Application of frequency-domain polyphase filtering to quadrature sampling

Andrew K. Halberstadt

Massachusetts Institute of Technology, Department of Electrical Engineering
Research Laboratory of Electronics, 50 Vassar Street, Cambridge, MA 02139

ABSTRACT

A new method for developing in-phase and quadrature samples of a bandlimited real signal is presented and compared with other methods. The proposed method improves the computational efficiency of a frequency-domain implementation of the required filtering by taking advantage of mirror-image symmetry between the filters on the in-phase and quadrature channels. This technique makes it possible to filter both the in-phase and quadrature channels at once using one standard FFT for complex-valued input and one standard IFFT for complex-valued output without adding computational overhead.

The proposed method is more computationally efficient than techniques using commonly available FFT routines for real-valued input. Specialized FFT routines derived specifically for real-valued input can achieve greater computational efficiency than the proposed method, but these routines are not as commonly available as the standard FFT routines used in the proposed method. For systems where the effective channel filter is short in length or has special structure, there are other implementations of the channel filters which are more efficient than the proposed method.

Keywords: I/Q conversion, quadrature sampling, polyphase filters

1 INTRODUCTION

In radar and sonar applications it is often useful to obtain complex-valued in-phase and quadrature samples of real-valued sensor data. As a result, numerous methods have been proposed[1–4] for performing quadrature sampling. These methods focus on efficient time-domain implementation of the filtering on each channel. In contrast, this paper presents a technique for improving the computational efficiency of a frequency-domain implementation of the required filtering by taking advantage of mirror-image symmetry between the filters on the in-phase and quadrature channels.

0-8194-1922-2/95/$6.00

2 THE PROPOSED METHOD

Fig. 1 illustrates the components in a conceptually simple in-phase and quadrature sampling system. The input to the A/D converter is a real, bandpass signal $x(t)$ whose spectrum has a bandwidth of B Hertz centered at f_o, as sketched in Fig. 1. The value of f_o is chosen to be one fourth of the sampling frequency so that the digital cosine and sine mixers occuring after the sampling process will be trivial. The sampling frequency must be high enough to satisfy the Nyquist criterion, i.e., it must be greater than $2(f_o + \frac{B}{2})$. The sine and cosine mixers in the two-channel portion of the system implement the equivalent of multiplying by the complex exponential $e^{-j\pi n/2}$ with the real part in the upper channel and the imaginary part in the lower channel. This operation translates the spectrum so that what was originally the positive-frequency portion of the spectrum is centered at $f = 0$. Since the same low-pass filtering and downsampling operations are applied to both channels, this is equivalent to having one channel processing the complex-valued input $x[n]e^{-j\pi n/2}$. Thinking in this way, the low-pass filters will remove what was the negative-frequency portion of the spectrum of $x(t)$. The signal $y[n] = i[n] + jq[n]$ is a complete representation of $x(t)$ as a complex-valued, low-pass signal sampled at a rate of B or higher. In contrast, conventional demodulation and sampling produces a real-valued, low-pass signal that requires a sampling rate of at least $2B$. The real part of the output, $i[n]$, is referred to as the in-phase channel, and the imaginary part of the output, $q[n]$, is referred to as the quadrature channel.

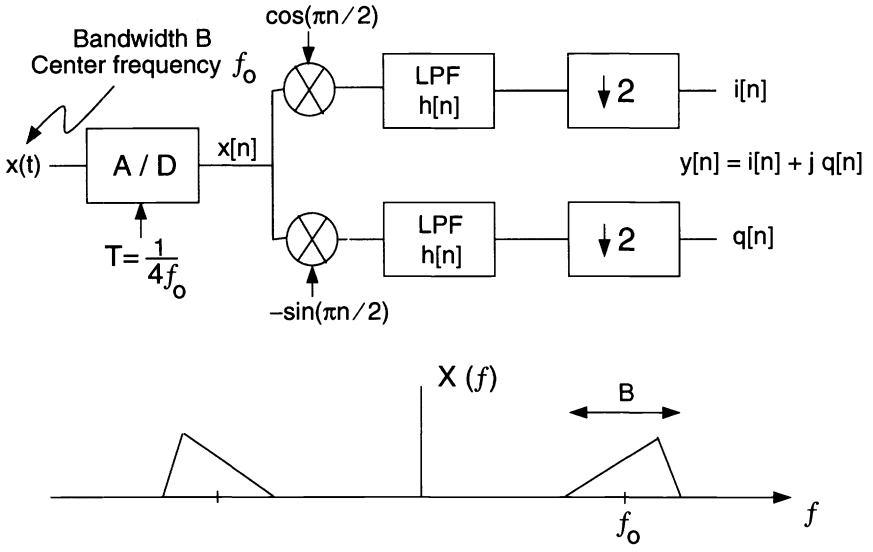

Figure 1: A simple quadrature sampling system

Recall that the relationship between f_o and f_s leads to trivial digital cosine and sine mixers, i.e., they take on values 1,0,-1,0, and 0,-1,0,1, respectively. The next step is to rearrange the computation in Fig. 1 so that the zeros introduced by the cosine and sine mixers are removed *before* the filtering is performed. Fig. 2 illustrates such a rearrangement. In this paper, $h[n]$ is referred to as the effective channel filter, and $h_i[n]$ and $h_q[n]$ are referred to as the in-phase and quadrature channel filters, respectively. If $h_i[n]$ and $h_q[n]$ are chosen according to

$$h_i[n] = h[2n] \tag{1}$$

$$h_q[n] = h[2n+1] \tag{2}$$

then the systems in Fig. 2 and Fig. 1 will produce the same output signals $i[n]$ and $q[n]$. The filters $h_i[n]$ and $h_q[n]$ are also referred to as polyphase filters.[5]

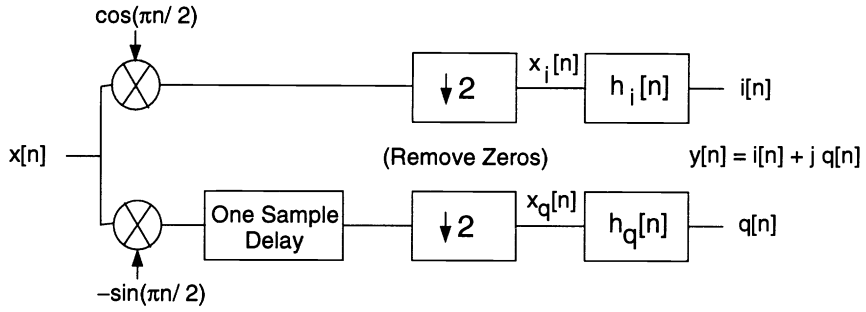

Figure 2: Rearrangement in terms of polyphase filters

If $h[n]$ is chosen to be a symmetric, even-length FIR filter which may be nonzero only for $0 \leq n \leq 2P-1$, then $h_i[n]$ and $h_q[n]$ will both have length P and will satisfy the relation

$$h_q[n] = \begin{cases} h_i[(P-1)-n] & 0 \leq n \leq P-1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Notice that this is a very mild constraint on $h[n]$ since an FIR linear phase filter is often desired in this context.

Eq. (3) implies that there is mirror image symmetry between $h_i[n]$ and $h_q[n]$, that is, $h_i[n]$ and $h_q[n]$ contain exactly the same values, but in reverse order. To take advantage of this symmetry, recall the following relation for discrete convolution. Let $y_r[n] = y[-n]$, $x_r[n] = x[-n]$, and $h_r[n] = h[-n]$, where the subscript r indicates time-reversal. If

$$y[n] = \sum_k h[k]x[n-k], \tag{4}$$

then

$$y_r[n] = \sum_k h_r[k]x_r[n-k]. \tag{5}$$

In other words, if the data and the filter are time-reversed, the output will also be time-reversed. This suggests, essentially, that if $x_q[n]$ is time-reversed and filtered by $h_i[n]$ then the time-reverse of $q[n]$ will result. Note that $x_i[n]$, $x_q[n]$, $i[n]$, and $q[n]$ are defined in Fig. 2. In order for this to work with causal filters and FFT processing, the time-reversal operation needs to be implemented using circular flipping and circular shifting.

To show how the circular flipping and shifting should be done, define $H_q[k]$ to be the N-point DFT of $h_q[n]$. Let $H_i[k]$, $X_q[k]$, and $Q[k]$ be similarly defined. It follows from Eq. (3) that

$$H_q[k] = W_N^{k(P-1)} H_i[((-k))_N] \tag{6}$$

$$= \begin{cases} H_i[0] & k = 0 \\ W_N^{k(P-1)} H_i[N-k] & 1 \leq k \leq N-1 \end{cases} \tag{7}$$

where $((m))_N = m \bmod N$ and $W_N = e^{-j2\pi/N}$. The goal is to express the output of the quadrature channel in terms of the in-phase filter $h_i[n]$ and a modified version of $x_q[n]$, so that only the $h_i[n]$ filter must be implemented. Consider the DFT of the quadrature channel output

$$q[n] \longleftrightarrow Q[k] = \left[W_N^{k(P-1)} H_i[((-k))_N]\right] X_q[k] \tag{8}$$

$$q[((-n))_N] \longleftrightarrow H_i[k]\left[W_N^{-k(P-1)} X_q[((-k))_N].\right] \tag{9}$$

To determine the time domain operations on $x_q[n]$ required to obtain the bracketed expression on the right of Eq.(9) note that

$$x_q[((-n-(P-1)))_N] \longleftrightarrow W_N^{-k(P-1)} X_q[((-k))_N] \tag{10}$$

Equations (9) and (10) together indicate how to obtain $i[n]$ and $q[n]$ through frequency-domain implementation of the $h_i[n]$ and $h_q[n]$ filters using a single FFT and IFFT. This procedure is illustrated in Fig. 3. $i[n]$ and $q[n]$ appear as the real and imaginary outputs of the IFFT block because $x_i[n]$, $x_q[n]$, and $h_i[n]$ are all real-valued.

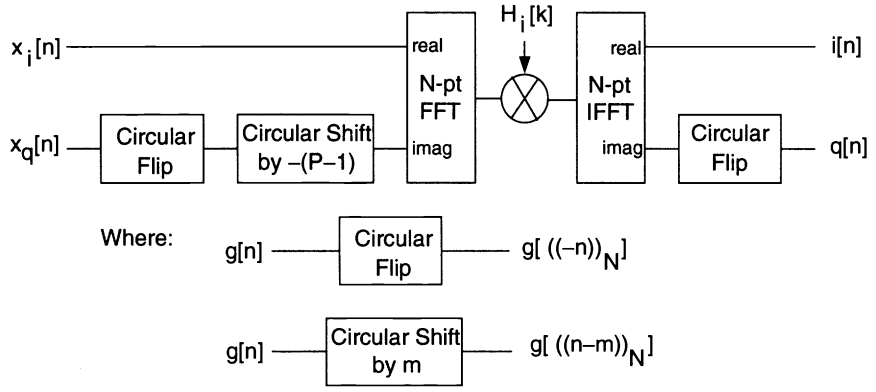

Figure 3: Proposed frequency-domain implementation

The appropriate choice for N, the length of the FFT's, will depend on the filter lengths and on the length of the input data stream. An understanding of block convolution and a comparison of the computational efficiency of different choices for N should guide the decision in any given application.

3 COMPARISON WITH OTHER METHODS

3.1 Mitchell's method

In Mitchell's method[2] the frequency response of $h[n]$ must be half-band[5] and symmetric about $\omega = \pi/2$, that is, the zero-phase frequency response must satisfy

$$H(e^{j\omega}) + H(e^{j(\pi-\omega)}) = 2h[0]. \tag{11}$$

This implies that the magnitude of the deviations from the ideal frequency response in the passband at ω_o and in the stopband at $(\pi - \omega_o)$ are equal, and therefore the passband and stopband error cannot be independently adjusted. The corresponding conditions on the zero-phase filter $h[n]$ are

$$h[n] = h[-n] \tag{12}$$

$$h[n] = 0 \text{ for } n = \pm 2, \pm 4, \pm 6, \ldots \tag{13}$$

If the constraint of half-band symmetry is appropriate in a given application, then Mitchell's method is a very good choice because the zeros in $h[n]$ imply that only one channel needs to be filtered. As a result, Mitchell's method is more computationally efficient than the proposed method. If $h[n]$ is long, then Mitchell's method must be implemented using FFT's or some other fast convolution algorithm to remain more efficient than the proposed method. The computational efficiency of Mitchell's method is achieved due to the half-band symmetry in $H(e^{j\omega})$.

3.2 Pellon's method

Pellon[4] takes a different approach to the design of the filters $h_i[n]$ and $h_q[n]$. He derives the specifications on these filters directly, then designs them as interpolating filters. His analysis of the errors introduced by the

filters $h_i[n]$ and $h_q[n]$ is very helpful. Pellon implements the filters directly in the time domain using convolution because he considers short filters of 8 or 12 taps. It is well known that implementing convolution in the frequency domain using a basic FFT-multiply-IFFT strategy is more efficient than direct convolution for long filters. Hence, the proposed method is more computationally efficient than direct convolution for filters with length greater than some threshold, P_t. The exact value of this threshold depends on what FFT routines are used. If a basic radix-4 FFT of length 256 with a 3/3 complex multiplication algorithm (each complex multiplication is computed using 3 real multiplications and 3 real additions) is used,[6] the value of P_t is approximately 29.

3.3 Comparison with real-valued FFT methods

The signals $x_i[n]$, $x_q[n]$, $h_i[n]$, and $h_q[n]$ are all real-valued. There are FFT routines available which take advantage of this to reduce the number of operations needed to compute their transforms. Fast DFT and convolution algorithms have been studied extensively.[6,8,11] The purpose of this section is not to explore the wide variety of possible algorithms for fast convolution, but rather to focus on a few of the methods for computing the FFT of real-valued data which are commonly used in FFT-multiply-IFFT schemes for convolution. The techniques examined in this section make no assumptions about the form of the effective channel filter, $h[n]$. See the references[6–8,11] for detailed discussion and operation counts for the algorithms described here.

When comparing computational complexity, only arithmetic operations are counted, so the circular flipping and shifting shown in Fig. 3 is not considered. However, this is reasonable since circular flipping and shifting simply changes the order in which data is read to and from memory. The reordering doesn't involve complicated scrambling of the original order, but instead results in a consecutive ordering that wraps around the end of the buffer. The memory addressing required is, therefore, not inherently expensive to implement. In fact, the data can be read into a buffer in blocks according to this scheme so it can then be processed in the usual way, then read out of the output buffer in flipped order.

In Figures 4 and 5, the multiplications by $H_i[k]$ and $H_q[k]$ can be performed with the FFT values in a packed form which removes the redundancy due to conjugate symmetry. If the packed form is employed, then the total number of real multiplications needed to calculate the point-wise product of the input and filter transforms is approximately the same for the systems in Figures 3, 4, and 5. Therefore, the significant differences in computational efficiency between the methods will arise from the calculation of the FFT's, not from forming the product of the input and filter transforms. For this reason, the efficiency of the FFT's is the focus of the discussion below.

3.3.1 Real-valued FFT's using complex-valued FFT's

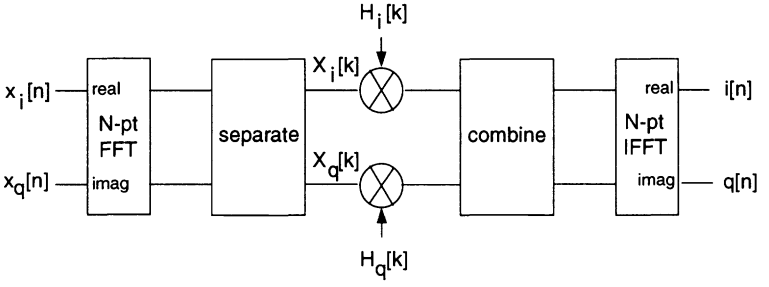

Figure 4: 2-for-1 real-valued FFT method

There are three common ways[7–12] to implement a real-valued FFT using a complex-valued FFT. The first is the very obvious but inefficient method of using a complex-valued FFT routine with the imaginary input set to zero. This is clearly wasteful. The second method, illustrated in Fig. 4, is called the "2-for-1" or "doubling" method because it gives two real-valued FFT's for the cost of one complex-valued FFT, plus some additional computation. This method is not as computationally efficient as the proposed method since it requires separating $X_i[k]$ and $X_q[k]$ in the frequency domain. The third method computes the FFT of a single real-valued sequence $x[n]$ of length $2N$ by forming the complex signal $x[2n] + jx[2n+1]$ of length N, taking the transform of the complex signal, then constructing $X[k]$ using some additional computation. This is referred to as the "packing" method. This method is illustrated in Fig. 5, emphasizing the fact that in this method the two channels are filtered independently. The performance depends on the algorithm being used in the "FFT for real data" block in the diagram. When the packing method is used, the system in Fig. 5 is less computationally efficient than the system based on the 2-for-1 method, so it is also less computationally efficient than the proposed method.

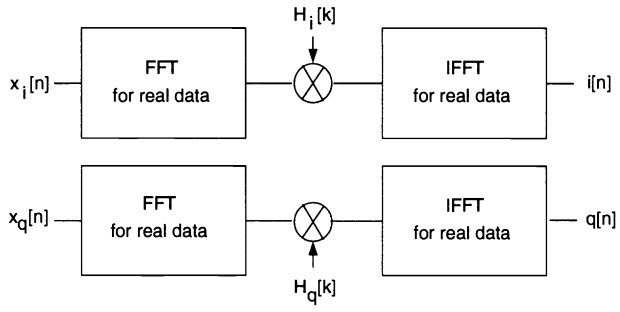

Figure 5: Real-valued FFT method, keeping the two channels separate

3.3.2 Real-valued FFT's not based on complex-valued FFT's

There are FFT algorithms derived specifically for use with real-valued input which achieve a factor of 2 reduction in multiplications and storage and slightly more than a factor of 2 reduction in additions compared to routines for complex-valued input.[7,8] This implies that specialized FFT algorithms for real-valued input data, used as shown in Fig. 5, are slightly more computationally efficient than the proposed method. Unfortunately, however, these routines are not as commonly or readily available as standard complex-valued FFT routines. For example, the SHARP LH9214 FFT processor can perform the 2-for-1 and packing methods for real-valued data, but cannot perform the specialized real-valued FFT methods described by Sorensen, *et al.*[7] Programmable processors could be used to implement special methods, but the system designer will also want to consider what routines are already available in the software library. The proposed method is particularly suited for systems using a dedicated FFT processor like the SHARP LH9214 because the circular flipping and shifting could be handled by an address generator or other data routing external to the actual computation required by the processor.

3.3.3 Summary of real-valued FFT methods comparison

Table I summarizes the comparison with real-valued FFT methods. The main purpose of the table is to give an indication of the order of magnitude of the percent change in computational efficiency which occurs when changing methods. Table I indicates that the differences in computational efficiency among these techniques are small compared to the total required computation. The operation counts shown here are based on Table II in Sorensen, *et al*,[7] and are particular to using optimized mixed-radix real-valued or complex-valued FFT routines, as appropriate, in each of the methods. In this context, optimization refers to avoiding trivial multiplications and additions so that the operation count is as low as possible. More significant percent changes in the operation

counts are possible if optimized, efficient routines used in one method are compared with unoptimized, less efficient routines used in another method.

TABLE I
COMPUTATIONAL EFFICIENCY IN THE FORWARD FFT STAGE FOR VARIOUS METHODS WITH THE FFT LENGTH EQUAL TO 256. THE OPERATION COUNTS ARE BASED ON TABLE II IN SORENSEN, *et al.*[7]

Method	Multiplications	Additions	% change in total computation compared to the proposed method	
			Multiplications	Additions
Proposed	1284	5380	–	–
2-for-1	1284	5888	0%	+9%
Packing	1408	6008	+10%	+12%
Specialized	1284	4872	0%	−9%

4 CONCLUSIONS

A novel frequency-domain approach to implementing the filtering in a quadrature sampling system has been presented. The proposed method reduces the required number of arithmetic operations by taking advantage of mirror-image symmetry between the filters on each channel.

Comparison with other methods yields several conclusions. First, for a given filter length, Mitchell's method is the most computationally efficient of the methods examined in this paper, but it requires that the effective channel filter be half-band symmetric. Second, frequency-domain implementation should be considered in cases where the channel filters are long. For short channel filters, direct convolution on each channel is better than FFT-multiply-IFFT schemes. Specialized number-theoretic schemes and other methods for fast convolution were not considered. Third, for systems with long effective channel filters which are not half-band symmetric, the computational efficiency of the proposed method is superior to methods based on commonly available real-valued FFT routines, but inferior to methods using specialized real-valued FFT routines. Simplicity of use and ready availability on a processor will be important factors in deciding among these methods. The final choice in any system will depend on an analysis of the advantages and disadvantages of each of the methods in light of the system design constraints.

5 ACKNOWLEDGMENTS

This work was sponsored in part by the Department of the Navy, Office of the Chief of Naval Research, contract number N00014-93-1-0686 as part of the Advanced Research Projects Agency's RASSP program. Special thanks are also due to the Lockheed Sanders RASSP program, since these ideas arose while working on the design of a synthetic aperture radar system for that program. The author wishes to thank Steve Lang, Cory Myers, Paul Fiore, Al Oppenheim, and Andy Singer for helpful ideas and discussion. Of course, responsibility for the final form of the manuscript rests with the author.

6 REFERENCES

[1] C. M. Rader, "A simple method for sampling in-phase and quadrature components." IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-20, No.6, pp. 821-824, November 1984.

[2] R. L. Mitchell, "Creating Complex Signal Samples From a Band-Limited Real Signal." IEEE Transactions on Aerospace and Electronic Systems, Vol. 25, No.3, pp. 425-427, May 1989.

[3] D. W. Rice and K. H. Wu, "Quadrature Sampling with high dynamic range." IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-18, No.4, pp. 736-739, November 1982.

[4] L. E. Pellon, "A Double Nyquist Digital Product Detector for Quadrature Sampling." IEEE Transactions on Signal Processing, Vol. 40, No.7, pp.1670-1681, July 1992.

[5] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, pp. 79-82,155-157, 1983.

[6] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* Addison-Wesley Publishing Company, pp. 118-127,152, 1985.

[7] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, "Real-Valued Fast Fourier Transform Algorithms." IEEE Trans. Acoust., Speech, Signal Process. Vol. 35, No. 6, pp 849-863, 1987.

[8] H. V. Sorensen and C. S. Burrus, "Fast DFT and Convolution Algorithms." In *Handbook for Digital Signal Processing* (S. K. Mitra and J. F. Kaiser, editors). New York, NY: John Wiley & Sons, pp. 491-610, 1993.

[9] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, pp. 316, 656-657, 1989.

[10] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing.* New York, NY: Macmillan, pp. 720-722, 1988.

[11] C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms.* John Wiley & Sons, pp.79-80, 1985.

[12] IEEE Acoustics, Speech and Signal Processing Society (IEEE ASSP Soc.), *Programs for Digital Signal Processing.* New York, NY: IEEE Press, 1979.