SPLIT MULTIPLE RADIX FFT

Ryszard Stasinski
Institute of Multimedia Telecommunications
Poznan University of Technology
Poznan, Poland
ryszard.stasinski@put.poznan.pl

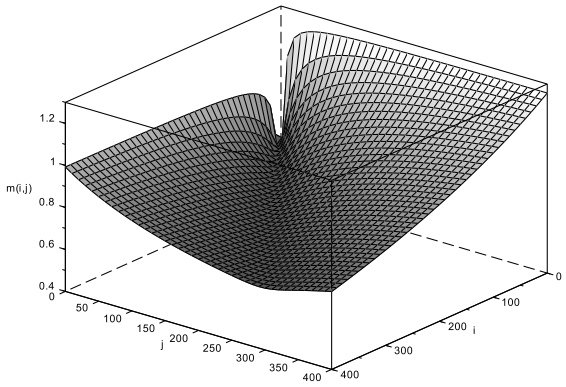

Fig. 1. Function $m(i,j)$ for radix-6:2&3 C-FFT.

***Abstract*—In the paper general rules for construction of split-radix FFTs having multiple auxiliary bases is presented. The algorithms exist for DFT sizes being products of mutually prime numbers. The algorithms have smaller arithmetical and multiplicative complexities than simpler FFTs, parameters allowing comparison are also introduced. The obtained results are remarkable, some of FFTs have smaller arithmetical, or at least multiplicative complexities than "standard" split-radix FFT. When compared to other algorithms for such N savings in the numbers of arithmetical operations emerge even for the smallest useful DFT sizes. The presented techniques provides new tools for construction of optimized FFTs, e.g. DFT modules for very large radix algorithms generated by automatic software, like FFTW.**

***Index Terms*—algorithms, DFT, FFT, split-radix**

I. Introduction

The discovery of Fast Fourier Transform algorithm (FFT) [1] promted many publications on algorithms for the Discrete Fourier Transform (DFT), including two special issues on this subject of IEEE AU Transactions [2]. If restricting our attention to FFT, the next important achievment was introduction of split-radix FFT [3], which improved version was published in [4]. Then, in [5] split-radix FFT with more than one auxiliary base was presented, as it is shown in Table 1 (parameter a_T), the algorithm has the smallest arithmetical complexity among FFTs. Its multiplications are defined in μ base, which leads to operation savings in radix-3, -6, and -12 FFTs [6].

Support: Ministry of Science and Education grant 0314/SBAD/0210.

The paper is motivated by a striking behaviour of recursive equation solution:

$$M(N) = N + M(N/2) + M(N/3) + M(N/6) \qquad (1)$$

being simplified formula on the FFT number of multiplications for an improved from this point of view version of [5] from [7], $N = 2^i \cdot 3^j$. Edge conditions are:

$$M(N) = N + M(N/2) + 2 \cdot M(N/4)$$

for $N = 2^i$ (split-radix-4:2 FFT, radix 2/4 in [3]),

$$M(N) = 2N + 3M(N/3)$$

for $N = 3^j$ (radix-3 FFT from [7]), $M(0) = M(2) = 0$. Figure 1 shows function $m(i,j)$ for this equation:

$$m(i,j) = \frac{M(2^i \cdot 3^j)}{2^i 3^j \log_2(2^i 3^j)} \qquad (2)$$

Note that visualized in the same way function for non-split-radix FFT for $N = 2^i 3^j$ forms a relativey flat surface, here the function is concave with bottom values much below those at the edges. Namely, for split-radix-4:2 FFT $m(i,0) \approx 1$ (edge algorithm), while here its values go down to approximately 0.7, and diminish even further with growing N. This behaviour of recursive equations results in different values of a_M and a_T parameters for FFTs radix-6μ, and radix-6μ:2&3 in Table 1, compare also three tadix-10T algorithms there. The depression goes along a line which direction is indicated by proportion of $i : j$ close to $4 : 3$, see parameter b in Table 1.

It is necessary to explain here the convention for denoting split-multiple-radix FFTs used in this paper, as current notation is impossible to implement:

radix-$K : M\&M'\& \ldots$,

where K is the main radix, and $M, M', \ldots$ are the auxiliary ones, defined as in section III. Then, the split-radix FFT is radix-4:2 one [3], FFTs from [5], [7] are radix-6:2&3 ones, and an ordinary radix-K FFT can be denoted as radix-K:1.

The paper is organized as follows: Firstly, it is explained why split-radix FFTs have smaller arithmetical complexity than the regular algorithms, section II. In the following section derivation of split-multiple-radix FFTs is presented, section III, followed by a section on complexity measures for the algorithms, section IV. Then, results for some FFT radices are provided, Table 1, section V. Exemplary radix-5:2&5 algorithm is presented in Figure 3. It is shown that indeed, the new split-multiple-radix FFTs are highly efficient, and some are even better than "standard" split-radix one [3].

II. Background

Fast Fourier Transform algorithm is derived using "divide and conquer" technique. The derivation is unbalanced [10]. Namely, let us assume that DFT size N is divisible by a number K. Then, computation of DFT samples $X(Kk), k = 0, 1, \ldots, N/K-1$, is computationally simpler than those for $X(Kk+i), i = 1, 2, \ldots, K-1$, [2]:

$$X(Kk) = \sum_{n'=0}^{N/K-1} [\sum_{n''=0}^{K-1} x(n''N/K + n')] W_{N/K}^{kn'} \quad (3)$$

$$X(Kk+i) = \sum_{n'=0}^{N/K-1} W_N^{in'} [\sum_{n''=0}^{K-1} x(n''N/K+n') W_K^{in''}] W_{N/K}^{kn'} \quad (4)$$

It seems that when $N = p^r$, p is prime, then the best choice for FFT radix is $K = p$. Note, however that the inner sum in (4) is the K-point DFT, and as there exist optimized DFT algorithms of small size, for some p better algorithms are obtained when K is a power of p. This observation led to split-radix FFTs: they have two bases, the auxiliary one $M = p$ for computing samples $X(Mk)$, and the primary one K for computing $X(Kk + i)$, this time index i takes on all values that are mutually prime with K[1].

Even more interesting is the case when $N = p^r q^s$, q is a prime, too. In such situation it seems that we should do a compromise, choose as a secondary base either p, or q. In [5] an algorithm for $p = 2, q = 3$ was presented in which both secondary bases were used, which resulted in an exceptionally low arithmetical complexity. The technique was explained in [7], where additionally the version of the algorithm with reduced multiplicative complexity was presented. This paper presents generalization of the idea.

III. Derivations

Computation of DFT for a series $x(n)$ can be interpreted as reduction of polynomial $X(Z)$ modulo all divisors of a polynomial $Z^N - 1$ [8]:

$$X(k) = X(Z) \mod (Z - W_N^k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad (5)$$

$$k = 0, 1, \ldots N-1;\ X(Z) = \sum_{n=0}^{N-1} x(n) Z^n;$$

W_N is a primitive root of unity of rank N. The computation process can be broken up into two stages: Firstly, residue polynomials modulo $Z^{N'} - 1$, and $P(Z) = (Z^N - 1)/(Z^{N'} - 1)$ are computed, N' is a divisor of N:

$$\begin{aligned} X_0(Z) &= X(Z) \mod Z^{N'} - 1, \\ X_1(Z) &= X(Z) \mod P(Z). \end{aligned} \quad (6)$$

Then, they are reduced in accordance with (5). Note that reductions modulo divisors of $Z^{N'} - 1$ are equivalent to computation of DFT of size N', the other operations are named N-point DFT reduced modulo $P(Z)$ [8]. The actual size of such DFT is $N - N'$. Namely, they lack reductions modulo $P(Z)$ (data are already reduced), reductions modulo $Z^{N'} - 1$ and the following them N'-point DFT, see equations (8). Reductions modulo $Z^{N'} - 1$ are trivial:

$$x_0(n') = \sum_{n''=0}^{N/N'-1} x(n' + n'' \cdot N'),\ n' = 0, 1, \ldots N' - 1, \quad (7)$$

$x_0(n')$ are coefficients of $X_0(Z)$. Note that for $K = N/N'$ (7) is equal to inner sum in (3), hence, samples $X(0)$ of K-point DFTs inside FFT butterflies can be interpreted as coefficients of polynomial $X_0(Z)$ when $N' = N/K$.

Let us consider derivation of radix-K FFT, in which $N/N' = M < K$. Input samples to n'-th K-point DFT in (3), (4) are:

$$x(n' + n'' \cdot N/K),\ n'' = 0, 1, \ldots K - 1.$$

A comparison with (7) shows that reductions modulo $Z^{N'} - 1 = Z^{N/M} - 1$ for the whole DFT end before computation of samples $X(0)$ of K-point butterfly DFTs, they form reductions modulo $Z^{K/M} - 1$ inside them. Split-radix butterfly is formed when operations of K-point DFTs following reductions modulo $Z^{K/M} - 1$ are removed.

For example, butterfly of the radix-10 FFT contains 10-point DFT, which can be mapped into 2×5-point one, i.e. 2-point DFTs go first. 2-point DFTs contain additions $x(n) + x(n + 5)$, which work as reductions modulo $Z^{10/2} - 1 = Z^5 - 1$, hence, radix-10:2 butterfly is obtained by removing 5-point DFT following these operations, it lacks in Figure 3.

Up to now it was not necessary to implement reductions modulo $P(Z)$ (6), calculations in accordance with (4) did the job. Moreover, if K is odd, structures linked with this reduction slightly increase the number of additions of the algorithm, see [7], where the reductions are implemented. However, when K is a product of mutually prime numbers, e.g. for N divisible by 6, parallel implementation of reductions modulo $Z^{N/2} - 1$, and $Z^{N/3} - 1$ leads to computation of DFT samples $X(6k)$ twice. Solution consists in puting reduction modulo $Z^{N/M'} - 1$ for the second base M' behind that modulo $P(Z)$ [5], [7], in this way reductions for the second base are done modulo $P'(Z) = \gcd\{P(Z), Z^{N/M'} - 1\} = (Z^{N/M'} - 1)/(Z^{(N/M')/M} - 1)$. As a result the DFT following the operations is N/M'-point one reduced modulo $P'(Z)$.

There are two types of butterflies of such split-radix-K FFTs: the basic ones start with reductions modulo $Z^{K/M} - 1$, and $(Z^K - 1)/(Z^{K/M} - 1)$, which stage is obsolete in butterflies beginning the N/M'-point DFT reduced modulo $P'(Z)$. Then, the second reductions are followed by (or the butterfly start with) K-point DFT reduced modulo $(Z^K - 1)/(Z^{K/M} - 1)$, and rotation factors at the end. Structures following reductions modulo $Z^{K/M} - 1$, $Z^{K/M'} - 1$ are removed. The above construction can be repeated for next auxiliary bases, e.g. for $K = 30$ possible M are 2, 3, and 5. In such case reductions modulo $Z^{K/M'} - 1$ are done in parallel

[1]For $p = 2$ all algorithms for M being a power of 2 smaller than the power of 2 for K have the same arithmetical complexity.

with modulo $(Z^K-1)/(Z^{K/M'}-1)$ ones, then follow those for $Z^{K/M''}-1$ and "doubly reduced" K-point DFTs.

Coming back to Figure 3, the other operations of 2-point DFTs are $x(n)-x(n+5)$, which are equivalent to reductions modulo $(Z^{10}-1)/(Z^5-1)=Z^5+1$, hence, the following 5-point DFT is equivalent to reduced modulo Z^5+1 10-point DFT. Reductions modulo $Z^{10/5}-1=Z^2-1$ consist in computation of sums $x(i)+x(2+i)+x(4+i)+x(6+i)+x(8+i)$, $i=0,1$, which is equivalent to computations of samples $X(0)$ of 5-point DFTs when 10-point DFT is organized as 5×2 one, i.e. in which 5-point DFTs come first. Note that sample $X0$ is equal to difference: the sum for $i=0$ minus that for $i=1$, which means that indeed, both reductions Z^2-1 and Z^5+1 are done. As every DFT algorithm for power of 2 starts with sums $x(n)\pm x(n+K/2)$, equivalent to reductions modulo $Z^{K/2}\pm 1$, a perfect value for p is 2. This stage is absent in initial butterflies of reduced radix-10:2&5 FFTs, compare [7], where radix-6:2&3 and radix-6:3&2 FFTs are presented.

IV. ASYMPTOTIC EVALUATIONS

Let us assume that an FFT main radix is $K=p^s\cdot q^t$, where p and q are mutually prime, $s,t\geq 1$. Then, a simplified recursive equations for the number of operations $T(N)$ for the N-point radix-K:$p\&q$ FFT is, compare [7]:

$$\begin{aligned} T(N)= &\ \gamma N+T(N/p)+T'(N/q)+\\ &+p^{s-1}q^{t-1}(p-1)(q-1)T(N/p^sq^t)\\ T'(N)= &\ \gamma' N+T'(N/q)+\\ &+p^{s-1}q^{t-1}(p-1)(q-1)T(N/p^sq^t)=\\ &=(\gamma'-\gamma)N+T(N)-T(N/p) \end{aligned} \tag{8}$$

where $T'(N)$ is for the reduced algorithm. Note that

$$T'(N)=(\gamma'-\gamma)N+T(N)-T(N/p) \tag{9}$$

hence,

$$\begin{aligned} T(N)= &\ cN+T(N/p)+T(N/q)-T(N/pq)+\\ &+p^{s-1}q^{t-1}(p-1)(q-1)T(N/p^sq^t) \end{aligned} \tag{10}$$

where:

$$c=\frac{(q-1)\gamma+\gamma'}{q}$$

Now assume that:

$$T(N)=a_T N\log_2 N \tag{11}$$

Substitution of (11) for $T(N)$ in (10) leads to the following expression on asymptotic coefficient a_T:

$$\begin{aligned} &a_T=\\ &=\frac{cpq}{[s(p-1)+1](q-1)\log p+[t(q-1)+1](p-1)\log q} \end{aligned} \tag{12}$$

It is interesting to find a formula on N for which (11) holds (direction of depression in Figure 1). Let us consider a simpler version of the algorithm for $s=t=1$ (10):

$$T(N)=cN+T(N/p)+T(N/q)+(d-1)T(N/pq), \tag{13}$$

$d=(p-1)(q-1)$. Let $N=p^vq^w$, $v>>w$. It can be proven that for large N the formula on the algorithm complexity is:

$$T(p^vq^w)=[av+(c-\frac{q-1}{q}a)\frac{p}{p-1}w]p^vq^w \tag{14}$$

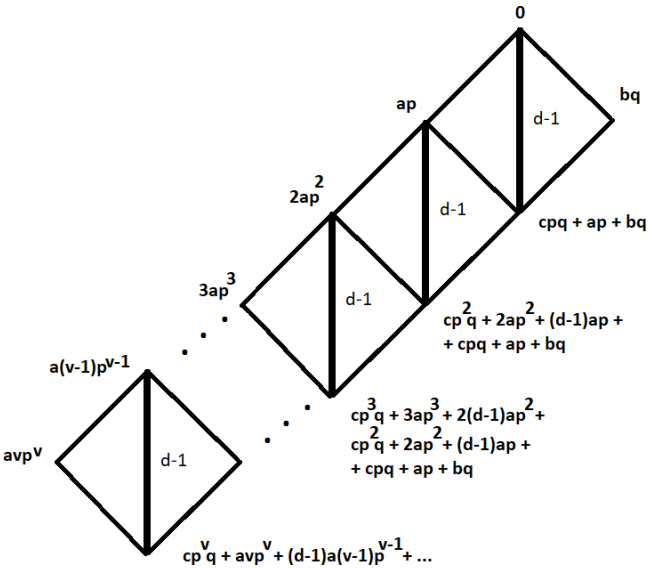

Fig. 2. Graphical representation of derivation of (15).

Notation and the first step of derivation (for $w=1$) is illustrated in Figure 2, the edge algorithm complexity is $aN\log_p N=avp^v$, the formula for the other edge is bwq^w. Then, the number of operations for the algorithm is:

$$T(p^vq)=cq\sum_{i=1}^{v}p^i+avp^v+ad\sum_{i=1}^{v-1}ip^i+bq \tag{15}$$

The next step of derivation consists in transforming (15) into (14) for $w=1$ by ignoring all terms $o(p^v)$. Then, we assume that (14) holds for all transform sizes for $w-1$, do computations as in Figure 2, and ignore terms $o(p^v)$. Equation for algorithms close to other edge is analogous, p,v,q,w,a should be replaced by q,w,p,v,b, respectively. The equations depict asymptotic surfaces to which nears solution of equation (13) for very large v and w. They meet when:

$$\frac{v}{w}=\frac{(q-1)p}{(p-1)q} \tag{16}$$

for which proportion algorithm complexity appears to be:

$$T(N)=\frac{cpq}{p(q-1)\log p+q(p-1)\log q}N\log N$$

We recognize here a_T value when $s=t=1$ (12). That is why in Table 1 b (base) parameter is introduced:

$$b=2^{\frac{c}{a_T}} \tag{17}$$

which probably indicates direction of depression not only when $K=pq$. For example, in equation (1) $c=1$, which results in:

$$a_M=\frac{6}{4+3\log 3}=0.68533...$$

$$b=2^{(4+3\log 3)/6}=(2^4\cdot 3^3)^{1/6}$$

see Figure 1.

In the case of radix-K:p&q&r FFT for $K = p^s \cdot q^t \cdot r^u$ recursive equations on the number of operations are:

$$\begin{array}{l} T(N) = \gamma N + T(N/p) + T'(N/q) + T''(N/r) + \\ +p^{s-1}q^{t-1}r^{u-1}(p-1)(q-1)(r-1)T(N/p^s q^t r^u) \\ T'(N) = (\gamma' - \gamma)N + T(N) - T(N/p) \\ T''(N) = (\gamma'' - \gamma)N + T(N) - T(N/p) - T'(N/q) \end{array} \tag{18}$$

compare (9), $T''(N)$ is for "doubly reduced" transform. The resultant expression on a_T contains fraction with very long denominator, hence, we provide it in a non-direct form:

$$\begin{array}{rl} \frac{c}{a_T}pqr = & [s(p-1)+1](q-1)(r-1)\log p + \\ & +[t(q-1)+1](p-1)(r-1)\log q + \\ & +[u(r-1)+1](p-1)(q-1)\log r \end{array} \tag{19}$$

where

$$c = \frac{(q-1)(r-1)\gamma + (r-1)\gamma' + q\gamma''}{qr}$$

V. RESULTS

In Table 1 arithmetical (a_T) and multiplicative (a_M) complexities of few split-multiple-radix FFTs are compared to those for reference FFTs: radix-2 and 4, split-radix [3], improved split-radix [4], denoted radix-4F:2, and the best non-split-radix-6 FFT, denoted 6μ [6]. FFT radix-4F:2 is the best algorithm for $N = 2^i$, hence, its parameters are in bold for easy comparison. The lowest a_T value is in bold, too. Asymptotic coefficients are obtained from coefficients c of recursive equations, calculated from numbers of multiplications (mults) and additions for FFT butterflies. As reduced and doubly reduced butterflies have smaller number of additions (adds', adds"), the c for the total number of arithmetical operations (total) is computed from γ coefficients, plus c for multiplications, see section IV. Formulas on base (b) indicate N, for which the algorithms are the best.

Two versions of algorithms are provided: optimized for the number of multiplications, and another optimized for the number of arithmetical operations. DFT algorithms are constructed using DFT modules from [8], Winograd nesting is applied, T means T-FFT [10], used in the last six algorithms 3-point reduced DFT is shown in [7].

Possible improvements are explained in Figure 3 for the radix-10:2&5 T-FFT. Basic idea of T-FFT is as follows [10]: When in derivation of radix-K decimation-in-frequency FFT index i in (4) is defined as follows (K can be odd, too):

$$i = -(K/2-1), \ldots, 0, 1, \ldots K/2,$$

pairs of complex conjugated rotation factors are obtained. Then, by combining them with output DFT operations $\mathbf{R}(\alpha)$ operations can be formed. In Figure 3 samples $\mathbf{X}_i$, $i = -3, -1, 1, 3$ are inputs to DFTs computing samples $X(Kk+i)$. $\mathbf{R}(\alpha)$ and $\mathbf{M}$ have the same form:

$$\begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

where for $\mathbf{M}$ $\alpha = 3\pi/10$, and for $\mathbf{R}(\alpha)$ $\alpha = 2\pi n/N$. They are computed by either 3 multiplications and 3 additions, or 4 and 2 ones. However, if $\mathbf{M}$ is divided by e.g. $\cos 3\pi/10$ (and $\mathbf{R}$ operations partly multiplied by it), $\mathbf{M}$ can be computed using 2 multiplications and 2 additions, only. Similarly, upper multipliers of DFT can be divided by m, and in this way the third scalar multiplier saved. Analogous optimizations can be done in butterflies of other radix-$2^k p$ FFTs, p is prime.

According to (4) in split-radix-10T:2 (and in "plain" radix-10T) FFT butterflies sample $X0$ in Figure 3 is multiplied by rotation factor W_N^{5n}, and followed by $N/10$-point DFT. Lack of this multiplier in radix-10T:2&5 butterflies results in reduced numbers of parameters a_T, a_M, hence, reduced arithmetical complexity of the algorithm, Table 1. This is despite the fact that the sample is followed by nominally greater DFT (but not actually). This saving emerges even for the smallest useful size of the FFT being $N = 50$. "Disappearing" rotation factors are characteristic of split-multiple-radix FFTs.

As can be seen, majority of presented in Table 1 split-multiple-radix FFTs have smaller multiplicative and arithmetical complexities than the reference FFTs, possibly except for the radix-4F:2 one. Moreover, there are some that in every respect are better than radix-4F:2 FFT: radix-K for $K = 2^k 3$, and radix-80:2&5. This is a quite remarkable result.

VI. CONCLUSION

A new class of Fast Fourier Transform algorithms is investigated in the paper, multiple-split-radix FFTs. In the algorithms construction imbalance of the divide-and-conquer technique applied to DFT formula is exploited to the end, hence, the algorithms have smaller arithmetical complexities than their standard counterparts. The split-radix FFTs with one auxiliary base are already known, the paper shows how to extend the idea to several auxiliary bases, which is the case when DFT size is a product of powers of prime numbers. The presented technique results in savings in numbers of arithmetical operations even for smallest usable data vectors, hence, it can be implemented in construction of optimized FFTs of any size.

REFERENCES

[1] J.W. Cooley, J.W. Tukey, "An algorithm for machine computation of complex Fourier series," Math. Comput., vol. 19, pp. 297–301, 1965.

[2] —, "Special issue on fast Fourier transform," IEEE Trans. Audio Electroacoust., vol. AU-15, and vol. AU-17, June 1967, and June 1969.

[3] P. Duhamel, H. Hollmann, "Split-radix FFT algorithm", Electron. Lett., vol. 20, No. 1, pp. 14–16, 1984.

[4] S.G. Johnson, M. Frigo, "A modified split-radix FFT with fewer arithmetic operations", IEEE Trans. Signal Processing, vol. 55 (1), pp. 111–119, 2007.

[5] J.-B. Martens, "Recursive cyclotomic factorization - a new algorithm for calculating the Discrete Fourier Transform", IEEE Trans. Acoust., Speech, Signal Proces., vol. ASSP-32, pp. 750–761, 1984.

[6] Y. Suzuki, T. Sone, and K. Kido, "Anew FFT algorithm of radix 3, 6, and 12", IEEE Trans. Acoust., Speech, Signal Proces., vol. ASSP-34, pp. 380–383, 1986.

[7] R. Stasinski, "Radix-K FFT using K-point convolutions", IEEE Trans. Signal Proces., vol. 42, pp. 743–750, 1994.

[8] H.J. Nussbaumer, "Fast Fourier transform and convolution algorithms", Springer-Verlag, 1981.

[9] R. Stasinski, "Easy generation of small-N discrete Fourier transform algorithms", IEE Proc., Pt. G, Vol. 133, pp. 133–139, 1986.

[10] R. Stasinski, "The techniques of the generalized fast Fourier transform algorithm", IEEE Trans. Signal Proces., vol. 39, pp. 1058–1069, 1994.

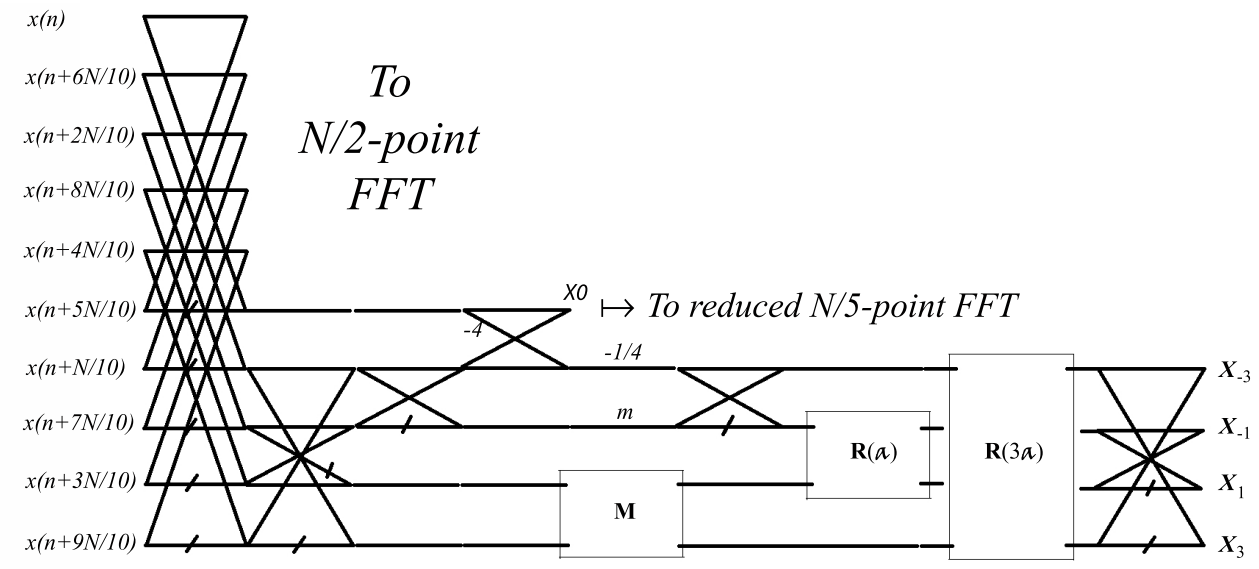

Fig. 3. Main butterfly of radix-10:2&5 T-FFT; $m = (\cos u - \cos 2u)/2$, $u = 2\pi/5$, $n = 0, 1, \ldots, N/10 - 1$; bar on branch means multiplication by -1.

TABLE I
SOME SPLIT-MULTIPLE-RADIX FFTS.

radix	b (17)	mults	adds	adds'	adds"	total	a_M	a_T
2	2	3	7	–	–	10	1.5000	5.0000
4	4	9	25	–	–	34	1.1250	4.2500
4:2 [3]		6	18	–	–	24	1.0000	4.0000
4F:2 [4]		6	50/3	–	–	68/3	**1.0000**	**3.7778**
6μ [6]	6	20	46	–	–	66	1.2895	4.2554
6μ:2&3 [5]	$(2^4 \cdot 3^3)^{1/6}$	8	28	16	–	32	0.9138	**3.6551**
6C:2&3 [7]		6	31	19	–	33	0.6853	3.7693
12:2&3	$(2^2 \cdot 3)^{1/2}$	16	72	48	–	80	0.7449	3.7192
12:2&3		20	68	44	–	80	0.9298	3.7192
24:2&3	$(2^8 \cdot 3^3)^{1/6}$	36	168	120	–	188	0.7056	3.6849
24:2&3		44	160	112	–	188	0.8624	3.6849
48:2&3	$(2^{10} \cdot 3^3)^{1/6}$	84	384	288	–	436	0.7116	3.6937
48:2&3		100	368	272	–	436	0.8472	3.6937
10:2&5	$(2^8 \cdot 5^5)^{1/10}$	20	68	48	–	84	1.0199	4.2836
10T:2&5		24	60	40	–	80	1.2239	4.0796
10T:2		30	66	–	–	96	1.3883	4.4425
10T	10	54	106	–	–	160	1.6256	4.8165
20:2&5	$(2^{12} \cdot 5^5)^{1/10}$	40	152	112	–	184	0.8471	3.8967
20:2&5		44	144	104	–	180	0.9318	3.8120
40:2&5	$(2^{16} \cdot 5^5)^{1/10}$	84	360	280	–	428	0.7606	3.8755
40:2&5		92	344	264	–	420	0.8330	3.8030
80:2&5	$(2^4 \cdot 5)^{1/2}$	180	824	664	–	972	0.7118	3.8438
80:2&5		188	792	632	–	948	0.7434	3.7489
14:2&7	$(2^{12} \cdot 7^7)^{1/14}$	32	122	94	–	150	1.0110	4.7391
14T:2&7		38	108	80	–	142	1.2006	4.4864
28:2&7	$(2^{18} \cdot 7^7)^{1/14}$	64	268	212	–	324	0.8499	4.3026
28:2&7		68	256	200	–	316	0.9030	4.1964
30:2&3&5	$(2^8 \cdot 3^6 \cdot 5^5)^{1/15}$	42	236	176	136	242	0.7212	4.1553
30:2&3&5		46	228	168	128	238	0.7899	4.0866
60:2&3&5	$(2^{12} \cdot 3^6 \cdot 5^5)^{1/15}$	84	504	384	304	516	0.6341	3.8950
60:2&3&5		92	488	368	288	508	0.6945	3.8346
84:2&3&7	$(2^{18} \cdot 3^9 \cdot 7^7)^{1/21}$	132	816	648	592	868	0.6356	4.1798
84:2&3&7		140	792	624	568	852	0.6742	4.1028