to compute a complex convolution modulo a Fermat number with fewer operations than with the conventional approach or Golub's algorithm. The price to be paid for this reduction in number of operations is that all multiplications and additions must be performed in the finite Fermat field or ring. This will usually lead to the use of word lengths longer than with the conventional approach, or Golub's algorithm, in order to prevent overflow in the final result. This means that the reduction in number of operations achieved with the proposed approach does not necessarily translate into processing workload reduction.

We show, however, in the next section that when the complex convolution is computed by means of Fermat Number Transforms, it is possible to reduce the number of operations without additional penalty in word length increase, thereby achieving an overall processing workload reduction.

Complex convolutions using Fermat Number Transforms

As outlined in [4] and [5], a promising approach to computing convolutions consists in replacing direct or Fast Fourier Transform implementation (FFT) by Fermat Number Transform (FNT) implementation.

In such an approach, the continuous convolution is converted into a series of circular convolutions on blocks of samples $\{x_n\}$ and $\{a_n\}$ to which zeros are appended to prevent folding and aliasing. FNT transforms $\{A_k\}$ and $\{X_k\}$ of $\{a_n\}$ and $\{x_n\}$ are then computed and, because the FNT transform has the cyclic convolution property, taking the inverse FNT transform of $\{A_k \cdot X_k\}$ yields the desired convolution products. As Fermat Number Transforms can be computed by fast algorithms without multiplications, this method yields a drastic reduction in number of multiplications when compared to either direct or FFT implementation.

Fermat Number Transforms are computed modulo a Fermat number. The method described in the preceding section for computing complex convolutions modulo a Fermat number is therefore directly applicable to the case of a FNT implementation. However, in contrast with the approach discussed in the preceding section, taking advantage of the particular representation of complex numbers in a Fermat ring to reduce the number of operations will not yield additional word length increases because word sizes must already be tailored for operation modulo a Fermat number in the FNT implementation [4, 5].

In order to make these points precisely, let us first consider the conventional computation of a complex cyclic convolution via FNT. The Fermat and Inverse Fermat Number Transforms can be defined as

$$\text{FNT}\ (x_n) \triangleq X_k = \left(\left(\sum_{n=0}^{N-1} x_n 2^{nk}\right)\right); \quad (14)$$

$$\text{I FNT}\ (X_k) \triangleq x_m = \left(\left(R \sum_{k=0}^{N-1} X_k 2^{-mk}\right)\right); \quad (15)$$

$$N = 2q \qquad R = 2^{-(r+1)} \qquad n, k = 0, 1, \cdots, N-1.$$

Assuming X_k, $\hat{X}_k$, A_k, $\hat{A}_k$ are respectively the Fermat Number Transforms of x_n, $\hat{x}_n$, a_n, $\hat{a}_n$, the in-phase and quadrature components of the complex circular convolution become

$$z_m = \text{I FNT}\ \{A_k X_k - \hat{A}_k \hat{X}_k\}, \quad (16)$$

$$\hat{z}_m = \text{I FNT}\ \{\hat{A}_k X_k + A_k \hat{X}_k\}. \quad (17)$$

We can see that for a complex circular convolution of N points, this method requires computing six Fermat or Inverse Fermat Number Transforms and $4N$ multiplications and $2N$ additions in the transform domain.

As all operations are performed modulo a Fermat number, we can reduce the number of multiplications in the transform domain by using the method described in the preceding section. Under these conditions, z_m and $\hat{z}_m$ become

$$z_m = ((-2^{q-1}(\text{IFNT}\ \{(A_k + 2^{q/2}\hat{A}_k)(X_k + 2^{q/2}\hat{X}_k) + (A_k - 2^{q/2}\hat{A}_k)(X_k - 2^{q/2}\hat{X}_k)\})); \quad (18)$$

$$\hat{z}_m = ((-2^{q-2/2}(\text{IFNT}\ \{(A_k + 2^{q/2}\hat{A}_k)(X_k + 2^{q/2}\hat{X}_k) - (A_k - 2^{q/2}\hat{A}_k)(X_k - 2^{q/2}\hat{X}_k)\}))). \quad (19)$$

If we compare the conventional approach (16), (17) to that corresponding to (18), (19), we can see that both methods require computing six Fermat or Inverse Fermat Transforms but that the proposed approach requires only $2N$ multiplications and $6N$ additions in the transform domain.

Moreover, if the filter is time invariant, $-2^{q-1}(A_k + 2^{q/2}\hat{A}_k), -2^{q-1}(A_k - 2^{q/2}\hat{A}_k)$ can be precomputed once and for all so that the number of additions in the transform domain reduces to $4N$.

The proposed approach permits, therefore, the computation of a circular convolution by means of FNT with an average of only two multiplications per complex output sample instead of four multiplications in the conventional case. This processing workload reduction is achieved without word length increase.

Conclusion

It has been shown that complex convolutions can be computed efficiently modulo a Fermat number thanks to the particular representation of complex numbers in the corresponding field or ring.

This result is especially significant when complex convolutions are computed by means of Fermat Number Transforms. In that case, all operations are already performed modulo a Fermat number so that the proposed approach permits halving the required number of multiplications without imposing additional overflow constraints over what is required for the conventional technique using Fermat Number Transforms.

The method described in this paper may be used for filtering complex signals and therefore can find application in a number of cases concerning, e.g., radars, sonars, and modems.

References

1. J. M. Pollard, "The Fast Fourier Transform in a Finite Field," *Math. Comput.* **25,** 365 (1971).
2. D. E. Knuth, *The Art of Computer Programming. Vol. 2., Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1969. Ch. 4.3.3.C, pp. 269-275.
3. B. Gold, C. M. Rader, A. V. Oppenheim, and T. G. Stockham, *Digital Processing of Signals*, McGraw-Hill Book Company, Inc., New York, 1969. Ch. 7, pp. 203-213.
4. C. M. Rader, "Discrete Convolutions via Mersenne Transforms," *IEEE Trans. Comput.* **C-21,** 1269 (1972).
5. R. C. Agarwal and C. S. Burrus, "Number Theoretic Transforms to Implement Fast Digital Convolution," *Proc. IEEE* **63,** 550 (1975).
6. R. D. Gitlin, E. Y. Ho, and J. E. Mazo, "Passband Equalization of Differentially Phase-Modulated Data Signals," *Bell System Tech. J.* **52,** 219 (1973).
7. R. C. Singleton, "An Algorithm for Computing the Mixed Radix Fast Fourier Transform," *IEEE Trans. Audio and Electroacoustics* **AU-17,** 93 (1969).

Received November 10, 1975

The author is located at Compagnie IBM France, Centre d'Etudes et Recherches 06610 La Gaude, France.