ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.13, No.18, 1–6

High-speed design for mixed radix FFT algorithm based on multi-bank memory strategy

Cuimei Ma[a)] **and Yanfei Wang**

Institute of Electronics, Chinese Academy of Sciences,
19 North Sihuan Road, Haidian, Beijing 100190, China

a) *macuimei@126.com*

Abstract: A multibank memory structure is introduced for mixed radix FFT algorithms to improve the real-time performance. In this structure, only one butterfly unit implementing mixed radix butterfly computation is required and the FFT computation keeps continuous. At last, comparisons are made to prove that the method in this paper is valid.

Keywords: mixed-radix FFT, low complexity, multibank memory structure, real-time performance

Classification: Integrated circuits

References

[1] R. C. Agarwal and J. W. Cooley: "Vectorized mixed radix discrete Fourier transform algorithms," Proc. IEEE **75** (1987) 1283 (DOI: 10.1109/PROC.1987.13880).

[2] X. J. Li, *et al.*: "A low power and small area FFT processor for OFDM demodulator," IEEE Trans. Consum. Electron. **53** (2007) 274 (DOI: 10.1109/TCE.2007.381685).

[3] Z. B. Zhang and X. W. Li: "Design and implementation of transform precoding in LTE system," Application of Integrated Circuits **36** (2010) 54.

[4] D. F. Liu, *et al.*: "Design and implementation of configurable FFT/IFFT in WIMAX system," J. Xidian Univ. **37** (2010) 813.

[5] S. S. Deng, *et al.*: "Design of high-speed FFT processor for length $N = q \times 2^m$," J. Comput. Res. Dev. **45** (2008) 1430.

[6] A. T. Jacobson, *et al.*: "The design of a reconfigurable continuous-flow mixed-radix FFT processor," IEEE Int. Sym. Circuit Syst. (2009) 1133 (DOI: 10.1109/ISCAS.2009.5117960).

[7] G. L. DeMuth: "Algorithms for defining mixed radix FFT flow graphs," IEEE Trans. Acoust. Speech Signal Process. **37** (1989) 1349 (DOI: 10.1109/29.31290).

[8] B. G. Jo and M. H. Sunwoo: "New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy," IEEE Trans. Circuits Syst. I, Reg. Papers **52** (2005) 911 (DOI: 10.1109/TCSI.2005.846667).

[9] C. F. Hsiao, *et al.*: "A generalized mixed-radix algorithm for memory-based FFT processors," IEEE Trans. Circuits Syst. II, Exp. Briefs **57** (2010) 26 (DOI: 10.1109/TCSII.2009.2037262).

[10] H. Sorokin and J. Takala: "Conflict-free parallel access scheme for mixed-radix FFT supporting I/O permutations," IEEE Conf. Acoustics, Speech, Signal Prog (ICASSP) (2011) 1709 (DOI: 10.1109/ICASSP.2011.5946830).

[11] C. M. Ma, *et al.*: "An efficient design for general mixed radix FFT processors,"

DOI: 10.1587/elex.13.20160646
Received June 29, 2016
Accepted July 22, 2016
Publicized August 5, 2016
Copyedited September 25, 2016

IEICE Electron. Express **13** (2016) 20160060 (DOI: 10.1587/elex.13.20160060).
[12] W. Li and L. Wanhammer: "A pipeline FFT processor," Proc. Workshop Signal Processing Systems (1999) 654 (DOI: 10.1109/SIPS.1999.822372).
[13] Y. T. Ma: "An effective memory addressing scheme for FFT processors," IEEE Trans. Signal Process. **47** (1999) 907 (DOI: 10.1109/78.747802).
[14] L. G. Johnson: "Conflict free memory addressing for dedicated FFT hardware," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **39** (1992) 312 (DOI: 10.1109/82.142032).

1 Introduction

It has always been recognized as very essential to digital signal processing applications that one have available a general mixed radix FFT program which is, at the same time, operationally efficient [1]. With the development of the orthogonal frequency-division multiplex technique in communication systems including DMB-T [2], LTE SC-FDMA [3], 3GPP LTE [4], et al., the mixed radix FFT becomes practical and useful and gains more and more attentions [5, 6, 7, 8, 9, 10, 11].

Generally, pipeline architectures are adopted to improve the real-time performance of FFTs at the cost of a great of hardware [12]. Therefore, many researches focus on memory-based architecture [9, 11], especially for mixed radix FFTs, which is mainly discussed in this letter. A multibank memory structure is presented because of the poor real-time performance of the in-place strategy for FFT computation.

Ma gives an efficient scheme to improve the processing rate for radix-2 FFTs [13]. Johnson provides a good viewpoint for multibank memory structure, but it only works for fixed radix-r FFT [14]. The method in [9] is similar to the one [14] which is modified to be used for mixed radix FFT. This method requires some adders and one module computation for distributing the input data to memory banks. A novel multibank memory structure is introduced. The key point of this method lies in the proper distribution of the input data to keep the following computations of FFTs and DFTs independent and parallel. Meanwhile, it needs neither addition nor modulo computations and the hardware design is simplified.

The rest of this paper is organized as follows. In Section 2, the existing method of accessing data address is described and its limitations are analyzed; meanwhile, the proposed method of address access generation is presented. A modified multibank memory structure is proposed in Section 3. Section 4 shows the 3780-point FFT design as an illustrative example and gives some comparisons; moreover, a configurable butterfly unit is discussed. Finally, conclusive remarks are provided in Section 5.

2 Existing method for mixed radix algorithm

The definition of a DFT is $X(k) = \sum_{n=1}^{N-1} x(n)W_N^{nk}$. For the size $N = N_1N_2$, a method to compute its DFT is just as follows:

DOI: 10.1587/elex.13.20160646
Received June 29, 2016
Accepted July 22, 2016
Publicized August 5, 2016
Copyedited September 25, 2016

$$\begin{cases} n = N_1 n_1 + n_0, & n_1, k_0 = 0, 1, \ldots, N_2 - 1 \\ k = N_2 k_1 + k_0, & n_0, k_1 = 0, 1, \ldots, N_1 - 1. \end{cases} \quad (1)$$

Eq. (1) shows the relationship between the indexes n and k and the index vectors (n_1, n_0) and (k_1, k_0), transforming one dimension $[0, N-1]$ into two dimensions $[0, N_1 - 1] \times [0, N_2 - 1]$.

Similarly, if $N = \prod_{i=1}^{t} r_i^{s_i}$ (where r_i and s_i mean a radix value and the corresponding integer power for the i^{th} stage FFT computation.) denotes the number of points as a general case.

In the following proposed design, the mixed radix FFT algorithm proposed in [11] based on Eq. (1) is applied regardless of the relationship between the factors [9]. Based on this method, a proposed method with multibank memory requires less modulo operations than the existing methods as well as improves the real-time performance.

3 Proposed multibank memory structure for mixed radix FFT

A "pipeline & parallel" structure for multibank memory is presented to improve the processing speed of the mixed radix FFT design. As it is about multibank memory structure, the proper number of memory banks is first discussed and then we analyze the distribution input data to the designated memory banks.

Let M denote the number of the memory banks. M is set to be the minimum of the banks number keeping dataflow continuous. Therefore, the minimum of the banks number is designed to the maximum of the radix values, i.e.,

$$M = max(r_1, r_2, \ldots, r_t). \quad (2)$$

3.1 Data distribution to banks

Here we only consider one mixed radix butterfly unit though multiple butterfly units can further speed up FFT computation.

We assume an accumulator ACC. Suppose that $ACC = (C_{s-1}C_{s-2}\ldots C_2C_1C_0)|_{MR}$ as an expression, which consists of s digits. C_{s-1} is the most significant digit and C_0 is the least significant digit. The value of each digit for ACC will be analyzed. $Y = (X)|_{MR}$ means that Y is expressed by X in mixed radix form.

First, input data should be distributed to M memory banks. There are two conditions to satisfy for ACC:

(1) The least significant digit of ACC is equal to M;
(2) The range of the $(s-1)$ most significant digits of ACC in decimal is between 0 and $N/M - 1$.

Therefore, the expression of distributing data to banks is expressed as Eq. (3).

$$Bank(j) = (C_{s-1}C_{s-2}\ldots C_1 j), \quad j = 0, 1, \ldots, M-1. \quad (3)$$

The mapping, just as Fig. 1, illustrates the relationship between ACC, the bank number and the offset address (OA) in the designated bank.

After distribution, each bank contains N/M input data. Then each bank computes N/M-point FFT. The access address generation follows the strategy analyzed in [11]. According to the theory of FFT, the memory banks are independent and pipeline. For each bank, it is accessed in pipeline and one operand is

DOI: 10.1587/elex.13.20160646
Received June 29, 2016
Accepted July 22, 2016
Publicized August 5, 2016
Copyedited September 25, 2016

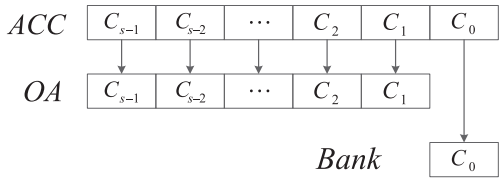

Fig. 1. Mapping relation between *ACC*, the bank number and the corresponding *OA*.

output in one clock cycle. For the last stage, i.e. radix-M computation, the relationship of banks is parallel. Therefore, the M operands is read out simultaneously for the butterfly inputs.

Finally, the computation time is discussed. In this multibank structure, the mixed radix FFT only needs one mixed radix butterfly unit, which can keep computation continuous. Therefore, the computation time of the butterfly unit itself is not considered. Because read and write of the memory bank are overlapped, the whole access time is not double of the read or write time. Here gives two common conditions to obtain access address clock cycles (T, including read and write) of N-point FFT.

Case 1: If $\mathrm{mod}(r_i, r_j) = 1$, $(i, j \in [1, t]\ and\ i \neq j)$, such as $N = 1536$ or 3780, then

$$T = \left(\sum_{i=1}^{t} N/r_i \times s_i\right) + 2 \times M - 1. \tag{4}$$

Case 2: If N is power of 2, like N equal to 1024 or 2048, then

$$T = N/M \times s + 2 \times M - 1. \tag{5}$$

In Case 2, the butterfly unit is configurable for any radix. For example, radix-8 butterfly can be set into two radix-4 or four radix-2 butterflies. Thus, this technique can shorten the computation cycles.

3.2 Multibank memory structure design

According to Eq. (2), M is 7. So 3780 data are distributed into 7 banks and there are 540 data for each bank.

According to Eq. (3), the data distributed in each bank and the corresponding *OA*s are listed in Table I.

Table I. Distribution of 3780 input data

OA	*Bank*(C_0)				
	0	1	j	5	6
0	$x(0)$	$x(1)$	$x(j)$	$x(5)$	$x(6)$
1	$x(7)$	$x(8)$	$x(7+j)$	$x(12)$	$x(13)$
k	$x(7k)$	$x(7k+1)$	$x(7k+j)$	$x(7k+5)$	$x(7k+6)$
539	$x(3773)$	$x(3774)$	$x(3773+j)$	$x(3778)$	$x(3779)$

The 7 sets of data are independent of each other for the 540-point FFT. When we compute the FFT, first of all, the pipeline structure is applied to implement the

DOI: 10.1587/elex.13.20160646
Received June 29, 2016
Accepted July 22, 2016
Publicized August 5, 2016
Copyedited September 25, 2016

540-point FFT computation, and then parallel structure is used for 7-point DFT computation.

4 Comparisons of different schemes

The major advantage of the proposed method is the running time. The running time is calculated by clock cycle. Table II lists the computation clock cycles of FFTs with different points.

Table II. Computation cycles comparisons

Points	[14]		[9]		Proposed	
	Cycle	Radix	Cycle	Radix	Cycle	Radix
1024	2560	4	1024	2/8	527	2/8
1536	–	–	2176	3/8	1103	3/8
2048	22528	2	2048	4/8	1039	4/8
3780	–	–	12040	3/4/5/7	6036	3/4/5/7
4096	4096	8	4096	8	2063	8

The results show that the computation cycles of the proposed method are almost half of the ones mentioned in [9], specially for the 1536- and 3780-point FFTs computation. If three butterfly units and 21 banks are used to compute the 3780-point FFT, the computation cycles are about 2120 and it is less 3780. Because the FFT algorithm is based on in-place strategy, no more memory is not required to keep the whole input data continuous.

Furthermore, a configurable butterfly unit for radix-3,4 and 5 is designed, just as Fig. 2. In Fig. 2, three selectors, i.e. C_{ti}, $(i = 0, 1, 2)$ and five multiplying factors, i.e. C_{5k}, $(k = 0, 1, 2, 3, 4)$ are designed. By configuring theses factors and input signals, just as Table III and Table IV, three butterflies, i.e. radix-3, 4 and 5, can be obtained. In Table III, $j = \sqrt{-1}$. Therefore, this method can simplify the hardware resources caused by multiple butterfly units.

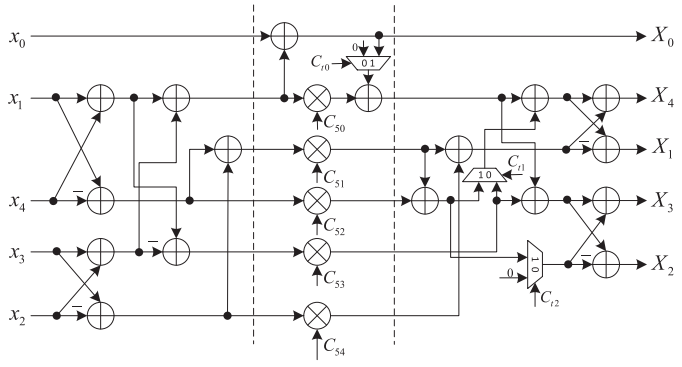

Fig. 2. Configurable butterfly design.

DOI: 10.1587/elex.13.20160646
Received June 29, 2016
Accepted July 22, 2016
Publicized August 5, 2016
Copyedited September 25, 2016

Table III. Configurable butterfly unit

Radix	C_{t0}	C_{t1}	C_{t2}	C_{50}	C_{51}	C_{52}	C_{53}	C_{54}
R-3	1	–	1	−1.5000	0	$-j0.8660$	0	0
R-4	0	1	0	0	0	1	−1	j
R-5	1	0	1	−1.2500	$j0.9511$	$-j0.3633$	0.5590	−1.5388

Table IV. Configuration for input and output data

Input data	R-3	R-4	R-5	Output data	R-3	R-4	R-5
x_0	x_0	0	x_0	X_0	X_0	X_0	X_0
x_1	x_1	x_0	x_1	X_1	–	X_3	X_1
x_2	0	x_1	x_2	X_2	X_2	–	X_2
x_3	0	x_3	x_3	X_3	X_1	X_2	X_3
x_4	x_2	x_2	x_4	X_4	–	X_1	X_4

5 Conclusion

The aim of the discussed method in this paper is the implementation in hardware platform, especially in FPGA. The theme of the letter is about the multi-bank structure of the mixed radix FFT. This strategy requires neither extra memories nor the data exchanges to avoid access conflicts, and it simplifies the structure complexity of the FFT parallel computations. This structure supports multi-butterfly units which includes multiple mixed radix butterflies. If we use multi-butterfly units, the number of the mixed radix butterflies is r_k ($k \in [1, s]$) times of M. Take 3780-point FFT as an example. The minimum of memory banks is $M = 7$ which support one mixed radix butterfly. If two butterfly units are used, the radixes are changed and radix-2 DFT arises which is undesirable. If the number of radix butterfly units is three, radix-3 and radix-7 DFTs are computed for the last two stages.

DOI: 10.1587/elex.13.20160646
Received June 29, 2016
Accepted July 22, 2016
Publicized August 5, 2016
Copyedited September 25, 2016