

Biplex Pipelined FFT

R. F. Emerson

Communications Systems Research Section

A method of implementing a pipelined fast Fourier transform (FFT) that makes full use of the elements that comprise it is presented. For a given bandwidth and spectral resolution this approach requires 25 percent less memory, and needs only half the logic speed required for the conventional implementation. Since the spectrum is broken into two parts, two analog-to-digital converters, and attendant mixer-filters, are needed. These two input elements need only operate at half the speed of the one they replace.

I. Introduction

The Biplex pipelined fast Fourier transform (FFT) is an improved hardware implementation of the pipelined FFT processor. After a brief discussion of the FFT algorithm and illustrations of some possible implementations, the Biplex approach will be presented. For a given bandwidth and spectral resolution, the Biplex FFT uses 75 percent of the memory, and logic of half the speed, needed for a conventional implementation. In addition, it saves one stage of complex arithmetic processing—the butterfly (*bf*). This gain is not free. Since the spectrum is split, an additional mixer-filter-converter module is required. Again, however, these elements need operate at only half the speed of those that they replace.

Briefly, the FFT is a method of rapidly computing the digital Fourier transform (Ref. 1). The bandwidth and resolution of the transform are determined by the sam-

pling rate and the number of points used to calculate the transform.

II. Implementation Background

The minimum amount of hardware to implement an FFT can be described by an in-place algorithm, see Fig. 1 (Ref. 1, p. 574). This approach requires storage for N complex points (N is the number of points used to compute the spectrum and must be a power of 2). In Fig. 1 these are represented as dots ($\bullet$). While four sets of eight are shown, these are the same eight cells with different contents as processing progresses from left to right. At level 0 (zero), the eight complex samples are stored into the locations 0 through 7. At level 3 ($\log_2(N)$ in general) the frequency points are available, indexed in bit-reversed order, and illustrated by the F_i 's in the figure. In this figure there are two directions that represent increasing

time: from top to bottom and from left to right. All points must be processed at level n before processing is started at level $n + 1$. Because of this, only one *bf* processor is required for the entire transformer. A *bf* processor performs a complex multiplication and two complex additions. These are implemented with four real multipliers and six real adders. Figure 2 shows, in increasing detail, the *bf* processor. In the first representation (a), A , B , C , and D are complex numbers, with A and B as inputs to the *bf*, and C and D as the outputs. The arrow with the letter I near it represents a complex multiplication by the I^{th} coefficient, W^I ($W^I = \text{cis}(2\pi I/N)$). Figure 2b shows this processor in more conventional block diagram form. The arithmetic units of Fig. 2b operate on complex numbers but are implemented by the combination of real arithmetic units of Fig. 2c. Here a complex number, A , is broken into its real A_r , and imaginary, A_i , parts. These are processed separately to produce the outputs.

Returning to Fig. 1 and scanning the storage cells to the left of the diagram, we see that only $N/2$ points need be stored before processing is started since the first *bf* operation is performed on samples zero and 4 (0 and $N/2$ in general). Only half of the resultants, $N/4$, need be stored at level 1 before processing can begin. Each resultant is, however, two values and therefore two sections of $N/4$ memory are required. For each level from this point on, the storage requirement is one-half the previous stage. By this process of distributing the memory and by adding a *bf* processor at each level, a pipelined processor can be constructed (Fig. 3). The switches between stages operate at increasing speeds ($T/2$, $T/4$, $\dots$), and are used to route the resultants through the memories at the proper times. Each switch has two positions—straight through or crossed. The pipeline processor requires storage for $1.5N$ complex points and $\log_2(N)$ *bfs* and switches. Both *bfs* and memory are used only half the time and operate at the input sampling rate. The next section will discuss several ways of improving this utilization.

III. Improvements Toward Full Utilization

Several approaches have been developed to improve the use of the elements in the pipelined processor.

The memory can be used all of the time by recirculating half the *bf* resultants in the present stage memory (through switching), before transferring them to the next stage (see Fig. 4). This has the further advantage of requiring only N complex storage locations rather than

$1.5N$. The *bfs* in this approach still operate only half the time and must be capable of processing at the sampling rate.

Another approach permits the use of the *bfs* all the time and at half the input sampling rate (see Fig. 5). This was derived by observing that two input points are required before processing and that two values result from each operation. Unfortunately, this method needs an additional input buffer of $1.5N$ complex storage cells. The total memory required for this method is, therefore, $3N$. It further requires that these input buffers operate at two different clocking rates.

The new approach presented in this report, Biplex (Fig. 6), takes advantage of the idle time of the processor in Fig. 3 by sampling a different signal when it would have been idle. The pipeline processor is switched between two sets of analog-to-digital converters, each sampling a separate signal. These are represented as complex mixers, CM1 and CM2. This approach can be used to process two independent signals or, by splitting a single signal into two adjacent channels, process one signal at the same resolution with lower speed logic and less memory. The values at various points in this interleaved operation are shown in Table 1. The letter heading of each column refers to a point labeled in Fig. 6. The value α_{ij} is the j^{th} sample of the i^{th} spectrum of the signal from CM1. β_{ij} refers to the signal from CM2. X_{ij} , Y_{ij} , and Z_{ij} are the results of the *bf* operations on the signal α and R_{ij} , S_{ij} , and T_{ij} are the results of processing signal β . Previous and subsequent spectra would fill the table, but were deleted for clarity. The entries of the table clearly show that each spectrum, while interleaved, is kept separate in time. Each appears at the output as a separate block of data. The decommutation rate is $T/2$.

IV. Benefits and Costs of the Biplex FFT

The benefits of the Biplex FFT arise out of reduced memory size and processor speed. For the same bandwidth and resolution only 75 percent of the memory is required when compared to the in-place algorithm. To illustrate, if a 40-MHz bandwidth signal is to be transformed with a resolution of about 300-Hz it would require 2^{17} or 131,072 complex storage locations using the in-place algorithm. The same requirements can be met using $1\frac{1}{2} \times 2^{16}$ locations with the Biplex processor. (20-MHz bandwidth at ≈ 300 Hz resolution takes 2^{16} locations in-place or $1\frac{1}{2} \times 2^{16}$ pipelined). This saving in memory is accompanied by the saving of one *bf* processor.

The reduced sampling bandwidth of the Bipler technique reduces the memory and processor speed requirements by the same amount. For the example, the Bipler processor uses logic elements with delays of from 30 to 40 ns rather than the 15 to 20 ns of the more conventional processor.

These benefits are not gained without some cost. Since the full bandwidth is split in two, two analog-to-digital converter elements (complex mixers) are required. These

elements, however, operate at half the speed and, therefore, may be easier and less costly to implement.

V. Conclusions

The use of the Bipler FFT significantly reduces equipment costs when operating near the state-of-the-art speeds. It may also make pipeline techniques applicable and desirable at lower speeds where previously the in-place algorithm was considered most economical.

Reference

1. Rabiner, L. R., and Gold, B., *Theory and Application of Digital Signal Processing*, p. 573ff, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

Table 1. Intermediate contents of Biplex FFT

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
α_{30}														
α_{31}														
α_{32}														
α_{33}														
β_{30}	α_{34}	α_{30}	X_{30}	X_{34}	X_{30}									
β_{31}	α_{35}	α_{31}	X_{31}	X_{35}	X_{31}									
β_{32}	α_{36}	α_{32}	X_{32}	X_{36}	X_{34}	X_{32}	X_{30}	Y_{30}	Y_{32}	Y_{30}				
β_{33}	α_{37}	α_{33}	X_{33}	X_{37}	X_{35}	X_{33}	X_{31}	Y_{31}	Y_{33}	Y_{32}	Y_{31}	Y_{30}	Z_{30}	Z_{31}
	β_{34}	β_{31}	R_{30}	R_{34}	R_{30}	X_{36}	X_{34}	Y_{34}	Y_{36}	Y_{34}	Y_{33}	Y_{32}	Z_{32}	Z_{33}
	β_{35}	β_{32}	R_{31}	R_{35}	R_{31}	X_{37}	X_{35}	Y_{35}	Y_{37}	Y_{36}	Y_{35}	Y_{34}	Z_{34}	Z_{35}
	β_{36}	β_{33}	R_{32}	R_{36}	R_{34}	R_{32}	R_{30}	S_{30}	S_{32}	S_{30}	Y_{37}	Y_{36}	Z_{36}	Z_{30}
	β_{37}	β_{34}	R_{33}	R_{37}	R_{35}	R_{33}	R_{31}	S_{31}	S_{33}	S_{32}	S_{31}	S_{30}	T_{30}	T_{31}
						R_{36}	R_{34}	S_{34}	S_{36}	S_{34}	S_{33}	S_{32}	T_{32}	T_{33}
						R_{37}	R_{35}	S_{35}	S_{37}	S_{36}	S_{35}	S_{34}	T_{34}	T_{36}
											S_{37}	S_{36}	T_{36}	T_{37}

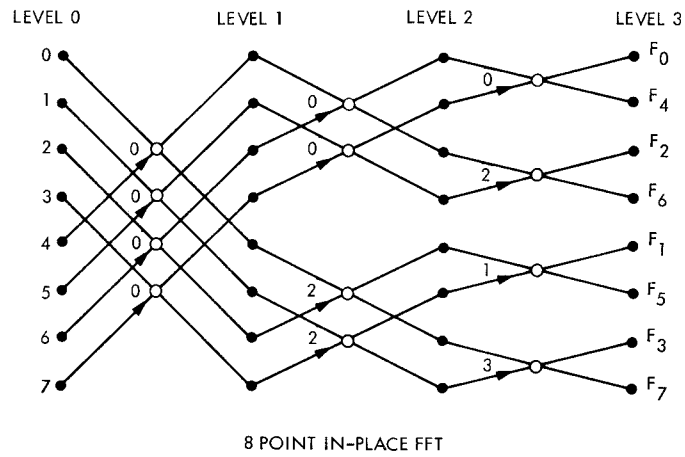


Fig. 1. Schematic of in-place FFT

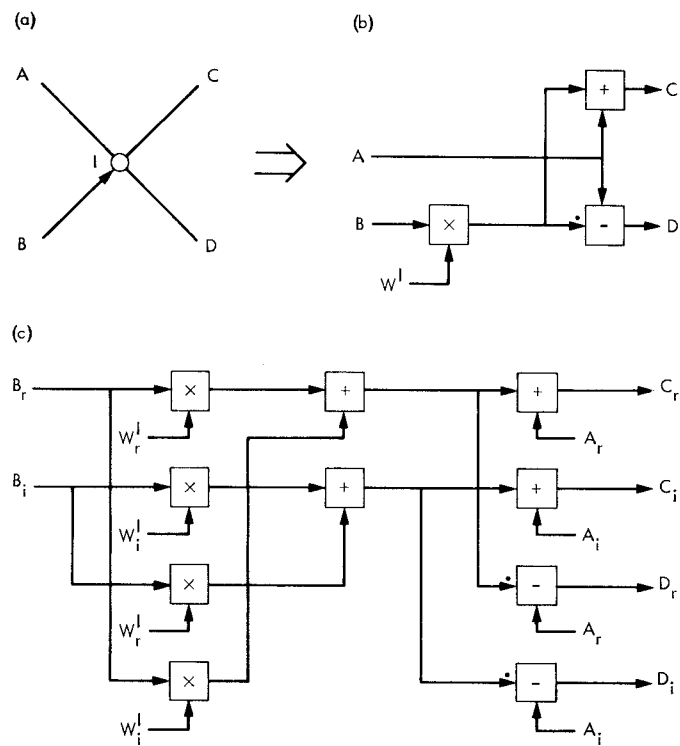


Fig. 2. Butterfly processor and expansion

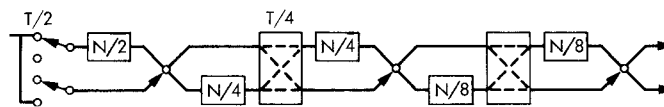


Fig. 3. Pipelined FFT processor

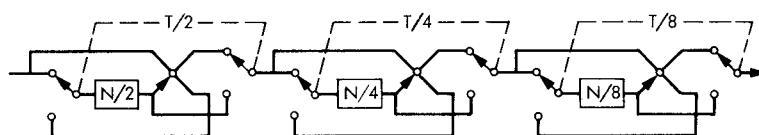


Fig. 4. Memory conservative pipelined FFT

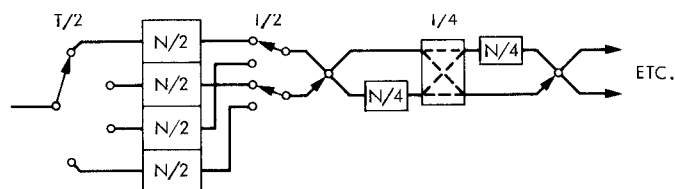


Fig. 5. Half-rate butterfly pipelined FFT

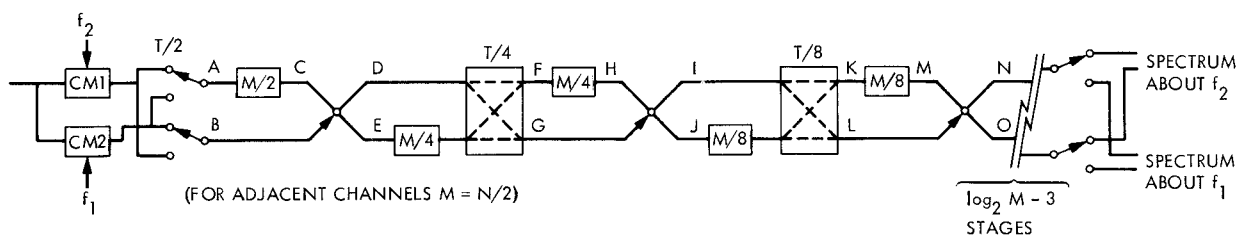


Fig. 6. Biplex FFT