A New Relation between "Twiddle Factors" in the Fast Fourier Transformation

Jozsef Suto[1,2], Stefan Oniga[1]
[1]*Department of Informatics Systems and Networks, University of Debrecen, Egyetem ter 1, 4032 Debrecen, Hungary*
[2]*Institute for Nuclear Research, Hungarian Academy of Science, Bem ter 18/c, H-4026 Debrecen, Hungary*
suto.jozsef@inf.unideb.hu

***Abstract*—The fast Fourier transformation algorithm (FFT) probably is the most important algorithm in the digital signal processing. It is an efficient algorithm to the discrete Fourier transformation which determines the frequency components of a discrete time-varying signal. Nowadays, it has a huge impact on the modern society because the FFT is running on more billion devices (e.g. smartphones) on the planet all the time and this tendency is continuously increasing. Moreover, this algorithm plays a key role in the computer science and engineering. Consequently, a well optimized algorithm can save tremendous resources (calculation capacity and memory).**

This paper presents a new relation between "twiddle factors" and gives an optimised form to the existing relations. In the paper the experimental results prove the efficiency of the proposed relations. By the new relations every radix-r and split-radix FFT will be more efficient because it accelerates the algorithm and/or saves memory.

***Index Terms*—Digital signal processing, Fourier transforms, optimization, signal processing algorithms.**

I. INTRODUCTION

The discrete Fourier transform (DFT) is among the most fundamental method in the digital signal processing (DSP). However, its wide use was restricted by its computational needs. In 1965 Cooley and Tukey presented an efficient algorithm to the DFT which reduces the number of operations from N^2 to $N\log_2(n)$ where N is power of two (2^n) [1]. It was an important milestone in the DSP research. Thereafter, many articles presented refinements to the original algorithm such us decimation in frequency (DIF), higher radices, specialised fast Fourier transform (FFT) to real data, etc. A clear and detailed review about the FFT evolution can be found in [2].

Nowadays, more methods exist for computing the DFT efficiently. Generally, the radix-4 and split-radix algorithms are used when the sample size is power of two whereas the prime factor algorithm is popular for size having co-prime factors [3]–[6].

In many cases, the very-large-scale integration (VLSI) design largely determines the usefulness of a given FFT algorithm. It means that the efficiency of an FFT algorithm depends on the applied processor architecture [7], [8]. In addition, the most FFT algorithms are well parallelisable, therefore, in multiprocessor systems the parallelisation accelerates the transformation [9]–[11]. However, in single processor systems the parallelisation does not cause significantly acceleration. In this case, a sequential algorithm is more advantageous.

Independently of the applied implementation technique and architecture the relations which will be discussed in section III are similarly important.

Manuscript received January 5, 2015; accepted May 24, 2015.

II. THEORETICAL BACKGROUND

The DFT is a main part of the Fourier analysis and it is a very important part of DSP because various DSP applications such as filtering and correlation analysis depend on it. Generally, the DFT of an x(α) time-varying signal with N samples

$$X(\kappa) = \sum_{\alpha=0}^{N-1} x(\alpha) T_N^{\alpha\kappa}, \tag{1}$$

where © $= 0,\ldots, N-1$.

$$T_N^{\alpha\kappa} = e^{-i\frac{2\pi}{N}\alpha\kappa}, \tag{2}$$

where $X(\kappa)$ denotes the frequency coefficients, $T_N^{\alpha\kappa}$ is the "twiddle factor", α is index in the time domain, κ is index in the frequency domain and i is the imaginary unit. Every fast algorithm for DFT is based on the "divide and conquer" idea. The radix-2 decimation in time (DIT) FFT decomposes the input signal into its even and odd components, thus the DFT can be written as

$$X(\kappa) = \sum_{\alpha=0}^{N/2-1} x(2\alpha) T_N^{2\alpha\kappa} + \sum_{\alpha=0}^{N/2-1} x(2\alpha+1) T_N^{(2\alpha+1)\kappa} = \\ = \sum_{\alpha=0}^{N/2-1} x(2\alpha) T_{N/2}^{\alpha\kappa} + T_N^{\kappa} \sum_{\alpha=0}^{N/2-1} x(2\alpha+1) T_{N/2}^{\alpha\kappa}. \tag{3}$$

In the above equation T_N^{κ} in the κ'th root of unit ("twiddle factor" where α = 1). Equation (3) can be written in a simpler form (4) if we use the symmetry (5) between roots:

$$X(\kappa) = DFT_{even}(\alpha) + T_N^{\kappa} DFT_{odd}(\alpha), \tag{4}$$

$$X(\kappa) = DFT_{even}(\alpha) - T_N^{\kappa} DFT_{odd}(\alpha), \quad (5)$$

$$T_N^{\kappa} = T_N^{\kappa+N/2}. \quad (6)$$

In (4) κ = 0, …, N/2-1, in (5) κ = N/2,…, N-1 and α = 0, …, N/2-1 in both formulas. When the number of samples is a power of four ($N = 4^n$), the radix-4 FFT algorithm (7) is more efficient than radix-2. The radix-4 and split-radix (mixture of radix-2 and radix-4) FFT algorithms take advantage of another symmetry of roots (8) to reduce the multiplications by ±i [12]. The radix-4 algorithm splits the input sequence into four subparts thereby it decreases the number of stages by one [13], [14]:

$$X(\kappa) = \sum_{\alpha=0}^{N/4-1} \left(\sum_{\rho=0}^{3} x(\alpha + \rho\frac{N}{4}) T_N^{N/4\rho\kappa} \right) T_N^{\alpha\kappa} = \\ = \sum_{\alpha=0}^{N/4-1} \left(\sum_{\rho=0}^{3} x(\alpha + \rho\frac{N}{4}) (-i)^{\rho\kappa} \right) T_N^{\alpha\kappa}, \quad (7)$$

$$T_N^{\kappa+N/4} = -iT_N^{\kappa}. \quad (8)$$

In the above equation ρ denotes the subpart index.

III. THE PROPOSED RELATIONS

Equation (7) can be written in the following form where Re and Im refers to the real a complex parts:

$$T_{N_{Im}}^{\kappa+N/4} = -T_{N_{Re}}^{\kappa}, \quad (9)$$

$$T_{N_{Re}}^{\kappa+N/4} = T_{N_{Im}}^{\kappa}. \quad (10)$$

The above new forms are more efficient than (8), because these do not require complex multiplication.

Beyond (6) and (8), another important relation exists between the roots. According to the previous statements, we should focus on the first N/4 roots. For instance, if we assume N = 16 and we use the Euler form (11), the first N/4 roots will be (12)–(15). The Euler form separates the real and imaginary parts of a complex number and it is very useful in programming because the programmer can contain the two parts in two different vectors in the program code:

$$e^{i\theta} = \cos(\theta) - i\sin(\theta), \quad (11)$$

$$e^{-i2\pi 0/16} = \cos(0) - i\sin(0), \quad (12)$$

$$e^{-i2\pi 1/16} = \cos(\frac{\pi}{8}) - i\sin(\frac{\pi}{8}), \quad (13)$$

$$e^{-i2\pi 2/16} = \cos(\frac{\pi}{4}) - i\sin(\frac{\pi}{4}), \quad (14)$$

$$e^{-i2\pi 3/16} = \cos(\frac{3\pi}{8}) - i\sin(\frac{3\pi}{8}). \quad (15)$$

In the above equations there is a further relation between (13) and (15) since:

$$\cos(\frac{\pi}{8}) = \sin(\frac{3\pi}{8}), \quad (16)$$

$$\cos(\frac{3\pi}{8}) = \sin(\frac{\pi}{8}). \quad (17)$$

This comes from the property of the sine and cosine functions because there is a π/2 shift between them [15]:

$$\cos(\theta) = \sin(\frac{\pi}{2} - \theta), \quad (18)$$

$$\sin(\theta) = \cos(\frac{\pi}{2} - \theta). \quad (19)$$

The above relations can be observed between the roots when N is power of two. Generally, it can be written as:

$$\cos(\frac{\pi}{2^{n-1}}) = \sin(\frac{(2^{n-2}-1)\pi}{2^{n-1}}), \quad (20)$$

$$\cos(\frac{(2^{n-2}-1)\pi}{2^{n-1}}) = \sin(\frac{\pi}{2^{n-1}}). \quad (21)$$

Equation (22) is the proof of (20) and (23) is the proof of (21):

$$\sin(\frac{(2^{n-2}-1)\pi}{2^{n-1}}) = \sin(\frac{2^{n-2}\pi - \pi}{2^{n-1}}) = \\ = \cos(\frac{\pi}{2} - (\frac{2^{n-2}\pi - \pi}{2^{n-1}})) = \cos(\frac{\pi}{2} - \frac{2^{n-2}\pi}{2^{n-1}} + \frac{\pi}{2^{n-1}}) = \\ = \cos(\frac{2^{n-2}\pi}{2^{n-2}} - \frac{2^{n-2}\pi}{2^{n-1}} + \frac{\pi}{2^{n-1}}) = \cos(\frac{\pi}{2^{n-1}}), \quad (22)$$

$$\sin(\frac{\pi}{2^{n-1}}) = \cos(\frac{\pi}{2} - \frac{\pi}{2^{n-1}}) = \\ = \cos(\frac{2^{n-2}\pi}{2^{n-1}} - \frac{\pi}{2^{n-1}}) = \cos(\frac{2^{n-2}\pi - \pi}{2^{n-1}}) = \\ = \cos(\frac{(2^{n-2}-1)\pi}{2^{n-1}}). \quad (23)$$

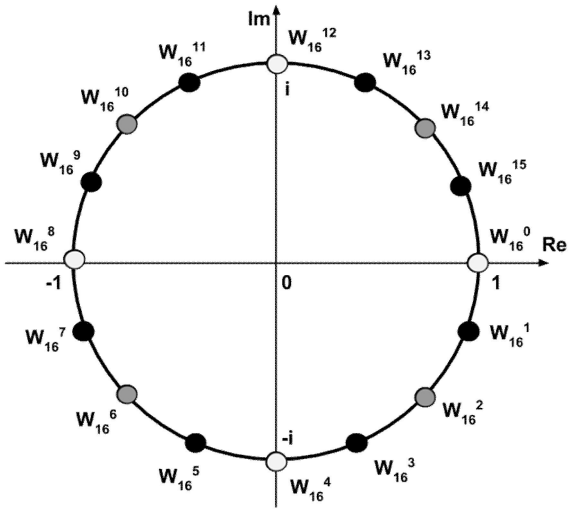

Fig. 1. The effect of the relations. On the figure the same colour indicates that roots which are in relation.

Finally, the newly generated properties (20) and (21) can

be written in more efficient and simpler forms (24) and (25), similarly as in (9) and (10), if we suppose that $N \geq 16$ and $0 < k < N/4$:

$$T^{\kappa}_{N_{Im}} = -T^{N/4-\kappa}_{N_{Re}}, \quad (23)$$

$$T^{\kappa}_{N_{Re}} = -T^{N/4-\kappa}_{N_{Im}}. \quad (24)$$

Figure 1 shows the location of the roots in the complex plane and the effect of (6), (9), (10), (24) and (25) relations (N = 16). This clearly illustrates that it is enough to compute the first (N/8 + 1) roots to the FFT algorithm.

The new relation reduces the number of complex roots again, thus it saves memory and/or computational capacity. Consequently, it is enough to calculate the first (N/8 + 1) roots because other roots can be derived.

IV. Experimental Results

In order to measure the effect of the proposed relations, we created a modified radix-2 FFT algorithm which utilises (6), (9), (10), (24) and (25). The new relations modify the original "butterfly" structure. Figure 2 illustrates a new structure where the proposed relations were applied in a radix-2 DIF FFT algorithm (N = 16).

The modified algorithm was compared with two general radix-2 implementations which can be found in [16], [17]. In most cases the implementation depends on the application type. We should implement the transformation according to the architecture, number of samples (N) and the applied programming language or abstraction level. This test mainly focuses on that case when the algorithms are sequential and the sample size is power of two. As it was mentioned previously, the efficiency of the algorithm depends on the architecture. Therefore, in the test four different architectures were used so that the survey will be more reliable.

– Microchip PIC32 32-bit MIPS processor
– Raspberry Pi (RPi) minicomputer
– BeagleBone Linux computer
– Simple PC

All used devices have different properties. The Chipkit Max32 board contains a PIC32 microcontroller which has 80 MHz clock signal and includes a 128K SDRAM. The central processing unit of the RPi is an ARM11 which is running at 700 MHz. The BeagleBone contains an AM335x 720 MHz ARM Cortex-A8 processor. Finally, the PC has an Intel Pentium 2.2 GHz processor. Obviously, the speed of an algorithm depends on lots of parameters but now these are insignificant. During the test, the three algorithms (modified FFT, FFT A and FFT B) get a random signal with different size and calculate the transformation 100 times. Finally, the program gives back an average value about the runtime.

Each FFT algorithms were implemented in ANSI C programming language, because all the four devices ensure C compiler. Table I shows the test results where the time dimension is in second.

TABLE I. TEST RESULTS.

Device	Samples (N)	Modified FFT	FFT A	FFT B
PIC32	1024	0.051	0.199	0.198
RPi	1024	0.009	0.017	0.013
BeagleBone	1024	0.006	0.030	0.034
PC	1024	0.0003	0.001	0.0007
PIC32	8192	0.650	2.271	2.259
RPi	8192	0.107	0.177	0.155
BeagleBone	8192	0.072	0.344	0.368
PC	8192	0.003	0.012	0.007
PIC32	65536	6.146	21.698	21.584
RPi	65536	1.297	1.629	1.569
BeagleBone	65536	0.849	3.322	3.375
PC	65536	0.038	0.115	0.069

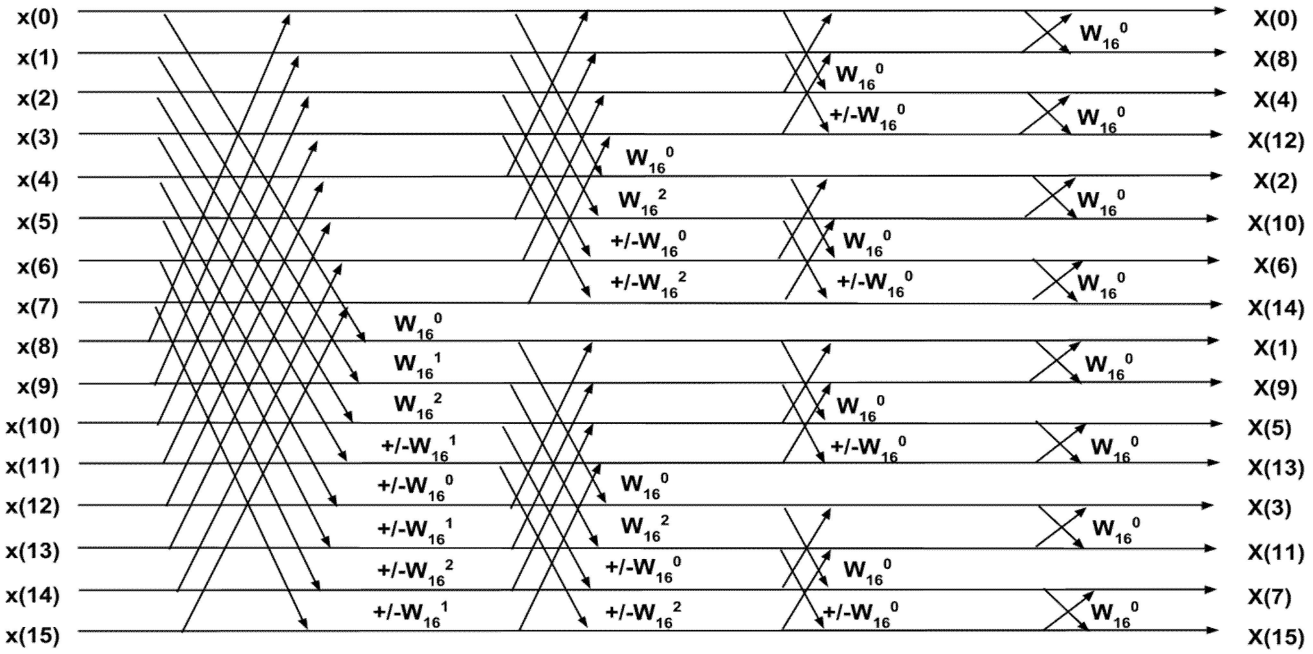

Fig. 2. The "butterfly" structure of the modified algorithm. On the figure +/- indicates the correct sing according to the proposed relations.

V. CONCLUSIONS

In the paper we presented a new relation between "twiddle factors" in (20) and (21) and its proof in (22) and (23) which was not mentioned in the literature yet. This relation reduces the number of necessary roots to the fast Fourier transformation. Moreover, we proposed more efficient forms to an existing and to the new relation in (9), (10), (24) and (25). To support the proposed relations, we made a survey where a modified radix-2 FFT was compared to two other algorithms. The experimental results clearly indicate that the modified FFT is more efficient than other general algorithms. Since less roots are enough for the algorithm thus the root calculation time is greatly reduced. In addition, the test results show architecture dependence of algorithms. FFT A is faster than FFT B on the BeagleBone while on other devices FFT A is slower. Furthermore, the modified FFT is more efficient on PIC32 and BeagleBone than on the RPi.

Today, the Internet of Things (IoT) is a dynamically extending research area. The IoT is a network of different things or objects which can communicate sense and interact to each other via the Internet [18]–[21]. In many cases, the objects are little devices which belong to two main categories: controllers and sensors. A controller can be a mini-computer, microcontroller and FPGA [22]–[23]. Generally, the controllers perform every data processing. However, most controllers have limited calculation and storage capacity. This can cause problem when the controller performs DSP algorithms or applications. It means that, an optimised FFT algorithm which utilises every relation (in optimised form) between the complex roots can be very useful on such devices. Moreover, the power consumption will be similarly more effective.

REFERENCES

[1] J. W. Cooley, J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", *Mathematics of Computation*, vol. 19, pp. 297-301, 1965. [Online]. Available: http://dx.doi.org/10.1090/S0025-5718-1965-0178586-1

[2] P. Duhamel, M. Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art", *Signal Processing*, vol. 19, pp. 259–299, 1990. [Online]. Available: http://dx.doi.org/10.1016/0165-1684(90)90158-U

[3] S. Bouguezel, M. O. Ahmad, M. N. S. Swamy, "An efficient split-radix FFT algorithm", in *Proc. Int. Symp. on Circuits and Systems*, Bangkok, 2003, pp. 25–28. [Online]. Available: http://dx.doi.org/10.1109/iscas.2003.1205774

[4] S. C. Chan, "Split vector-radix fast Fourier transform", *IEEE Trans. Signal Processing*, vol. 40, pp. 2029–2039, 1992. [Online]. Available: http://dx.doi.org/10.1109/78.150004

[5] S. Bouguezel, M. Omair Ahmad, M. N. S. Swamy, "Improved radix-4 and radix-8 FFT algorithms", in *Proc. Int. Symp. on Circuits and Systems*, Vancouver, 2004, pp. 561–564. [Online]. Available: http://dx.doi.org/10.1109/iscas.2004.1328808

[6] S. Bouguezel, M. Omair Ahmad, M. N. S. Swamy, "A new radix-2/8 FFT algorithm for length-q x 2^m DFTs", *IEEE Trans. on Circuits and System*, vol. 51, pp. 1723–1732, 2004. [Online]. Available: http://dx.doi.org/10.1109/TCSI.2004.834508

[7] D. Rodriguez, "A new FFT algorithm and its implementation on the DSP 96002", in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Toronto, 1991, pp. 2189–2192.

[8] D. A. Bader, V. Agarwal, "FFTC: fastest Fourier transform for the IBM cell broadband engine", *Lecture Notes in Computer Science*, vol. 4873, pp. 172–184, 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77220-0_19

[9] Z. Cui-xiang, H. Gou-qiang, H. Ming-He, "Some new parallel fast Fourier transform algorithms", in *Proc. 6th Int. Conf. on Parallel and Distributed Computing, Applications and Technologies*, Dalian, 2005, pp. 624–628. [Online]. Available: http://dx.doi.org/10.1109/pdcat.2005.224

[10] Yuan-Nan Chang, K. K. Parhi, "Efficient FFT implementation using digit-serial arithmetic", in *Proc. IEEE Workshop on Signal Processing Systems*, Taipei, 1999, pp. 645–653.

[11] E. Chu, A. George, "FFT algorithms and their adaptation to parallel processing", *Linear Algebra and its Applications*, vol. 284, pp. 95–124, 1998. [Online]. Available: http://dx.doi.org/10.1016/S0024-3795(98)10086-1

[12] L. Jia, B. Li, Y. Gao, H. Tenhunen, "Implementation of a low power 128-point FFT", in *Proc. 5th Int. Conf. on Solid-State and Integrated Circuit Technology*, Beijing, 1998, pp. 369–372.

[13] A. Mertins, *Signal Analysis: Wavelets, Filter Banks, Time-frequency Transforms and Applications.* Chichester, England: John Wiley & Sons, 1999, ch 4. [Online]. Available: http://dx.doi.org/10.1002/0470841834

[14] J. Suto, S. Oniga, Gy. Hegyesi, "A simple fast Fourier transformation algorithm to microcontrollers and mini computers", in *18th Int. Conf. on Intelligent Engineering Systems*, Tihany, 2014, pp. 61–65. [Online]. Available: http://dx.doi.org/10.1109/ines.2014.6909342

[15] P. Shirley, S. Marschner, M. Ashikhmin, M. Gleicher, N. Hoffman, G. Johnson, T. Munzner, E. Reinhard, K. Sung, W. B. Thompson, P. Willemsen, B. Wyvill, *Fundamentals of Computer Graphics*, CRC press, Boca Raton, 2009, pp. 18–20.

[16] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing.* USA: California Technical Publisher, 1999, pp. 141–242.

[17] C. V. Loan, *Computational Framework for the fast Fourier Transform*. USA: Siam, 1992, ch. 1. [Online]. Available: http://dx.doi.org/10.1137/1.9781611970999

[18] Gy. Terdik, Z. Gal, "Advances and practice in Internet of Things: a case study", in *Proc. IEEE 4th Int. Conf on Cognitive Infocommunications*, Budapest, 2013, pp. 435–440. [Online]. Available: http://dx.doi.org/10.1109/coginfocom.2013.6719286

[19] C. Lung, S. Oniga, A. Buchman, A. Tisan, "Wireless data acquisition system for IoT applications", *Carpathian J. of Electronic and Computer Engineering*, vol. 6, pp. 64–67, 2013.

[20] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions", *Future Generation Computer Syst.*, vol. 29, pp. 1645–1660, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.01.010

[21] Z. Gal, B. Almasi, T. Daboczi, R. Vida, S. Oniga, S. Baran, I. Farkas, "Internet of Things: application areas and research results of the FIRST project", *Infocommunications J.*, vol. 6, pp. 37–44, 2014.

[22] S. Oniga, J. Suto, "Human activity recognition using neural networks", in *Proc. 15th Int. Carpathian Control Conf. (ICCC 2014)*, Velke Karlovice, Czech Republic, 2014, pp. 403–406. [Online]. Available: http://dx.doi.org/10.1109/carpathiancc.2014.6843636

[23] J. Suto, S. Oniga, I. Orha, "Microcontroller based health monitoring system", in *Proc. 19th Int. Symp. for Design and Technology in Electronic Packaging*, Galati, 2013, pp. 227–230. [Online]. Available: http://dx.doi.org/10.1109/siitme.2013.6743679