IMPLEMENTATION OF "SPLIT-RADIX" FFT ALGORITHMS FOR COMPLEX, REAL, AND REAL-SYMMETRIC DATA

Pierre DUHAMEL and Henk HOLLMANN

CNET/PAB/RPE - 38-40, rue du Général Leclerc - 92131 ISSY-LES-MOULINEAUX (FRANCE)

ABSTRACT

A new algorithm is presented for the fast computation of the Discrete Fourier Transform. This algorithm belongs to that class of recently proposed 2^n-FFT's which present the same arithmetic complexity (the lowest among any previously published one). Moreover, this algorithm has the advantage of being performed "in-place", by repetitive use of a "butterfly"-type structure, without any data reordering inside the algorithm. Furthermore, it can easily be applied to real and real symmetric data with reduced arithmetic complexity by removing all redundancy in the algorithm.

I - INTRODUCTION

Since the early paper by COOLEY-TUCKEY [1], a lot of work has been done on FFT algorithm, and this has resulted in classes of algorithms such a radix-2^m algorithms, Winograd algorithm (WFTA) and Prime Factor Algorithms (PFA).

After several improvements [2,3], new algorithms for DFT computation on a set of 2^n data [7,8, 9,10] were proposed very recently.

This paper is concerned with the evaluation and implementation of one of these recent algorithms : the "split-radix" FFT algorithm [7], which seems to include the advantages of the recent methods, namely :

. The lowest number of multiplications, together with real-factor algorithms [2,3].

. The lowest number of additions among the 2^m algorithms [2,3,8,9,10].

. The same regularity as radix-4 algorithms.

. The same flexibility as radix-2 algorithms (useful for all $N=2^n$).

. No reordering of the data inside the algorithm.

. Possibility of in-place implementation for real and real symmetric data with reduced arithmetic complexity.

. Numerically as well conditioned as radix-4 algorithms.

After briefly recalling the split-radix algorithm, we describe how to increase the regularity of the initial algorithm by means of a permutation of the outputs of the computational cell. Afterwards, we show how the split-radix algorithm can be adapted to real and real symmetric data. Finally, we enlight some consequences of the characteristics of the presented algorithms : possible improvements of other transforms and compatibility of the different arithmetic complexities involved.

II - THE SPLIT-RADIX ALGORITHM

The "split-radix" algorithm is based on the following decomposition :

$$\text{if} \quad X_k = \sum_{n=0}^{N-1} x_n W_N^{nk}$$

is the DFT to be computed, it is decomposed into :

$$(1)\quad \left\{\begin{aligned} X_{2k} &= \sum_{n=0}^{N/2-1} \left(x_n + x_{n+\frac{N}{2}}\right) W_N^{2nk} \\ X_{4k+1} &= \sum_{n=0}^{N/4-1} \left[\left(x_n - x_{n+\frac{N}{2}}\right) + j\left(x_{n+\frac{N}{4}} - x_{n+3\frac{N}{4}}\right)\right] W_N^{n}\, W_N^{4nk} \\ X_{4k+3} &= \sum_{n=0}^{N/4-1} \left[\left(x_n - x_{n+\frac{N}{2}}\right) - j\left(x_{n+\frac{N}{4}} - x_{n+\frac{3N}{4}}\right)\right] W_N^{3n}\, W_N^{4nk} \end{aligned}\right.$$

The first stage of a split radix decimation in frequency decomposition then replaces a DFT of length N by one DFT of length N/2 and two DFT's of length N/4 at the cost of (N/2 - 4) general complex multiplications (3 real mult + 3 adds), and 2 multiplications by the eight root of unity (2 real mult+2 adds).

The length-N DFT is then obtained by successive use of such decompositions, up to the last stage where some usual radix-2 butterflies (without twiddle factors) are needed (see fig. 1 for a length 32-split-radix FFT).

III - "SPLIT-RADIX" ON COMPLEX DATA

III.1 Arithmetic complexity

Let M_n^C (resp. A_n^C) be the number of real multiplications (resp. additions) needed to perform a 2^n complex DFT with the split-radix algorithm. By eq. (1), we get :

$$M_n^C = M_{n-1}^C + 2\, M_{n-2}^C + 3.2^{n-1} - 8$$

and, with the initial conditions $M_1=0$, $M_2=0$ we obtain :

$$(2)\qquad M_n^C = 2^n\,(n-3) + 4$$

Discarding for a while the number of additions needed to perform the complex multiplications, the remaining ones can easily be evaluated as : $n.2^{n+1}$, since, at each of the n stages, a new point is generated by a complex addition. Then, since the number of real additions needed to compute a complex product is equal to the number of multiplications, we have :

$$(3)\qquad A_n^C = n.\, 2^{n+1} + M_n^C = 3.2^n\,(n-1) + 4$$

20.8.1

CH2118-8/85/0000-0784 $1.00 © 1985 IEEE

These quantities are computed in tables 1 and 2, and compared with classical algorithms (radix 2,4,8, real factors), and more recent one, by WANG [9], VETTERLI-NUSSBAUMER [8], and MARTENS [10].

Observing tables 1 and 2 shows that the split-radix algorithm has the lowest number of both multiplications and additions, together with FFCT [8] and RCFA [10]. (The number of operations given in [10] has been corrected in tables 1 and 2 to take into account the fact a multiplication by $\frac{\sqrt{2}}{8}$ 1 needs only 2 real multiplications and 2 real additions).

III-3 Implementation in the case of complex data

From eq. (1), it is quite obvious that the split-radix algorithm can be performed in-place, by repetitive use of the "butterfly" type of structure given in fig. 2.

Furthermore, the minimum number of arithmetic operations is obtained with only 4 types of butterflies : the general one (6 real mult), as given in fig. 2, plus two special cases : k=0 (without multiplications), and k = N/8 (4 real multiplications). The 4^{th} one is a usual radix 2 butterfly (without multiplication).

The implementation of the algorithm described in fig. 1 is based on the fact that, at each stage n°i of the algorithm, the butterflies are always applied in a repetitive manner to blocks of length $N/2^i$.

Neglecting the overhead (computation of the multiplicative constants and of the address of the blocks) gives a run-time of 92.2 ms for a 1024-point FFT, versus 100.3 ms for the same FFT computed with the compact radix-4 program by MORRIS [14]. (time-averages over 200 runs on a Honeywell-Bull DPS 8).

Furthermore, careful inspection of the program shows that the number of each floating-point operation (load, store, addition, substraction, multiplication) is lower in split-radix than in radix-4 (table 3). This fact tends to indicate that this program should run faster than radix-4 on any machine (for comparable implementations).

For some specific applications it may be useful to increase the regularity of the algorithm : this can be done by the use of a butterfly with permuted outputs, as shown in fig. 3.

In that case, the length-32 split-radix diagram becomes as shown in fig. 4. The price that has to be paid for this increase in the regularity of the program is twofold : first, the outputs are not in bit-reversed order anymore, and in-place reordering is not possible. Secondly, this permutation induces more floating-point loads and stores.

In fact, it turns out that, for a software implementation, the resulting program is notably slower (96.7 ms for a 1K DFT). Nevertheless, the regularity of the resulting diagram may be attractive for hardware implementation.

IV - "SPLIT RADIX" ALGORITHM ON REAL DATA

Redundancy reduction is very easy when a split-radix FFT algorithm is applied to real signals :

In fact, when the sequence x_n is real, X_k and X_{N-k} are complex conjugates. When reported in eq. (1), this means that it is useless to compute both X_{4k+1} and X_{4k+3} , since

$$\left\{ X_{4k+3} \right\} = \left\{ X_{N-(4k'+1)} \right\} = \left\{ X_{4k'+1} \right\}^*$$

IV-1 Arithmetic complexity

In the case of real signals, eq. [1] now transforms a real DFT of length 2^n into a real DFT of length 2^{n-1} (to get X_{2k}), plus a complex DFT of length 2^{n-2} (to get X_{4k+1}), at the cost of $3(2^{n-2}-2)+2$ real multiplications and $3(2^{n-2}-2)+2$ real additions due to the twiddle factors, plus 2^n additions.

Thus, we can get :

$$(4) \quad M_n^{\ r} = 2^{n-1}(n-3)+2 = M_n^{\ c} / 2$$

$$(5) \quad A_n^{\ r} = (3n-5)\, 2^{n-1} + 4$$

A remark that will be useful later is that the multiplicative complexity is exactly half of the one in the complex case, while the number of additions is less than one half of the corresponding one in the complex case. Here again, the split-radix algorithm is seen to have the same arithmetic complexity as FFCT [13], and significantly less operations than the algorithm by Bergland [17].

IV-2 Implementation

As we intend to perform the split-radix algorithm "in-place" on real data, we need a schematic representation of the way the algorithm works in the different stages and of the place the intermediate results are stored in.

To make this representation clearer, let us begin with the split-radix on complex data, as shown in fig. 5a) for a 16-point FFT.

This diagram can be explained as follows :

On stage 0, the butterflies are applied to one block of length N, and this block is useful in the computation of every transformed sample X_k, k=0,---15. After this first step, we have now, as given in eq. (1) one block of length N/2, belonging to stage 1, from which the even points X_{2k}, k=0, --- 7 are computed, and two blocks of length N/4, hence belonging to stage 2, used in the computation of X_{4k+1}, k=0, --- 3 and X_{4k+3} k=0, --- 3 respectively. This process is then applied recursively to all blocks of length $N/2^i$, appearing at stage i.

When the split-radix algorithm is applied to real data, the block of stage 0 is real, inducing a block of length N/2 at stage 1, which is also real. The other two blocks, of length N/4, at stage 2, are complex, but, as noticed in § IV-1, X_{4k+3} needs not be computed, so that the corresponding locations in the diagram can be used to store the imaginary part of the block used to compute X_{4k+1} . This process is repeated until the whole transform is obtained.

V - "SPLIT-RADIX" ALGORITHM ON REAL SYMMETRIC DATA

Redundancy reduction is also possible in real "split-radix" algorithm in the case of real symmetric entries, giving rise to an "in-place" algorithm with the following complexities.

$$(6) \quad M_n^{\ rs} = 2^{n-2}(n-3)+1 = M_n^{\ r} / 2$$

$$\text{and : } A_n^{\ rs} = 2^{n-2}(3n-7)+n+3$$

VI - CONSEQUENCES

A number of different algorithms can be revisited and improved by the use of split radix decompositions, namely DCT and odd DFT's (to be used in 2-D DFT computation by polynomial transforms).

More important consequences can be established from the consideration of the arithmetic complexities of the different algorithms involved :

In fact, it appears that all the considered algorithms (DFT's on complex, real, real symmetric data, odd DFT's, DCT) are now of compatible complexity : i.e. an improvement on any of these algorithms allows us to derive better ones for the others, since they can be expressed mutually :

a)	complex DFT 2^n	2 real DFT 2^n
		+ 2^{n+1} additions
	real DFT 2^n	real DFT 2^{n-1}+complex DFT 2^{n-2}
		$(3.2^{n-2} - 4)$ multiplications
		$(2^n + 3.2^{n-2} - 4)$ additions
b)	real DFT 2^n	1 real symmetric DFT 2^n
		+ 1 real antisymmetric DFT 2^n
		+ $(2^n - 2)$ additions
	real symmetric DFT 2^n	1 real symmetric DFT 2^{n-1}
		+ 1 inverse real DFT
		+ $3(2^{n-3} - 1)\pm 1$ multiplications
		+ $(3n - 4)2^{n-3} + 1$ additions
c)	real DFT 2^n	1 real DFT 2^{n-1}
		+ 2 DCT 2^{n-2}
		+ $(3.2^{n-1} - 2)$ additions
	DCT 2^n	1 real DFT 2^n
		+ $(3.2^{n-1} - 2)$ multiplications
		+ $(3.2^{n-1} - 3)$ additions
d)	complex DFT 2^n	1 complex DFT 2^{n-1}
		+ 1 odd DFT 2^{n-1}
		+ 2^n additions
	odd DFT 2^{n-1}	2 complex DFT's 2^{n-2}
		+ $2(3.2^{n-2} - 4)$ multiplications
		+ $(2^n + 3.2^{n-1} - 8)$ additions

Furthermore, a comparison has been made in [13] between the number of true complex multiplications (i.e. $\neq \; j = W_N^{N/4}$) in a complex split-radix algorithm, and in a method based on a decomposition of the DFT into a number of polynomial products, these polynomial products being themselves computed with optimum algorithms.

These results tend to indicate that all these algorithms described above could be optimum (as far as the number of multiplications is concerned) up to length 64, and optimum in a property defined subclass for longer transforms, since the divergence between both multiplicative complexities, begins to appear exactly when the degrees of the polynomial products involved are too high for the optimum polynomial products algorithms to be of practical interest.

VII - CONCLUSION

New algorithms were proposed for FFT computation on complex, real, and real symmetric data, with possible application to DCT, odd DFT's and polynomial transforms.

Although obtained only by redundancy reduction in the complex split-radix algorithms, they were shown to have the same arithmetic complexity as the best algorithms recently proposed, while having the advantage of a more regular implementation.

Furthermore, there are strong indications that they could be optimum (as far as the number of multiplications is concerned) in a properly defined subclass of algorithms.

REFERENCES

[1] J.W. COOLEY, J.W. TUKEY : "An algorithm for machine computation of complexFourier series". Math. Comput., 1965, vol. 19, pp. 297-301.
[2] C.M. RADER, N.M. BRENNER : "A new principle for fast Fourier Transformation". IEEE trans. ASSP., 1976, vol. 24, pp. 264-265.
[3] R.D. PREUSS : "Very fast computation of the Radix-2, Discrete Fourier Transform". IEEE Trans. ASSP., 1982, vol. 30, pp. 595-607.
[4] H.J. NUSSBAUMER : "Fast Fourier Transform and convolution Algorithm". Springer Verlag., 1981.
[5] H. JOHNSON, C.S. BURRUS : "Twiddle factors in the radix-2 FFT". Proc. of the 1982 Asilomar Conference on CIrcuits Systems and Computers, pp. 413-416.
[6] L.R. MORRIS : "Automatic Generation of Time Efficient Digital Signal Processing Software". IEEE Trans. on ASSP., vol. ASSP-25, n°1, pp. 74-79, Feb. 1977.
[7] P. DUHAMEL, H. HOLLMANN : "Split-radix FFT algorithm". Electronics Letters, vol. 20, n°1, pp. 14-16, Jan. 1984.
[8] M. VETTERLI, H.J. NUSSBAUMER : "Simple FFT and DCT algorithms with reduced number of operations". Signal Processing, vol. 6, n° 4, pp. 267-278, July 1984.
[9] Z. WANG : "Fast Algorithms for the Discrete W Transforms and the Discrete Fourier Transform". IEEE Trans. on ASSP., vol. ASSP-32, n°4, pp. 803-816, Aug. 1984.
[10] J.B. MARTENS : "Recursive cyclotomic factorization - A new algorithm for calculating the Discrete Fourier Transform". IEEE Trans. on ASSP., vol. ASSP-32, n°2, pp. 750-761, April 1984.
[11] G.D. BERGLAND : "A fast Fourier Transform algorithm for real-valued series". Commun. ACM., vol. 11, pp. 703-710, Oct. 1968.
[12] H. ZIEGLER : "A Fast Fourier Transform Algorithm for symmetric real-valued series". IEEE Trans. Audio Electroacoust., vol. A 4-20, n° 5, pp. 353-356, Dec. 1972.
[13] P. DUHAMEL, H. HOLLMANN : "Existence of a 2^n-FFT algorithm with a number of multiplications lower than 2^{n+1}". Electronics Letters, vol. 20, n° 17, pp. 690-692, Aug. 1984.
[14] L.R. MORRIS : Private Communication.
[15] M. VETTERLI : "FFT's of signals with symmetries and application to autocorrelation computation". Submitted to MELECON 85. Madrid.

N	radix 2	radix 4	radix 8		WANG [14]	FFCT [13]	RCFA [15]	split radix
16	24	20		20	20	20	20	20
32	88			68	68	68	68	68
64	264	208	204	196	204	196	196	196
128	712			516	564	516	516	516
256	1800	1392		1284	1468	1284	1284	1284
512	4360		3204	3076	3652	3076	3076	3076
1024	10248	7856		7172	8780	7172	7172	7172
2048				16388	20564	16388	16388	16388

Table 1 : number of non-trivial real multiplications to compute a length-N complex DFT

N	radix 2	radix 4	radix 8	Rader Brenner [2]	WANG [14]	FFCT [13]	RCFA [15]	split radix
16	152	148		148	148	148	148	148
32	408			424	388	388	388	388
64	1032	976	972	1104	972	964	964	964
128	2504			2720	2356	2308	2308	2308
256	5896	5488		6464	5564	5380	5380	5380
512	13566		12420	14976	12868	12292	12292	12292
1024	30728	28336		34048	29260	27652	27652	27652
2048				76288	65620	61444	61444	61444

Table : 2 number of real additions to compute a length-N complex DFT.

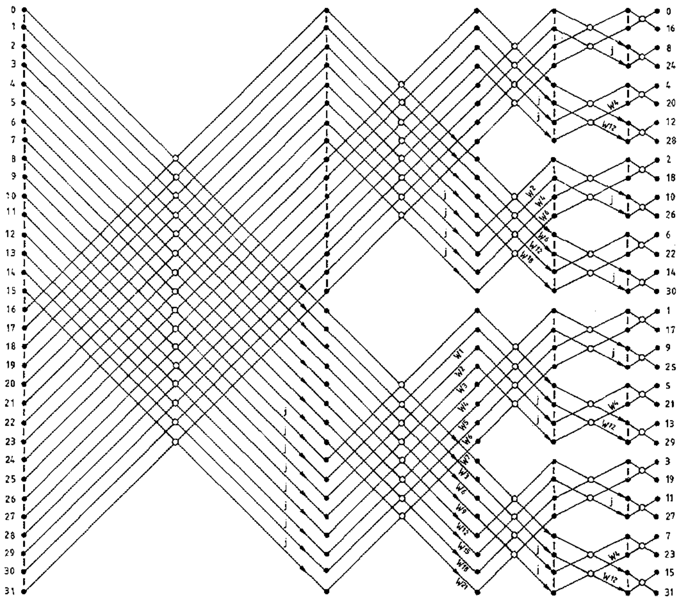

Fig. 1 : length 32 split-radix diagram

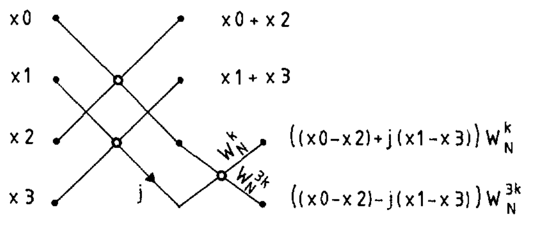

Fig. 2 : the "butterfly" used in the split. radix algorithm

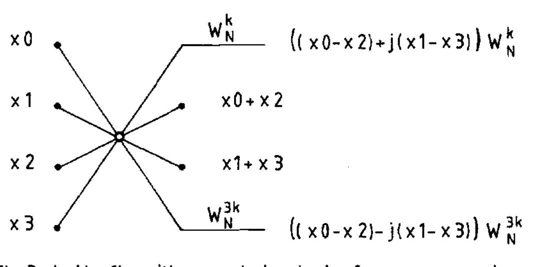

Fig. 3 : butterfly with permuted outputs for a more regular "split. radix" algorithm

	FFT 4	split radix
fld	22 884	21 280
fstr	28 506	27 652
fad	19 120	18 749
fsab	9 386	9 073
fmp	8 026	7 342

Table 3 : number of floating-point operations in a 1024 point FFT.

(Estimations based on the assembler generated by the fortran compiler on a Honeywell-Bull dps 8 under MULTICS operating system).

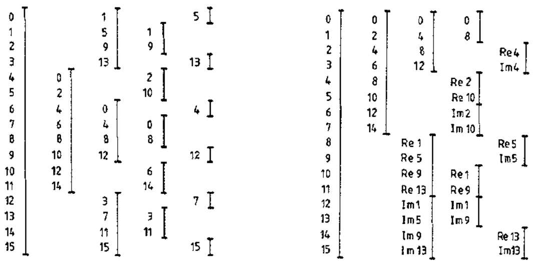

Fig. 6 : schematic representation of a 16.point split radix algorithm with increased regularity
a) on complex data b) on real data

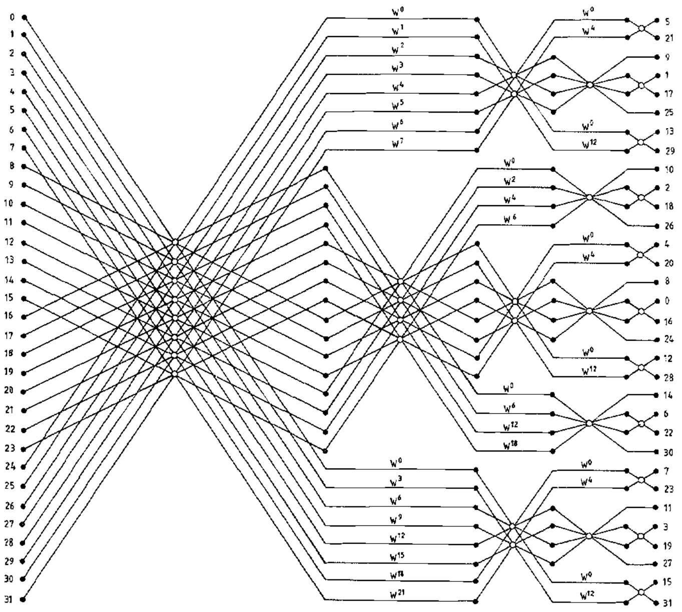

Fig. 4 : symmetric "split. radix" diagram.

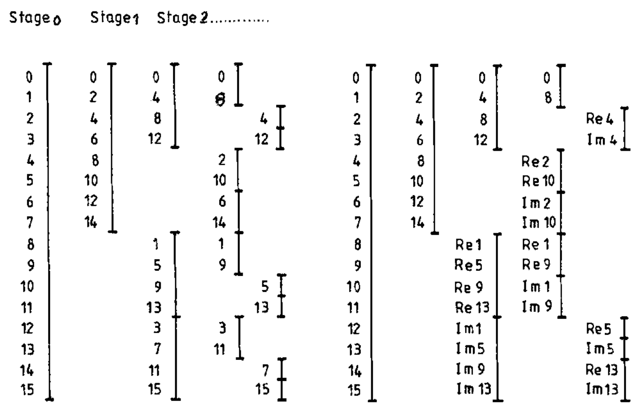

Fig. 5 : schematic representation of a 16.point split radix algorithm
a) on complex data b) on real data