SIMPLIFIED ADDRESSING SCHEME FOR MIXED RADIX FFT ALGORITHMS

Cuimei Ma[1], *Yizhuang Xie*[1], *He Chen*[1], *Yi Deng*[2], *Wen Yan*[1]

[1]Beijing Institute of Technology, Beijing, 100081, China;
[2]Virginia Polytechnic Institute and State University, Arlington, VA 22044 USA

ABSTRACT

A mixed radix algorithm for the in-place fast Fourier transform (FFT), which is broadly used in most embedded signal processing fields, can be explicitly expressed by an iterative equation based on the Cooley-Tukey algorithm. The expression can be applied to either decimation-in-time (DIT) or decimation-in-frequency (DIF) FFTs with ordered inputs. For many newly emerging low power portable computing applications, such as mobile high definition video compressing, mobile fast and accurate satellite location, etc., the existing methods perform either resource consuming or non-flexible. In this paper, we propose a new addressing scheme for efficiently implementing mixed radix FFTs. In this scheme, we elaborately design an accumulator that can generate accessing addresses for the operands, as well as the twiddle factors. The analytical results show that the proposed scheme reduces the algorithm complexity meanwhile helps the designer to efficiently choose an arbitrary FFT to design the in-place architecture.

Index Terms—Fast Fourier Transform, mixed radix, address generation, in place, arithmetical complexity.

1. INTRODUCTION

Fast Fourier transform (FFT) algorithms play a key role in improving the feasibility of discrete Fourier transform (DFT), which is broadly used in most digital processing applications. For practical FFT uses, there are radix-2, radix-4 and split-radix [1] FFTs. Meanwhile, the research on radix-2^k [2] FFT has resulted in the instantiation of such methods as radix-2^2 [3], radix-2^3 [4], and even radix-2^4 FFT [5]. Although the derivation and programming are intuitive for radix-2^k FFT, the drawback is that the number of points has to be restricted to powers of two or four, which restricts its application in resource-limited portable computing scenarios.

Recently, an optimal choice for the FFT size is in demand and many non-power-of-two FFTs, such as 3^k and 6^k- point FFT, have been studied [6,7]. However, those methods are all in radix-$q\times2^k$ FFTs [8], where q is prime, such as 3, 5, and 7, et al. Because the mixed radix FFT can be used in general scenarios, it becomes practical and useful. Some mixed radix FFT algorithms are studied, such as radix-2/4 [9,10] and radix-$2/2^k$ FFTs. They are based on radix-2 FFT. However, if the padding-zero method is used to satisfy the radix-2^kFFT, it will consume a larger amount of memory than the mixed radix FFT. Since the memory cost is a significant part of the FFT processor, minimizing the necessary size is an effective way for the area reduction. Therefore, arbitrary mixed radix FFTs problems are discussed in [11, 12].

Two methods, the pipelined and memory-based architectures [13, 14], have been proposed for different applications in various FFT processors. Although much higher throughput than memory based designs, the pipeline architectures have a larger area cost. Therefore, the in-place strategy [15] is taken and only one memory with N complex words is needed.

However, the in-place strategy has a complex design circuit, which has to generate addresses for both operands and twiddle factors. Demuth [11] proposed a nested loop index generation algorithm to index inputs and outputs of FFT stages and another way to index twiddle factor exponents. This method needs many parameters to get the address, which is difficult to implement using hardware. Hsiao [12] gave the index mapping method for the generalized mixed radix algorithm with some complex modulo operations. A bit-level representation of the accessing rule was mentioned by Sorokin [16], which is used in different processing stages for a radix-2/4 FFT. In this paper, we will extend it to an arbitrary mixed radix FFT.

An iterative expression that is applied to radix-r_1/r_2 in-place architecture is derived. The N data are stored in RAM and the N twiddle factors are in ROM. An accumulator is set to make the accessing address map easily to the hardware circuits and there is no modulo operation. The illustrative example is based on radix-2/3 decimation-in-time (DIT) in-place 12-point FFT. By this method, an appropriate FFT size is chosen to minimize the memory size and a simplified address control is designed.

2. REVIEW OF MIXED-RADIX ALGORITHM

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

The N-point DFT of an N-point sequence $\{x(n)\}$,

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn},\ 0\le k\le N-1 \tag{1}$$

where $W_N^{kn}=\exp(-j2\pi kn/N)$, and $j=\sqrt{-1}$.

Suppose that the FFT size satisfies $N=r_1^{s_1}\times r_2^{s_2}$, where r_1 and r_2 are two radices, s_1 and s_2 are the corresponding integer powers, and a parameter s is $s=s_1+s_2$. We assume that the algorithm uses s_1 radix-r_1 stages followed by s_2 radix-r_2 stages. According to Cooley-Tukey algorithm, the time and frequency indices, i.e. n and k, are analyzed. The expressions for both n and k with s digits in terms of r_1, r_2, s_1 and s_2 are obtained as follows.

$$\begin{cases} n=r_1^{s_1}\times r_2^{s_2-1}\times n_{s-1}+\ldots+r_1^{s_1}\times r_2^{1}\times n_{s_1+1}+r_1^{s_1}\times\cdot n_{s_1}\\ \quad +r_1^{s_1-1}\times n_{s_1-1}+\ldots+r_1^{1}\times n_1+n_0,\\ k=r_1^{s_1-1}\times r_2^{s_2}\times k_{s-1}+\ldots+r_1^{1}\times r_2^{s_2}\times k_{s_2+1}+r_2^{s_2}\times k_{s_2}\\ \quad +r_2^{s_2-1}\times k_{s_2-1}+\ldots+r_2^{1}\times k_1+k_0, \end{cases} \tag{2}$$

where if $0\le i\le s_1-1$, $n_i\in[0,r_1-1]$, else $n_i\in[0,r_2-1]$; if $0\le i\le s_2-1, k_i\in[0,r_2-1]$, else $n_i\in[0,r_1-1]$.

For brevity, assume that $c_i=r_1^i$ when $i=0,1,\ldots,s_1$, $c_{s_1+i}=r_1^{s_1}\times r_2^i$ when $i=1,\ldots,s_2-1$. $c'_i=r_2^i$ when $i=0,\ldots,s_2$, $c'_{s_2+i}=r_2^{s_2}\times r_1^i$ when $i=1,2,\ldots,s_1-1$. Therefore, Eq.(2) is rewritten as

$$\begin{cases} n=c_{s-1}\times n_{s-1}+c_{s-2}\times n_{s-2}+\ldots+c_1\times n_1+c_0\times n_0,\\ k=c'_{s-1}\times k_{s-1}+c'_{s-2}\times k_{s-2}+\ldots+c'_1\times k_1+c'_0\times k_0. \end{cases} \tag{3}$$

For notational convenience, Eq. (3) can be written as

$$\begin{cases} n=(n_{s-1}n_{s-2}\ldots n_2n_1n_0)\big|_{mixed\ radix},\\ k=(k_{s-1}k_{s-2}\ldots{}_3k_2k_1k_0)\big|_{mixed\ radix}. \end{cases} \tag{4}$$

When Eq. (3) is substituted into Eq. (1), we decompose N-point DFT into s iterations and the mth ($1\le m\le s$) iteration is as follows.

$$x_m(k_0k_1\ldots k_{m-1}n_{s-1-m}n_{s-2-m}\ldots n_0)=\sum_{n_{s-m}=0}^{r'-1}x_{m-1}(k_0k_1\ldots k_{m-2}n_{s-m}n_{s-m-1}\ldots n_0)W_N^{k(c_{s-m}n_{s-m})} \tag{5}$$

where $r'\in(r_1,r_2)$. When m satisfies $1\le m\le s_2$, $r'=r_2$, otherwise $r'=r_1$.

From Eq. (5), $x_{m-1}(k_0k_1\ldots k_{m-2}n_{s-m}n_{s-m-1}\ldots n_0)$ denotes one of r' operands for radix-r' butterfly in the mth stage and $x_m(k_0k_1\ldots k_{m-1}n_{s-m-1}\ldots n_0)$ represents the butterfly output. Because the in-place algorithm is used, we can analyze $Addr(m)=(k_0k_1\ldots k_{m-1}n_{s-m-1}n_{s-m-2}\ldots n_0)\big|_{mixed\ radix}$ to explore the address generations in order to get the operands.

Table 1. Variable digit with cycles in each stage.

Stage	Address representation	Variable digit
$1\le m<s$	$(k_0k_1\ldots k_{m-1}n_{s-m-1}n_{s-m-2}\ldots n_0)\big\|_{mixed\ radix}$	k_{m-1}
s	$(k_0k_1k_2\ldots k_{s-1})\big\|_{mixed\ radix}$	k_{s-1}

Table 2. Different radix of C_i in each stage.

Stage	$C_i(i=0,\ldots,s-1)$	Radix
$m\ (1\le m\le s_2)$	$C_1,C_2,\ldots C_{s_1}$	r_1
	$C_0,C_{s_1+1},C_{s_1+2},\ldots,C_{s-1}$	r_2
$m\ (s_2<m\le s)$	$C_{s_1-1},\ldots,C_1,C_0$	r_1
	$C_{s_1},C_{s_1+1},C_{s_1+2},\ldots,C_{s-2},C_{s-1}$	r_2

Furthermore, $W^{k(c_{s-m}\times n_{s-m})}$ is the corresponding twiddle factor. We store the twiddle factors $W^i=\exp(-j2\pi i/N)$ in a lookup table sequentially, where i ranges from 0 to N-1. We can get them from the lookup table by analyzing the exponent part, i.e. $k(c_{s-m}n_{s-m})$.

The following section will present the addressing scheme by the iteration representation of n and k in the mth stage.

3. THE PROPOSED ADDRESSING SCHEME

3.1. Address generation for operands

The r' consecutive addresses for the r' operands of radix-r' butterfly can be obtained in r' clock cycles. For every stage, we should find which digit is variable from 0 to $r'-1$ and the other digits are constant in $(k_0k_1\ldots k_{m-1}n_{s-m-1}n_{s-m-2}\ldots n_0)\big|_{mixed\ \ radix}$. The variable digit for every stage is listed in Table 1.

Assume an accumulator, which is represented by $ACC=(C_{s-1}C_{s-2}C_{s-3}\ldots C_2C_1C_0)\big|_{mixed\ radix}$ for mapping the data addresses. C_i is the ith digit, C_{s-1} is the most significant digit and C_0 the least significant digit. Only C_0 keeps varying with cycles, similar to k_{m-1} at the mth stage, just as shown in Table 1.

Because each digit in $Addr$ is either r_1 or r_2, C_i in ACC is either r_1 or r_2. Therefore, there list the value of C_i in Table 2. The relationship between $Addr$ and ACC

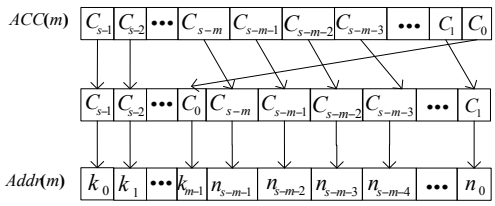

Fig.1. Addresses generating for *N*-point FFT using ACC in the *m*th stage.

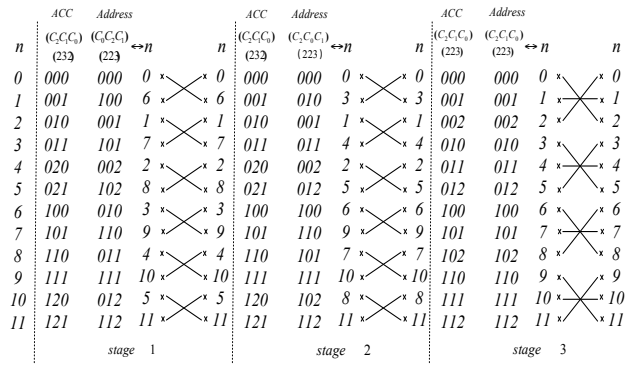

Fig.2. Access for operands of 12-point radix2-3 FFT.

in the *m*th stage is illustrated in Fig.1.

Fig.1 shows that the address of the *m*th stage of FFT is obtained from ACC. The third row is the address, which is represented using time and frequency indices, i.e. n and k. Fig.2 illustrates the addressing scheme for 12-point FFT, i.e. $r_1 = 3$, $r_2 = 2$, $s_1 = 1$, $s_2 = 2$. In the first column, n represents the number of the memory depth from 0 to 11 and also means the time sequence. When the stage of the FFT is the first, $ACC(1) = (C_2C_1C_0) = (232)$. According to Fig.1, C_0 is shifted to the left of the C_2, and we obtain that $Addr(1) = (C_0C_2C_1) = (223)$. $Addr(1)$ is variable as shown in column 3 and the addresses of the operands in memory are achieved by the conversion the mixed-radix numbers to the decimal representations and the values of n are listed in column 4. For example, the first radix-2 butterfly computation, the addresses of the two operands are 0 and 6 separately. The outputs are stored in the same addresses. For the next two stages, $ACC(2) = (C_2C_1C_0) = (232)$ and $ACC(3) = (C_2C_1C_0) = (223)$. $Addr(2) = (C_2C_0C_1) = (223)$ by shifting C_0 to the left of C_1 and $Addr(3) = ACC(3)$. Therefore, we can get the right addresses of the operands according to Fig.1.

3.2. Address generation for twiddle factors

Substitute the expression k in Eq.(2) into $W^{k(c_{s-m} \times n_{s-m})}$, and the following expression is obtained.

Table 3. The expression of β in each stage.

Stage	Accessing address, ((*s*-1)-digit)
1	$(000...00)\big\vert_{mixed\ radix}$
$m(2 \le m < s)$	$(C_{s-m+1}C_{s-m+2}...C_{s-1}0...00)\big\vert_{mixed\ radix}$
s	$(C_1C_2C_3...C_{s-1})\big\vert_{mixed\ radix}$

$$W^{(r_1^{s-1} \times r_2^{s} \times k_{s-1} + ... + r_2^{s} \times k_{s_2} + r_2^{s_2-1} \times k_{s_2-1} + ... + r_2^{1} \times k_1 + k_0)(c_{s-m} \times n_{s-m})}$$
$$= \underbrace{W^{c_{s-m}n_{s-m}\sum_{i=m}^{s-1} c'_i k_i}}_{Part1} \underbrace{W^{c_{s-m}n_{s-m}c'_{m-1}k_{m-1}}}_{Part2} \underbrace{W^{c_{s-m}n_{s-m}\sum_{i=0}^{m-2} c'_i k_i}}_{Part3} \quad (6)$$

There are three parts of the expression. We describe them separately and analyze which part is used to get the address of twiddle factor.

Part 1: We obtain that $W^{c_{s-m}n_{s-m}\sum_{i=m}^{s-1} c'_i k_i} = 1$ in each stage.

Part 2: $W^{c_{s-m}n_{s-m}c'_i k_i}$ denotes r'-point DFT matrix, for example, $[1\ 1; 1\ -1]$ is a 2-point DFT matrix. Different stages use different r'-point DFT matrix. When $1 \le m \le s_2$, it is a r_2-point DFT matrix; otherwise, it is a r_1-point DFT matrix.

The first two parts have nothing to do with the address of twiddle factor, so we can get the address from part 3.

We suppose a parameter $\beta' = \sum_{i=0}^{m-2} c'_i k_i$ when $m = 2,3,...,s$, and $\beta' = 0$ when $m = 1$. β' can be expressed by $(C_{s-m+1}C_{s-m+2}...C_{s-2}C_{s-1})\big\vert_{mixed-radix}$ according to ACC when $m = 2,3,...,s$. Let $\beta = c_{s-m} \times \beta'$. The Table 3 lists the expression β in each stage.

Suppose $ACC_{reverse} = (C_0C_1C_2...C_{s-3}C_{s-2}C_{s-1})\big\vert_{mixed\ radix}$ and it means the digit reverse of the ACC. Table 3 shows that the expression of β is related to $ACC_{reverse}$. β can be represented in VHDL when $m = 2,3,...,s$ as follows :
$\beta(m) <= \left[ACC_{reverse}((m-2)\ downto\ 0)\ \&\ zeros((s-2)\ downto\ (m-1)) \right]$, where $zeros$ denotes a zero vector.

Because $n_{s-m} = 0,1,\cdots,r'-1$ in part 3, the addresses of the r' twiddle factors for a butterfly unit in stage *m* can be acquired by n_{s-m} multiplication with $\beta(m)$, as Eq.(7). The accessing addresses of twiddle factors are obtained.

$$Addr_{twi}(m) = \begin{cases} 0, & n_{s-m} = 0; \\ \beta(m), & n_{s-m} = 1; \\ ... & \\ (r'-1) \times \beta(m), & n_{s-m} = r'-1. \end{cases} \quad (7)$$

Fig.3 shows the accessing address for twiddle factors of 12-point FFT. The column n is the same meaning as that is in the first column in Fig.2. For each butterfly

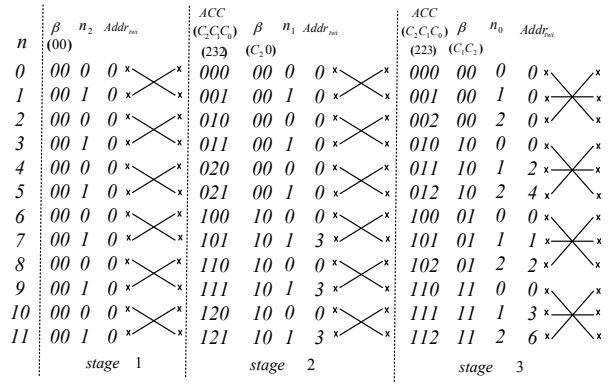

Fig.3. Accesses for twiddle factors of radix-2/3 FFT.

computation, the left addresses of it in Fig.2 are for the input operands the ones in Fig.3 are for the twiddle factors.

Therefore, we easily get the addresses of the operands and the twiddle factors using one accumulator for the mixed radix FFTs. The *ACC* satisfies the conditions listed in Table2. Figure 1 gives the relation between the addresses of operands and the accumulator. Table 3 lists the relation between the addresses of the twiddle factors and the same accumulator.

4. COMPARISONS

The architectures of the in-place algorithms are generally consistent. The key comparison part is the address generation. Meanwhile, because the twiddle factor is not considered in [12], only a comparison on the address generation for the operands between the method in [12] and the proposed scheme is given. The intermediate values in [11] are hard to be implemented in hardware, so there is no comparison with the scheme in [11].

For simplicity, a 12-point radix-2/3 FFT is taken as the illustrative example. Fig.4 (a) shows the proposed scheme and Fig.4 (b) shows the scheme in [12]. By comparison, the novel scheme has two characteristics:

(I) It keeps the architecture of FFT consistent for every stage. Thus, we only design one architecture to get the accessing address by the *ACC* . For the method in [12], the architecture of address generation for each stage is different. We have to design three different architectures to obtain the corresponding accessing addresses for the each stage. If the FFT point becomes larger, more resources are needed for the address generation.

(II)It requires no complex modulo operations. The larger the FFT size is, the more modulo operations are needed in [12].

Table 4 lists the number of mathematical operations for the 12-point FFT.

Therefore, the proposed scheme simplifies the complexity of generating addresses.

5. CONCLUSIONS AND DISCUSSION

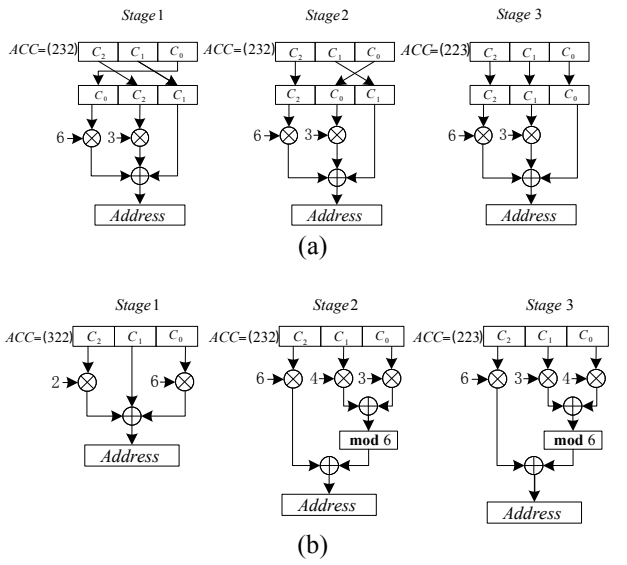

Fig.4. Address generation for 12-point radix-2/3 FFT by (a) the proposed method, and (b) the method in [12].

Table 4. Mathematical operations comparisons of Hsiao's design and ours

Scheme	Our Scheme	Hsiao's Design [12]
2-input addition	2	6
2-input multiplication	2	8
Modulo operation	0	2

An iterative approach can be applied to analyze the address generation for mixed-radix in-place FFT with ordered inputs. The accessing addresses for the operands and the twiddle factors can be achieved from one accumulator. It is easy to implement this accumulator in hardware circuits.

With respect to the tradeoff, mixed radix FFT costs more than one butterfly unit compared with fixed radix FFT. However, a unified architecture, just like that described in [17], can be achieved to compute arbitrary two butterflies. Therefore, the increased resources of the butterfly unit with unified architecture have small impact on the overall resources.

6. REFERENCES

[1] T.Z. Sung, H.C. Hsin, et al, "Low-power and high-speed CORDIC-based split-radix FFT processor for OFDM systems," *Digit. Signal Prog*, vol. 20, no. 2, pp. 511-527, 2010.

[2] M. Garrido, J. Grajal, M.A. Sanchez, et al, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 21, no.1, pp. 23-32, 2013.

[3] J. Li, F. Liu, T. Long, et al, "Research on Pipeline R22SDF FFT," *IET Int. Radar Conf*,, Guilin, China, Apr. 20-22 2009, pp.1-5.

[4] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-Point Fourier Transform Chip for High-Speed Wireless LAN Application Using OFDM," *IEEE J. Solid-State Circuit,* vol. 39 no.3, pp.484-493, 2004.

[5] H. Liu and H. Lee, "A high performance four parallel 128/64-point radix-2^4 FFT/IFFT processor for MIMO-OFDM systems," *IEEE Asia Pacific Conf. Circuits Syst. (APCCSA)*, Macao, China, Nov. 30-Dec. 3 2008, pp.834-837.

[6] E. Dubois and A. Venetsanopoulos, "A new algorithm for the radix-3 FFT," *IEEE Trans. Acoustics, Speech, Signal Prog, vol.*29, no.4, pp. 939-941, 1982.

[7] D. Takahash, "A new radix-6 FFT algorithm suitable for multiply-add instruction," *IEEE Int. Conf. Acoustics, Speech, Signal Prog*., Istanbul, Turkey, Jun. 5-9 2000, vol.6, pp.3343-3346.

[8] S.S. Deng, Y. Sun, L.S. Zhang, et al, "Design of High-Speed FFT Processor for Length N=$q\times2^m$," *J. Comput. Res. Dev.*, vol. 45, no. 8, pp. 1430-1438, 2008.

[9] A.T. Jacobson, D.N. Truong, and B.M. Baas, "The design of a reconfigurable continuous-flow mixed-radix FFT Processor," *IEEE Int. Sym. Circuit Syst.,* Taipei, China, May 24-27 2009, pp.1133-1136.

[10] H. Xiao, An Pan, Y. Chen, et al, "Low-cost reconfigurable VLSI architecture for fast Fourier transform,"*IEEE Trans. Consum*。Electron, vol. 54, no.4, pp.1617-1622, 2008.

[11] G.L.Demuth, "Algorithms for defining mixed radix FFT flow graphs," *IEEE trans. acoustics, speech, signal prog*, vol. 37, no. 9, pp. 1349-1358, 1989.

[12] C.F.Hsiao, Y.Chen, C.Y.Lee, "A generalized mixed-radix algorithm for memory-based FFT processors," *IEEE trans. circuit* syst.-II, vol. 57, no. 1, pp.26-30, 2010.

[13] C. Yu, M.H.Yen, P.A. Hsiung, et al, "A Low-Power 64-point Pipeline FFT/IFFT Processor for OFDM Applications," *IEEE Trans. Consum. Electron.*, vol. 57, no.1, pp.40-45, 2011.

[14] C.L. Wey, S.Y. Lin, and W.C. Tang, "Efficient Memory-based FFT Processors for OFDM Applications," *IEEE Int. Conf. Electro/Information Technol,* 2007, Chicago, US, May 17-20, pp. 345-350.

[15] B.G. Jo and M.H. Sunwoo, "New Continuous-Flow Mixed-Radix (CFMR) FFT Processor Using Novel In-Place Strategy," *IEEE Trans. Circuit Syst.-I*, vol. 52, no. 5, pp. 911-919, 2005.

[16] H. Sorokin, J. Takala, "Conflict-free parallel access scheme for mixed-radix FFT supporting I/O permutations," *IEEE Conf. Aconstics, Speech, Signal Prog (ICASSP2011*), Prague, C.Z., May 22-27 2011, pp. 2709-1712.

[17] F. Qureshi, M. Garrido and O. Gustafsson, "Unified architecture for 2, 3, 4, 5, and 7-point DFTs based on Wingrad Fourier transform algorithm," Electron. Lett., vol. 49, no.5, pp. 348-349, 2013.