Original Article

www.jmss.mui.ac.ir

Calculation of Computational Complexity for *Radix-2^p* Fast Fourier Transform Algorithms for Medical Signals

Rassoul Amirfattahi

Digital Signal processing Research Lab., Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, 84156-83111, Iran

Submission: 25-08-2013 Accepted: 03-09-2013

ABSTRACT

Owing to its simplicity *radix-2* is a popular algorithm to implement fast fourier transform. *Radix-2^p* algorithms have the same order of computational complexity as higher radices algorithms, but still retain the simplicity of *radix-2*. By defining a new concept, twiddle factor template, in this paper, we propose a method for exact calculation of multiplicative complexity for *radix-2^p* algorithms. The methodology is described for *radix-2*, *radix-2^2* and *radix-2^3* algorithms. Results show that *radix-2^2* and *radix-2^3* have significantly less computational complexity compared with *radix-2*. Another interesting result is that while the number of complex multiplications in *radix-2^3* algorithm is slightly more than *radix-2^2*, the number of real multiplications for *radix-2^3* is less than *radix-2^2*. This is because of the twiddle factors in the form of $W^{\frac{(2k+1)N}{8}}$ which need less number of real multiplications and are more frequent in *radix-2^3* algorithm.

Key words: *Computational complexity, fast fourier transform, medical signals, radix-2^p*

INTRODUCTION

Fast fourier transforms (FFT) is a key tool in most of digital signal processing systems such as medical systems. FFT is an effective method for calculation of discrete fourier transform (DFT).

Radix-2 method proposed by Cooley and Tukey[1] is a classical algorithm for FFT calculation. Due to high computational complexity of FFT, higher radices algorithms such as *radix-4* and *radix-8* have been proposed to reduce computational complexity. On the other side, for real-time applications, such as medical applications, hardware implementation of FFT is interested. Another application is in digital communication systems based on Orthogonal Frequency Division Multiplexing, where FFT/IFFT block processes input data in their physical layer. Simplicity of the algorithm is very important to have efficient hardware architectures.

Although *radix-2* algorithms have the same order of computational complexity as *radix-4* and *radix-8* algorithms, their flow graphs are as simple as *radix-2* algorithm. In *radix-2^p* algorithms "*p*" is a natural number. These algorithms were introduced with *radix-2^2* in 1996[2] and are developing for higher radices.[3-9] Using *radix-2^p* to calculate FFT for real signals like medical signals is very efficient.[10]

Although it is clear that their complexity is less than *radix-2* algorithm, any systematic method to calculate computational complexity of *radix-2^p* algorithms has not been proposed yet. This paper proposes a methodology to compute the number of complex and real multiplications, exactly.

The rest of this article is organized as follows. First the importance of FFT algorithm in medical applications is described. Then, *radix-2* and *radix-2^p* algorithms are explained. In the following section we define twiddle factor template (TFT). The proposed methodology for calculation of computational complexity by using TFT is described in the next section. After that we compare the results computed for *radix-2^2* and *radix-2^3* algorithms. Finally, we conclude the article.

FFT FOR MEDICAL SIGNALS

Digital signal processing is a key tool in medical applications. In order to extract some features of a medical signal, not visible in time domain, we need to transform signal representation into the frequency domain. For example, FFT is used to extract abnormalities of electrocardiogram signals for distinguishing heart diseases.[11] Or it is used to process electroencephalogram signal for seizure prediction.[12] The FFT plays an important role in different

Address for correspondence:
Dr. Rassoul Amirfattahi, Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, 84156-83111, Iran.
E-mail: fattahi@cc.iut.ac.ir

real-time applications. In some medical applications,[11-13] the power spectral density of various real-valued signals has to be estimated. This requires calculation of the FFT repetitively on many overlapping windows of the signals.

Since, FFT algorithms are very common in transferring a time-domain signal into the frequency domain, their efficient implementation for medical signals is very important. Having a methodology for calculation of computational complexity of FFT algorithms is helpful for their evaluation.

Radix-2 FFT Algorithm

The *N*-point DFT for a sequence $x(n)$ is defined as:[14]

$$X[k]=\sum_{n=0}^{N-1}x(n)\ \ W_N^{nk}\quad k=0,1,\ldots,\ N-1 \tag{1}$$

where $W_N=e^{-j\frac{2\pi}{N}}$. *Radix-2* FFT algorithm reduces the order of computational complexity of Eq. 1 by decimating even and odd indices of input samples. There are two kinds of decimation:[14] decimation in the time domain and decimation in frequency (DIF) domain. Figure 1 shows the flow graph for *radix-2* DIF FFT for $N = 16$.

The flow graph in Figure 1 composed of Butterfly (BF) components with 2 inputs and 2 outputs. The structure of a BF is shown in Figure 2.

The bottom output of each BF is multiplied by W_N^k. W_N^k is called twiddle factor and k is its rotation angle.

Radix-2p Algorithms

This section describes *radix*-2^2 and *radix*-2^3 approaches and their advantages in details and then generalization of the idea to *radix*-2^p algorithms is described.

Radix-2^2: This approach proposed by He and Torkelson.[2] Main idea is to implement two stages of DIF simultaneously. This provides simplification of twiddle factors in these two stages. Time and frequency indices are decomposed as follows:

$$n=\left\langle\frac{N}{2}n_1+\frac{N}{4}n_2+n_3\right\rangle_N \tag{2}$$

$$k=\left\langle k_1+2k_2+4k_3\right\rangle_N \tag{3}$$

where $<\ >_N$ represents modulo N. k_i and n_i in these equation are integer numbers used to decompose k and n, respectively. Substituting (2) and (3) in (1) results:

$$X[k_1+2k_2+4k_3]=\sum_{n_3=0}^{\frac{N}{4}-1}\sum_{n_2=0}^{1}\sum_{n_1=0}^{1}x\left(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3\right)W_N^{\left(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3\right)k}$$

$$=\sum_{n_3=0}^{\frac{N}{4}-1}\sum_{n_2=0}^{1}\left[x\left(\frac{N}{4}n_2+n_3\right)+(-1)^{k_1}x\left(\frac{N}{2}+\frac{N}{4}n_2+n_3\right)\right]W_N^{\left(\frac{N}{4}n_2+n_3\right)k}$$

$$BF\left(\frac{N}{4}n_2+n_3\right)$$

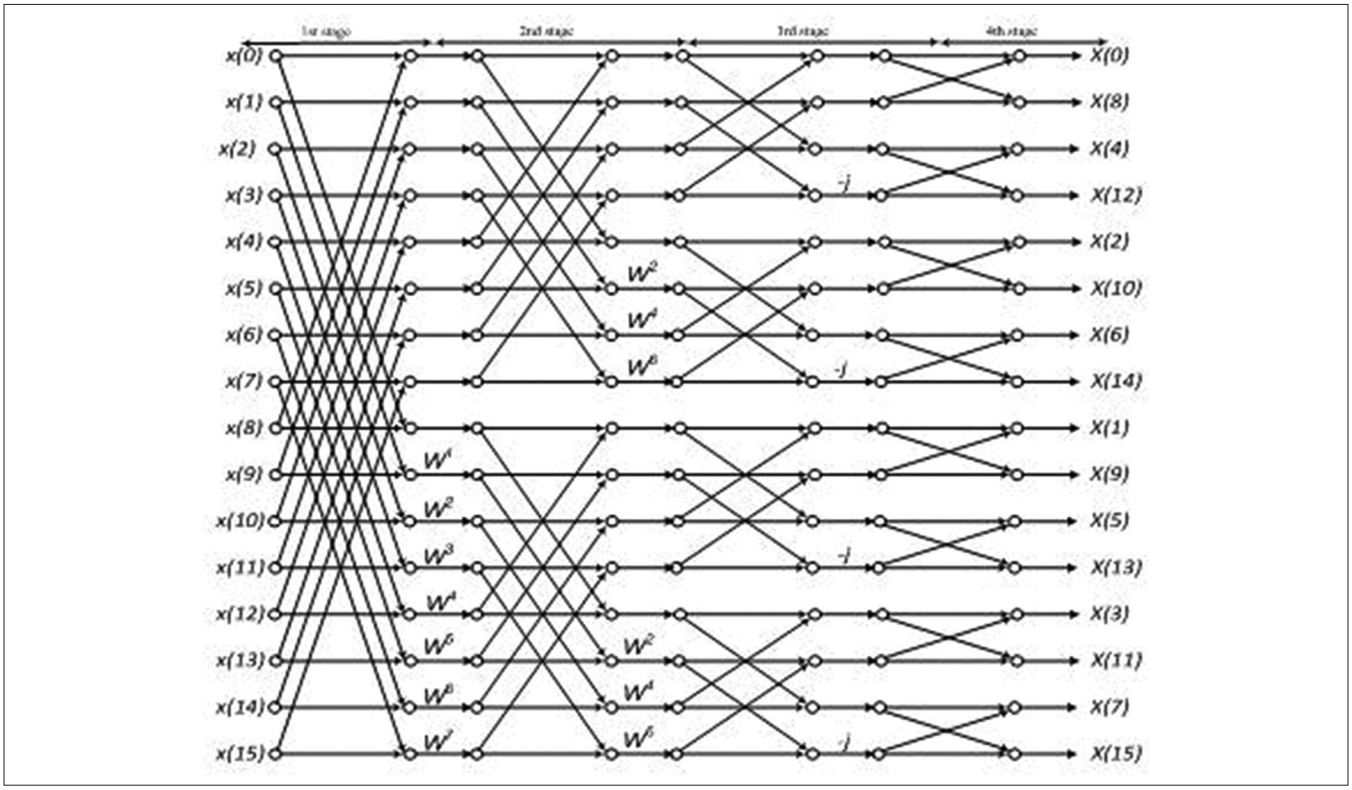

Figure 1: *Radix-2* decimation in frequency fast fourier transforms for $N = 16$

$$= \sum_{n_3=0}^{\frac{N}{4}-1}\left[\underbrace{\left(BF\left(n_3\right)+(-j)^{k_1+2k_2}\,BF\left(\frac{N}{4}+n_3\right)\right)}_{H(k_1,k_2,n_3)}W_N^{n_3(k_1+2k_2)}\right]W_N^{4n_3k_3}$$

$$= \sum_{n_3=0}^{\frac{N}{4}-1}H(k_1,k_2,n_3)W_{\frac{N}{4}}^{n_3k_3} \tag{4}$$

As (4) implies N-point FFT of $X[k]$ is converted to $\frac{N}{4}$-point FFT of $H(k_1, k_2, n_3)$ By changing k_1 and k_2 four different values of H are chosen. In other words, that an N-point FFT can be computed by implementing two stages of decimation together and then computing four $\frac{N}{4}$-point FFTs. Figure 3 shows the structure achieved by (4) for $N = 16$.

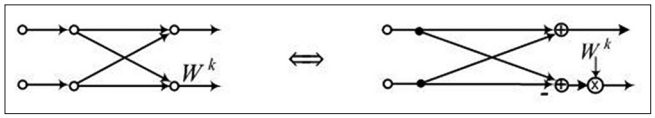

Figure 2: Input structure of a butterfly component in decimation in frequency fast fourier transforms flow graph

The decomposition can be repeated for $\frac{N}{4}$-point FFT blocks. Figure 4 shows the final *radix-2²* flow graph for $N = 16$.

Figure 4 has less number of nontrivial twiddle factors compared with Figure 1. Furthermore, for every two stages in Figure 4 one of them has trivial twiddle factors, $-j$.

Radix-2³: The *radix-2³* FFT algorithm is proposed in.[3] Similar to *radix-2²*, we can derive the *radix-2³* algorithm by using the following new indices:[3]

$$n = \left\langle \frac{N}{2}n_1 + \frac{N}{4}n_2 + \frac{N}{8}n_3 + n_4 \right\rangle_N \tag{5}$$

$$k = \left\langle k_1 + 2k_2 + 4k_3 + 8k_4 \right\rangle_N \tag{6}$$

Figure 5 shows the flow graph of *radix-2³* algorithm for $N = 64$. In Figure 5 for each three stages the first one has twiddle factors equal to $-j$ and the second one contains twiddle factors in the form of $W^{\frac{N}{8}}$ or $-jW^{\frac{N}{8}}$.

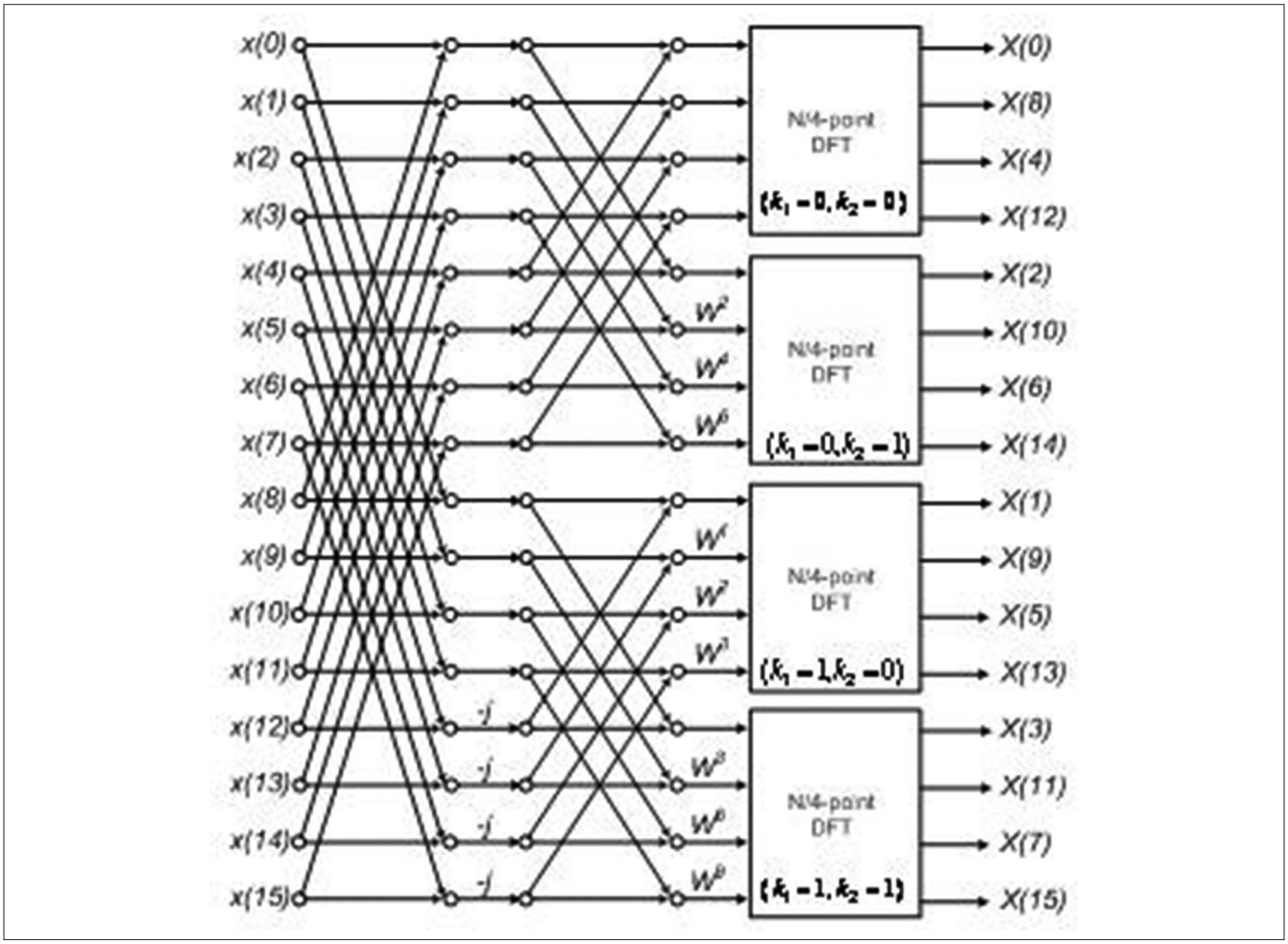

Figure 3: First decomposition of *radix-* 2^2 algorithm for $N = 16$

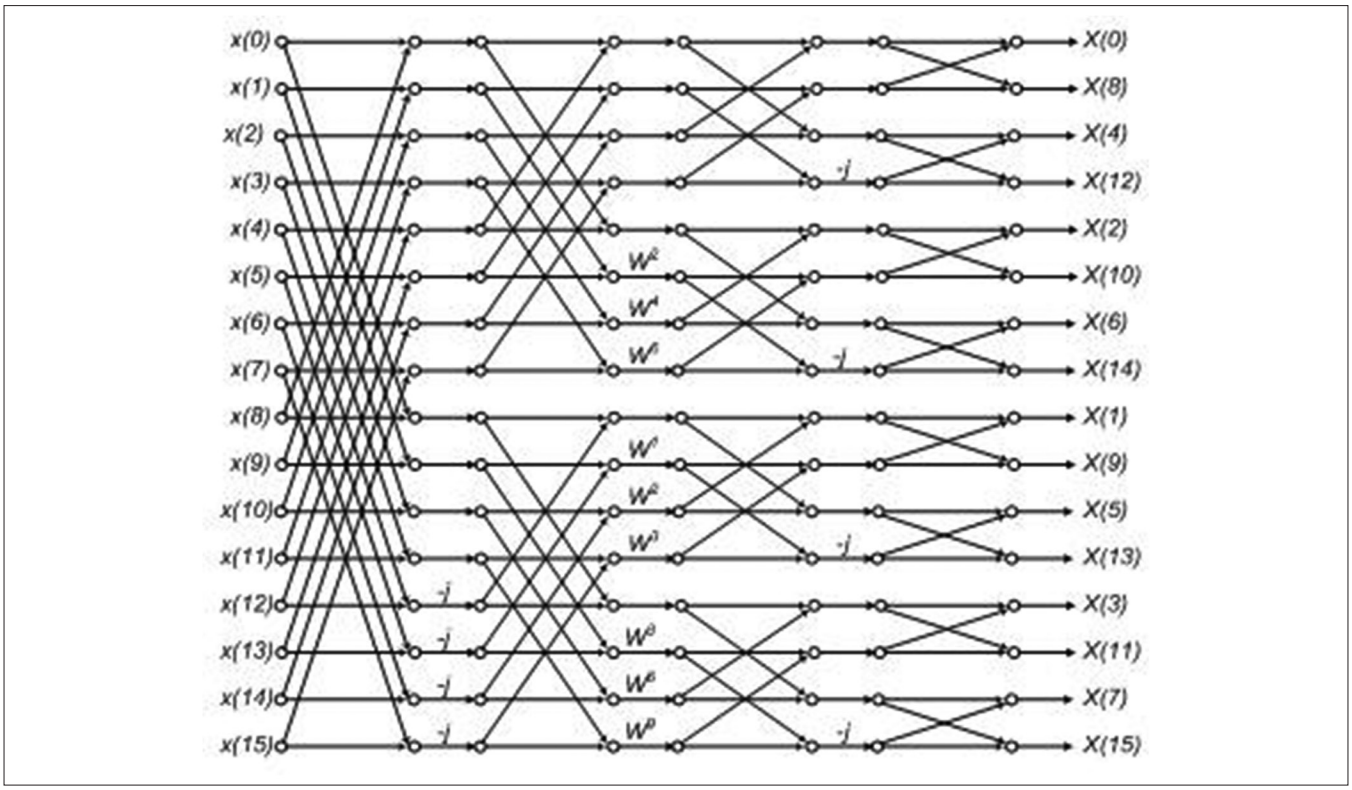

Figure 4: 16-point fast fourier transforms flow graph based on *radix-2*^{2}

In general for *radix-2*^{p} algorithms time and frequency indices are decomposed in the following forms:

$$n = \left\langle \frac{N}{2} n_1 + \frac{N}{4} n_2 + \cdots + \frac{N}{2^p} n_p + n_{p+1} \right\rangle_N \quad (7)$$

$$k = \left\langle k_1 + 2k_2 + \cdots + 2^p k_{p+1} \right\rangle_N$$

PROPOSED METHODOLOGY

This section describes the proposed method to compute multiplicative complexity of *radix-2*^{p} algorithms.

TFT

For different FFT algorithms, different twiddle factors appear at different positions in the flow graph. By comparing Figures 1 and 4, 5, it can be seen that for each algorithm there is a regular arrangement for twiddle factors. TFT related to each algorithm, graphically represents the place and rotation value for each twiddle factor in the flow graph of the algorithm. For example in *radix-2* flow graph, shown in Figure 1, twiddle factors appear in the bottom outputs of BFs in each stage. Furthermore, the rotation angles of twiddle factors in each stage are twice compared with the rotation angles of twiddle factors in their previous stage. It means that twiddle factors in the first stage are W^1, W^2, W^3,…, while for the second stage they are W^2, W^4, W^6,…. So Figure 6 shows TFT for *radix-2* algorithm. The numbers in Figure 6 represent rotation angles of twiddle factors. In other words instead of W^1, W^2, W^3…, numbers 1, 2, 3,… are displayed in TFT. Twiddle factors in the form of W^0 are not shown in TFT.

In the same manner, we can derive the TFT for *radix-2*^{2} and *radix-2*^{3} as shown in Figures 7 and 8, respectively. In TFT for *radix-2*^{p} algorithms each successive *P* stages is called a section. Sections have a similar pattern of twiddle factors as shown in Figures 7 and 8, respectively for *radix-2*^{2} and *radix-2*^{3} algorithms.

CALCULATION OF MULTIPLICATIVE COMPLEXITY

Complex Multiplicative Complexity

We determine the TFT of each algorithm according to its mathematical equations. In the next step, the number of complex twiddle factors in the first section of the TFT is determined. It is noticeable that $W_N^0 = 1$, $W_N^{\frac{N}{2}} = -1$, $W_N^{\frac{N}{4}} = -j$ and $W_N^{\frac{3N}{4}} = j$ are not counted here. Then, the number and length of blocks in the second section similar to the first section are determined. Now, we can calculate the total number of complex twiddle factors for each algorithm by:

- Defining a recursive equation and solving it or
- Defining a series and calculating its summation.

These methods are described for *radix-2*^{2} and *radix-2*^{3} algorithms.

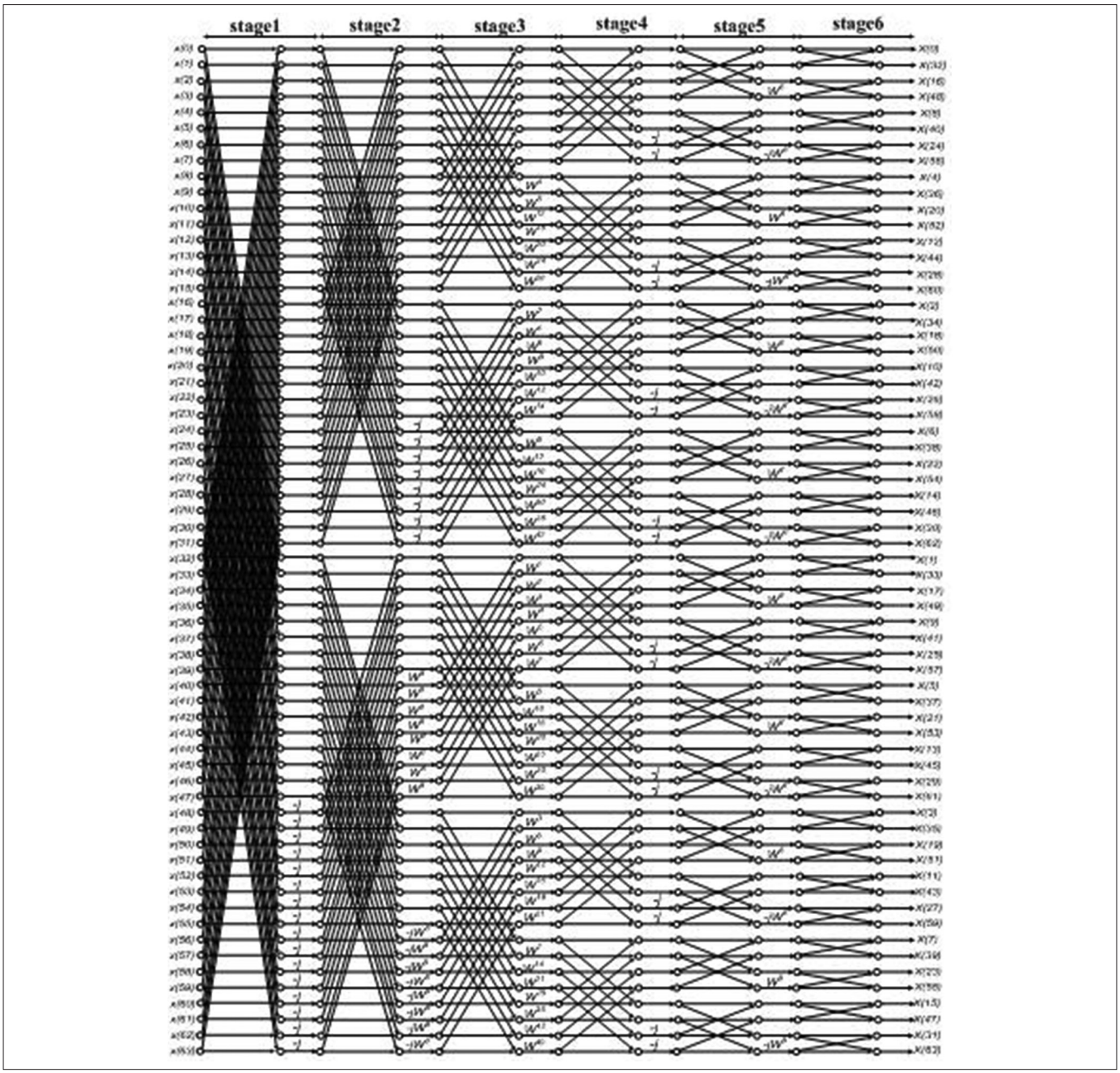

Figure 5: Flow graph of *radix-2*^{3} algorithm for $N = 64$

Radix-2^2*:* As Figure 7 shows twiddle factors in the first stage of each section in *radix-2*^{2} algorithm are $W_N^{\frac{N}{4}} = -j$. In stage 2 of Figure 7 three blocks with length $\frac{N}{4}$ have complex twiddle factors. The first twiddle factor in each block is $W^0 = 1$ and there is a twiddle factor equal to $W^{\frac{N}{4}} = -j$ in the first block. Hence, the number of nontrivial twiddle factors in the first section is $\frac{3}{4}N - 4$. Second section contains four blocks similar to the first section with length $\frac{N}{4}$. The recursive equation for the total number of complex multiplications can be written as:

$$M_c(N) = \left(\frac{3}{4}N - 4\right) + 4 \times M_c\left(\frac{N}{4}\right) \quad N \geq 16,\ M_c(4) = 0 \tag{8}$$

In the other way, we can derive the $M_c(N)$ as the summation of the following series:

$$M_c(n) = \underbrace{\frac{3}{4}N - 4}_{1\,(.4} + \underbrace{4 \times \left[\frac{3}{4}\left(\frac{N}{4}\right) - 4\right]}_{2\,(.4} + \underbrace{16 \times \left[\frac{3}{4}\left(\frac{N}{16}\right) - 4\right]}_{3\,(.4} + \ldots + \underbrace{\left[\frac{3}{4}N - \frac{N}{4}\right]}_{(\log_4 N - 1)\,(.4} = \frac{3}{4}N\left(\log_4 N - 1\right) - \sum_{i=1}^{\log_4 N - 1} 4^i \tag{9}$$

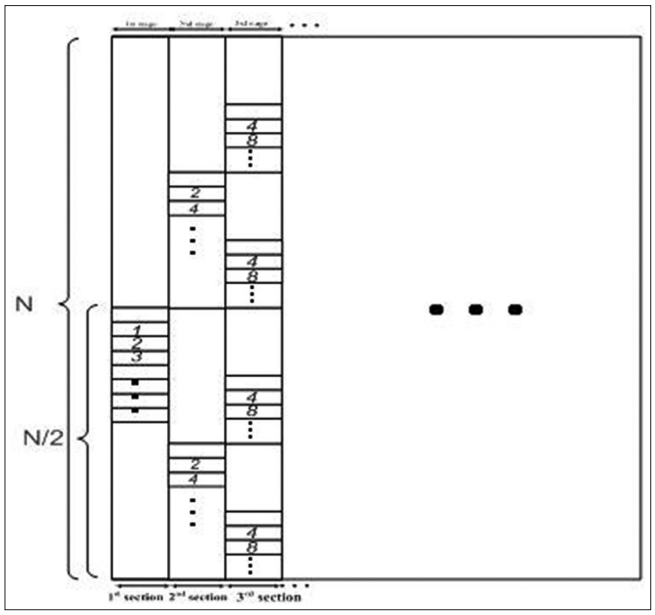

Figure 6: Twiddle factor template for *radix-2* fast fourier transforms algorithm

Both the above approaches achieve 10 for the total number of nontrivial twiddle factors in *radix-2²* algorithm.

$$M_c(N) = \frac{3}{4}N\log_4 N - \frac{13}{12}N - \frac{4}{3} \tag{10}$$

Radix-2³: According to the TFT in Figure 8 recursive equation and series for the number of complex multiplications in *radix-2³* algorithm are derived as 12 and 13, respectively.

$$M_c(N) = \frac{9}{8}\times N - 8 + 8\times M_c\left(\frac{N}{8}\right) N \geq 64,\ \ M_c(8) = 0 \tag{11}$$

$$\frac{N}{4} + \frac{7N}{8} - 8 + 8\times\left[\frac{1}{4}\left(\frac{N}{8}\right) + \frac{7}{8}\left(\frac{N}{8}\right) - 8\right] + \ldots + \frac{N}{8}\times 2 = \frac{9}{8}N\log_8 N + \frac{N}{8} - \sum_{i=1}^{\log_8 N} 8^i \tag{12}$$

The total number of complex multiplications is calculated as 14.

$$M_c(N) = \frac{9}{8}N\log_8 N - \frac{57}{56}N + \frac{8}{7} \tag{13}$$

Real multiplicative complexity

Although the number of complex multiplications provides a good estimation for multiplicative complexity of an FFT algorithm, it isn't the actual number of multiplications. Based on (15) multiplication of two complex numbers, $x + jy$ and $a + jb$, can be implemented by three real multiplications (multiplication of two real numbers).

$$(x+y)a - y(a+b) + j[(x+y)a + x(b-a)] \tag{14}$$

On the other side, twiddle factors in the form of $W^{\frac{(2k+1)N}{8}}$ for $k = 0, 1, 2, 3$, have same real and imaginary parts and need only two real multiplications. By computing the number of these special twiddle factors in the TFT we can obtain the exact number of real multiplications for each algorithm.

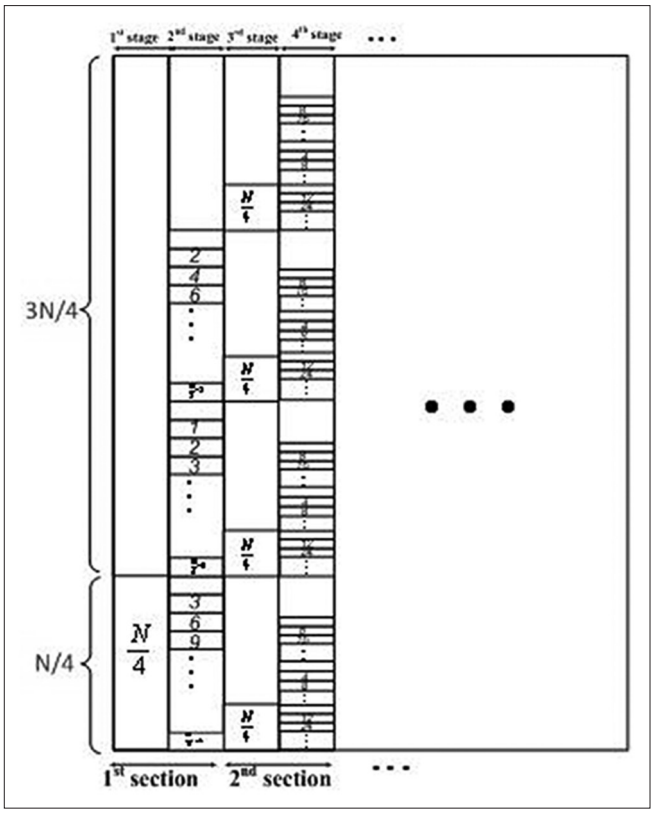

Figure 7: Twiddle factor template for *radix-2²*

Radix-2²: Using the TFT in Figure 7, we compute the number of special twiddle factors for the first section of the TFT. There are 4 twiddle factors in the first section, which need two real multiplications. The number of real multiplications in the first section is computed as 16.

Number of real multiplications in the 1st stage

$$= 3\times\left(\frac{3}{4}N - 4\right) - 4 = \frac{9}{4}N - 16 \tag{15}$$

Like complex multiplications, using the recursive equation in 17 or series in 18, we can compute the total number of real multiplications as 19.

$$M_r(N) = \frac{9}{4}N - 16 + 4\times M_r\left(\frac{N}{4}\right)\ \ N \geq 16 \tag{16}$$

$$M_r(N) = \frac{9}{4}N - 16 + 4\times\left(\frac{9}{4}\times\left(\frac{N}{4}\right) - 16\right) + \cdots + \left(\frac{9}{4}N - N\right)\ \ N \geq 16 \tag{17}$$

$$M_r(N) = \frac{9}{4}N\times\log_4 N - \frac{43}{12}N + \frac{16}{3}\ \ N \geq 16 \tag{18}$$

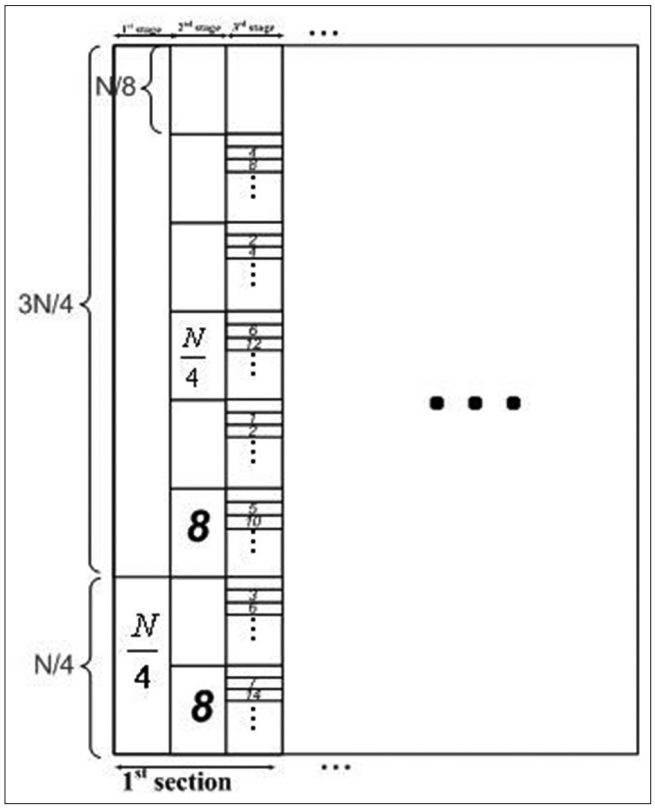

Figure 8: Twiddle factor template for *radix-2³*

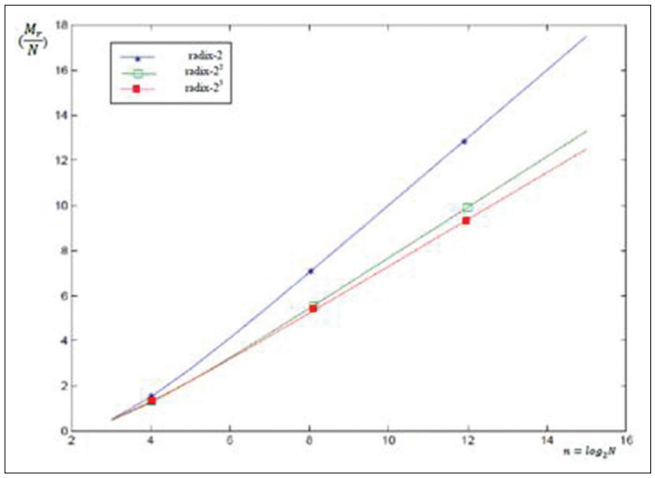

Figure 10: Real multiplicative complexity

Radix-2³: The total number of complex twiddle factors in the first section of TFT in Figure 8 is $\left(\frac{N}{4}+\frac{7N}{8}-8\right)$. Although $\frac{N}{4}+4$ of them are special twiddle factors. Hence, the number of real multiplications in the first section is:

$$\text{Number of real multiplications in the 1st stage} = 3\times\left(\frac{N}{4}+\frac{7N}{8}-8\right)-\left(\frac{N}{4}+4\right)=\frac{25N}{8}-28 \quad (19)$$

Then recursive equation for *radix-2³* algorithm is:

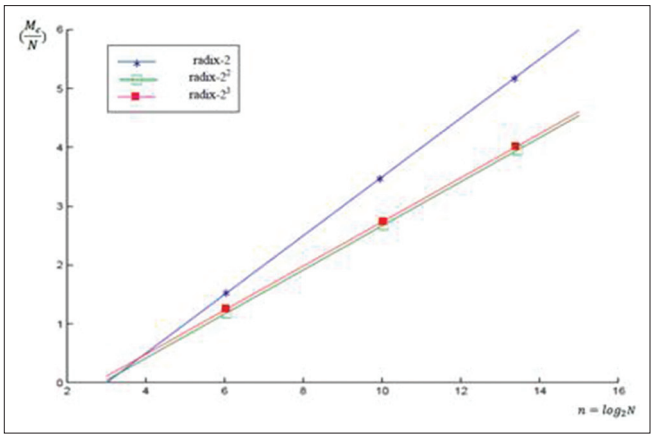

Figure 9: Complex multiplicative complexity

$$M_r(N)=8\times M_r\left(\frac{N}{8}\right)+\left(\frac{25N}{8}-28\right)\quad N\geq 64 \quad (20)$$

By solving 21 the total number of real multiplications in *radix-2³* is computed as 22.

$$M_r(N)=\frac{25}{8}N\times log_8 N-\frac{25}{8}N+4\quad N\geq 64 \quad (21)$$

Comparison

We compare the computational complexity of *radix-2, radix-2²* and *radix-2³* algorithms using the equations obtained in the previous sections. Figures 9 and 10 show the number of complex and real multiplications respectively. As these figures show the computational complexities for *radix-2²* and *radix-2³* algorithms are significantly less than *radix-2* algorithm.

Compared to *radix-2²*, for *radix-2³* algorithm complex multiplications are slightly more. However the number of real multiplications in *radix-2³* is less than the number of real multiplications in *radix-2²* algorithm. As *N* increases the difference between the number of real multiplications for *radix-2²* and *radix-2³* is more remarkable.

CONCLUSION

In this paper, we proposed a methodology to compute complex and real multiplicative complexities for *radix-2ᵖ* algorithms. The method uses TFT in order to exploit regularity in these algorithms. As two special cases calculation of the computational complexity for *radix-2²* and *radix-2³* algorithms using the proposed method was described. Finally, *radix-2*, *radix-2²* and *radix-2³* were compared regarding to their complex and real multiplicative complexities. The method can easily be extended for *radix-2ᵖ* algorithms.

REFERENCES

1. Cooley JW, Tukey JW. An algorithm for the machine calculation of complex fourier series. Math Comput 1965;19:297-301.

2. He S, Torkelson M. A new approach to pipeline FFT processor. In Proceedings of 10th International Conference on Parallel Processing Symposium. 1996;766-70.
3. He S, Torkelson M. Design and implementation of a 1024-point pipeline FFT processor. In Proceedings of the IEEE Custom Integrated Circuits Conference. 1998;131-4.
4. Lee J, Lee H, Cho SI, Choi SS. A high-speed two parallel radix-2^4 FFT/IFFT processor for MB-OFDM UWB systems. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2008;E91-A:1206-11.
5. Shin M, Lee H. A high-speed four parallel radix-2^4 FFT/IFFT processor for UWB applications. In Proceedings of the IEEE International Symposium on Circuits and Systems. 2008;960-3.
6. Fan CP, Lee MS, Su GA. A low multiplier and multiplication costs 256-point FFT implementation with simplified radix-24 SDF architecture. In Proceedings of IEEE Asia Pacific Conference on Circuits and Systems. 2006;1935-8.
7. Cho T, Lee H, Park J, Park C. A high-speed low-complexity modified radix-25 FFT processor for gigabit WPAN applications. In IEEE International Symposium on Circuits and Systems. 2011;1259-62.
8. Garrido M, Grajal J, Sánchez MA, Gustafsson O. Pipelined radix-2*k* feedforward FFT architectures. IEEE Trans VLSI Syst 2013;21:23-32.
9. Cortés A, Vélez I, Sevillano JF. Radix *rk* FFTs: Matricial representation and SDC/SDF pipeline implementation. IEEE Trans Signal Process 2009;57:2824-39.
10. Salehi SA, Amirfattahi R, Parhi KK. Pipelined architectures for real-valued FFT and Hermitian-symmetric IFFT with real data paths. IEEE Transactions on Circuits and Systems II, In press.
11. Cheng MH, Chen LC, Hung YC, Yang CM. A real-time maximum likelihood heart-rate estimator for wearable textile sensors. In Proceedings of 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 2008;254-7.
12. Park Y, Luo L, Parhi KK, Netoff T. Seizure prediction with spectral power of EEG using cost-sensitive support vector machines. Epilepsia 2011;52:1761-70.
13. Yazicioglu RF, Merken P, Puers R, Van Hoof C. Low-power low-noise 8-channel EEG front-end ASIC for ambulatory acquisition systems. In Proceedings of the 32nd European Solid-State Circuits Conference. 2006;247-50.
14. Oppenheim AV, Schafer RW, Buck JR. Discrete-Time Signal Processing. 2nd ed. Englewood Cliffs, NJ: Prentice Hall; 1998.

How to cite this article: Amirfattahi R. Calculation of Computational Complexity for *Radix*-2^p Fast Fourier Transform Algorithms for Medical Signals. J Med Sign Sens 2012;3:195-217-24.
Source of Support: Nil, **Conflict of Interest:** None declared

BIOGRAPHIES

Rassoul Amirfattahi was born in 1969. He received BS degree in Electrical Engineering from Isfahan University of technology, Isfahan, Iran in 1993, MS degree in Biomedical Engineering and Ph.D degree in Electrical Engineering both from Amirkabir University of technology (The Tehran Polytechnic), Tehran, Iran in 1996 and 2002 respectively. From 2003 he joined Isfahan University of Technology while he is currently an Associate Professor and director of digital signal processing research laboratory at department of Electrical and Computer Engineering. His research interests include Biomedical signal and image processing, speech and audio analysis, Biological system modeling and DSP algorithms. He is an author or coauthor of more than 150 technical papers, one book and two book chapters.

E-mail: fattahi@cc.iut.ac.ir