An efficient design for general mixed radix FFT processors

Cuimei Ma^{1a)}, He Chen², Yijian Liu³, and Yanfei Wang¹

¹ *Institute of Electronics, Chinese Academy of Sciences,
19 North Sihuan Road, Haidian, Beijing 100190, China*

² *Beijing Institute of Technology,
No.5 South Zhongguancun Street, Haidian, Beijing 100081, China*

³ *Hefei University of Technology, Tunxi Road No.193 Hefei, Anhui, 230009, China*
a) macuimei@126.com

Abstract: A general mixed-radix FFT design for in-place strategy is derived and a low-complexity scheme for efficiently implementing mixed-radix FFTs is proposed. In this method, we develop an accumulator that can simply and practically generate addresses for the operands, as well as the twiddle factors. This approach extends the range of FFT size and reduces the hardware complexity of any non-power-of-two memory-based FFTs. Finally, the 3780-point FFT is taken an example to illustrate the validation of the proposed method.

Keywords: generalized mixed-radix FFT, in place, access address, low complexity

Classification: Integrated circuits

References

- [1] R. C. Agarwal and J. W. Cooley: Proc. IEEE **75** (1987) 1283. DOI:10.1109/PROC.1987.13880
- [2] X. J. Li, Z. S. Lai and J. M. Cui: IEEE Trans. Consum. Electron. **53** (2007) 274. DOI:10.1109/TCE.2007.381685
- [3] Z. B. Zhang and X. W. Li: Appl. Integr. Circuits **36** [4] (2010) 54.
- [4] D. F. Liu, T. M. Lei, Z. Ma, Y. Li and Y. Wang: J. Xidian Univ. **37** (2010) 813. DOI:10.3969/j.issn.1001-2400.2010.05.007
- [5] S. S. Deng, Y. Sun, L. S. Zhang, Z. F. Mo and Y. K. Xie: J. Comput. Res. Dev. **45** (2008) 1430.
- [6] A. T. Jacobson, D. N. Truong and B. M. Baas: IEEE Int. Sym. Circuit Syst. (2009) 1133. DOI:10.1109/ISCAS.2009.5117960
- [7] G. L. Demuth: IEEE Trans. Acoustics Speech Signal Process. **37** (1989) 1349. DOI:10.1109/29.31290
- [8] B. G. Jo and M. H. Sunwoo: IEEE Trans. Circuits Syst. I, Reg. Papers **52** (2005) 911. DOI:10.1109/TCSI.2005.846667
- [9] C. F. Hsiao, Y. Chen and C. Y. Lee: IEEE Trans. Circuits Syst. II, Exp. Briefs **57** (2010) 26. DOI:10.1109/TCSII.2009.2037262
- [10] H. Sorokin and J. Takala: IEEE Conf. Acoustics, Speech, Signal Prog (ICASSP) (2011) 1709. DOI:10.1109/ICASSP.2011.5946830
- [11] C. Yang, Y. Z. Xie, H. Chen and Y. Deng: IEEE International System-on-Chip

- Conference (SOCC) (2014) 299. DOI:10.1109/SOCC.2014.6948944
- [12] K. J. Yang, S. H. Tsai and G. C. H. Chuang: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **21** (2013) 720. DOI:10.1109/TVLSI.2012.2194315
- [13] C. M. Ma, H. Chen, J. Y. Yu and T. Long: Sci. China Inf. Sci. **56** [4] (2013) 1. DOI:10.1007/s11432-013-4826-5
- [14] L. G. Johnson: 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 important to digital signal processing applications that one have available a general mixed radix FFT program which is, at the same time, operationally efficient [1]. Since the radix-2 FFT algorithm can be made to yield a very efficient and simple program, it was, for a long time, considered desirable to modify one's problem to fit the program and use the transform length N equal to a power of 2, which restricts its application.

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].

For different applications in various FFT processors, two key architectures, pipeline and memory based [11, 12, 13, 14], have been proposed. Pipeline architecture offers speed advantages but it requires more hardware resources at the same time. On the other hand, although memory-based architecture is slow, it requires least amount of hardware resources and it is often implemented with in-place strategy, overwriting the previous block of data [14]. For the generalized mixed radix FFTs design, it is more suitable to employ memory-based architecture especially for long-size DFT applications. Here, the in-place approach is adopted to design mixed radix FFTs.

However, the in-place strategy has a complex design circuit. Traditional methods usually require complicated circuits, such as [7, 9]. In [9, 10], the data access for twiddle factors is not discussed. The scheme described here can be applied to general mixed-radix FFTs, i.e. radix- $r_1/r_2/\cdots/r_t$ FFT, where r_i is some radix value. By this method, an arbitrary FFT size is chosen and a low-complexity address generation is obtained.

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. The proposed method of address generation is presented in Section 3. Section 4 shows the 3780-point FFT design as an illustrative example. 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:

$$\begin{cases} n = (N_2 n_1 + A_2 n_2) \bmod N, & n_1, k_1 = 0, 1, \dots, N_1 - 1 \\ k = (B_1 k_1 + N_1 k_2) \bmod N, & n_2, k_2 = 0, 1, \dots, N_2 - 1. \end{cases} \quad (1)$$

Eq. (1) shows the relationship between the indexes n and k and the index vectors (n_1, n_2) and (k_1, k_2) , transforming one dimension $[0, N - 1]$ into two dimensions $[0, N_1 - 1] \times [0, N_2 - 1]$. The selection of coefficients A_2 and B_1 depends on the relation between N_1 and N_2 [9]. If N_1 and N_2 are relatively prime, the prime factor FFT can be employed. Though it can eliminate the requirement for twiddle factor multiplication, the prime factor FFT makes access address generation complex. The indexes n and k can be obtain by some modulo operations. When the number of points becomes larger, they require more modulo operations to implement address generation circuits which must calculate the correct address for each input/output of each butterfly in each column. If N_1 and N_2 are not relatively prime, let $A_2 = B_1 = 1$ and $(N_2 n_1 + n_2)$ and $(k_1 + N_1 k_2)$ are less than N . Thus no modulo computation is required.

In the following proposed design, regardless of the relationship between the factors, the mixed radix FFT algorithm is expressed by an iterative equation based on Cooley-Tukey algorithm. The expression can be applied to either decimation-in-time or decimation-in-frequency FFTs with ordered inputs. The proposed method requires less modulo operations than the existing methods and can be applied to implement the general mixed radix FFTs efficiently.

3 Proposed method for address generation

Let $N = \prod_{i=1}^t r_i^{s_i}$ (where r_i and s_i mean a radix value and the corresponding integer power for the i^{th} stage FFT computation.) denote the number of points of a general mixed radix FFTs.

We assume the computation order $r_1, r_2, r_3, \dots, r_t$ and define a parameter $s = \sum_{i=1}^t s_i$ as the total of the FFT stages.

At first an accumulator (ACC) is assumed and it can be mapped into addresses of operands as well as the corresponding twiddle factors.

3.1 ACC design

Suppose that $ACC = (C_{s-1} C_{s-2} \dots C_2 C_1 C_0)_{MR}$ 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.

If the architecture of address mapping is consistent for every stage, the hardware complexity is low and we only design ACC to get the access address by shifting the digit's order of ACC . Therefore, ACC is designed for every FFT stage just as follows.

$$\begin{aligned} &\text{When } k \in [1, s_1], \text{ and } r_k = r_1: \\ &ACC(k) = (\underbrace{r_1 \cdot \dots \cdot r_1}_{s_1-1} \cdot \underbrace{r_t \cdot \dots \cdot r_t}_{s_t} r_1)_{MR} \end{aligned}$$

When $k \in [\sum_{m=1}^{j-1} s_m + 1, \sum_{m=1}^j s_m]$, ($2 \leq j \leq t-1$) and $r_k = r_j$:

$$ACC(k) = (\underbrace{r_1 \cdots r_1}_{s_1} \cdots \underbrace{r_j \cdots r_j}_{s_{j-1}} \cdots \underbrace{r_t \cdots r_t}_{s_t} r_j)_{MR}$$

When $k \in [\sum_{m=1}^{t-1} s_m + 1, s]$ and $r_k = r_t$:

$$ACC(k) = (\underbrace{r_1 \cdots r_1}_{s_1} \cdots \underbrace{r_t \cdots r_t}_{s_t})_{MR}$$

We get that for the current stage k , the least significant digit C_0 is just equal to the current-stage radix and the order of the $(s-1)$ most significant digits just follow the assumed order exclusive of the current-stage radix.

3.2 Address mapping for operands

Access addresses for operands can be mapped directly by ACC and the relationship between ACC and $Addr_{op}$ is $Addr_{op}(k) = (\underbrace{r_1 \cdots r_1}_{s_1} \underbrace{r_2 \cdots r_2}_{s_2} \cdots \underbrace{r_t \cdots r_t}_{s_t})_{MR}$, $k \in [1, s]$

and illustrated as Fig. 1. It shows the same structure for every stage, i.e. it keeps the mapping structure consistent. Because the access address is expressed in mixed radix, decimal representation can be obtained with number system conversion. Because ACC is generated with clock cycle, the current-stage r_k operands are obtained in r_k consecutive clock cycles.

Fig. 1. Address generator of operands for by ACC .

3.3 Address mapping for twiddle factors

Suppose two parameters $ACC_{reverse}$ and β which are used to compute the twiddle factor's address.

$ACC_{reverse}$ is just the reverse-order expression of ACC .

β (β consists of $(s-1)$ digits) can be obtained by $ACC_{reverse}$.

For stage 1, because the required twiddle factors are '1', the exponents are '0'. So β is set to zeros.

For stage k ($k \geq 2$), there are three aspects to be considered:

- (1) The least $(k-1)$ significant digits of $ACC_{reverse}$ are fetched as the most significant digits of β ;
- (2) The other $(s-k)$ digits of β are padded zeros;
- (3) The most digit of $ACC_{reverse}$, i.e. C_0 , is not considered. Hence, β is expressed for every stage shown as follows.

$$\text{When } k = 1: \beta(k) = (\underbrace{0 \cdots 0}_{s-1})_{MR};$$

$$\text{When } k \in [2, s_1]: \beta(k) = (\underbrace{r_1 \cdots r_1}_{k-1} \underbrace{0 \cdots 0}_{s-k})_{MR};$$

$$\text{When } k \in [\sum_{m=1}^{j-1} s_m + 1, \sum_{m=1}^j s_m]: \beta(k) = (\underbrace{r_j \cdots r_j}_{k-(s_1+\cdots+s_{j-1})-1} \cdots \underbrace{r_1 \cdots r_1}_{s_1} \underbrace{0 \cdots 0}_{s-k})|_{MR}.$$

According to β , access addresses of twiddle factors for radix- r_k ($1 \leq k \leq t$) butterfly computation can be obtained by $\text{Addr}_{twi}(k) = (n_k - 1) \times \beta(k)$, ($n_k \in [0, r_k - 1]$). Because the addresses of twiddle factors are not correlative with C_0 , when the r_k operands are obtained in r_k consecutive clock cycles, the r_k twiddle factors fetched in one clock cycle.

Finally, the relationship between the input and output orders is discussed and it is expressed as follows:

$$\begin{aligned} \text{Input order: } & (\underbrace{r_1 \cdots r_1}_{s_1} \underbrace{r_2 \cdots r_2}_{s_2} \cdots \underbrace{r_t \cdots r_t}_{s_t})|_{MR}, \\ \text{output order: } & (\underbrace{r_t \cdots r_t}_{s_t} \underbrace{r_3 \cdots r_3}_{s_3} \underbrace{r_2 \cdots r_2}_{s_2} \underbrace{r_1 \cdots r_1}_{s_1})|_{MR}. \end{aligned}$$

If $N = 2 \times 4^s$, the relationship between input order and output order is simplified to the one in [8].

According to the above discussion, the key point of the proposed scheme is to design *ACC*. It can be directly mapped to access addresses of the operands as well as the twiddle factors. Modulo operation is required only in designing *ACC* but not in mapping to generate access addresses.

Since 3780-point FFT is an important computation in DMB-T modulation system, we take it as an example to illustrate. 3780 can be decomposed into $4 \times 3 \times 3 \times 3 \times 5 \times 7$, i.e. $r_1 = 4$, $r_2 = 3$, $r_3 = 5$, $r_4 = 7$, $s_1 = s_3 = s_4 = 1$ and $r_2 = 3$. Therefore, the design for each stage is shown in Table I.

Table I. *ACC*, Addr_{op} and β design for 3780-Point FFT

Stage (k)	r_k	<i>ACC</i>	Addr_{op}	β
1	4	(333574)	(433357)	(00000)
2	3	(433573)	(433357)	(40000)
3	3	(433573)	(433357)	(34000)
4	3	(433573)	(433357)	(33400)
5	5	(433375)	(433357)	(33340)
6	7	(433357)	(433357)	(53334)

Take stage 4 as an example and the generation of each parameter is shown in Fig. 2.

n denotes the address of output of stage 3 in the memory which ranges from 0 to 3779. n'_1 and n'' mean the decimal representations of Addr_{op} and β , respectively. n'_2 denotes the write address that is the same as the read one due to the in-place strategy.

n	ACC (433573)	$Addr_{op}$ (433357)	$ACC_{reverse}$ (375334)	β (33400)	n''	n_k ($n_k \times n''$)	$Addr_{rel}$ n'_1	radix-3 computation	n'_2
0	000000	000000	000000	00000	0	0	0	$\times$	$\times 0$
1	000001	000100	100000	00000	0	1	0	$35 \times$	$\times 35$
2	000002	000200	200000	00000	0	2	0	$70 \times$	$\times 70$
3	000010	000001	010000	00000	0	0	0	$1 \times$	$\times 1$
4	000011	000101	110000	00000	0	1	0	$36 \times$	$\times 36$
5	000012	000201	210000	00000	0	2	0	$71 \times$	$\times 71$
...	...	...	...	...	...	...	...	...	...
21	000100	000010	001000	00000	0	0	0	$7 \times$	$\times 7$
22	000101	000110	101000	00000	0	1	0	$42 \times$	$\times 42$
23	000102	000210	201000	00000	0	2	0	$77 \times$	$\times 77$
...	...	...	...	...	...	...	...	...	...
103	012060	012006	060210	21000	780	0	0	$531 \times$	$\times 531$
104	012061	012106	160210	21000	780	1	780	$566 \times$	$\times 566$
105	012062	012206	260210	21000	780	2	1560	$601 \times$	$\times 601$
...	...	...	...	...	...	...	...	...	...
3777	322460	322046	064223	22300	1225	0	0	$3709 \times$	$\times 3709$
3778	322461	322146	164223	22300	1225	1	1225	$3744 \times$	$\times 3744$
3779	322462	322246	264223	22300	1225	2	2450	$3779 \times$	$\times 3779$

Fig. 2. Address generation for stage 4 of 3780-point FFT.

4 Comparisons and analysis

Because twiddle factor is not considered in [9] and the intermediate values in [7] are hard to be implemented in hardware, there is no comparison about the address generation for the twiddle factors.

A comparison on the address generation for the operands is given. 60-point radix-2/3/5 FFT is taken as an example.

Fig. 3. Comparison of address generation.

Fig. 3(a) shows the proposed scheme and Fig. 3(b) shows the scheme in [9]. There are mainly two outstanding aspects of the proposed method: (a) Consistent architecture of FFT for every stage is required and only one architecture is designed to get the address address. For the method in [9], the architecture of address generation for each stage is variable. We have to design different architectures for every stage to obtain the corresponding access address. (b) No complex modulo

operations is demanded. Hence, the larger the FFT size is, the more resources and modulo operations are needed in [9].

Table II lists the number of mathematical operations. Therefore, the proposed scheme simplifies the complexity of generating addresses.

Table II. Mathematical operations comparisons

Scheme	Proposed method	the method [9]
2-input addition	2	6
2-input multiplication	2	9
Modulo operation	0	5

5 Conclusion

The aim of the discussed method in this paper is the implementation in hardware platform, especially in FPGA. The mixed radix FFT also can be applied to the field of the remote sensing, such as synthetic aperture radar. For the high performance and fast computation, FPGA instead of DSP is usually used to implement the part of the relative algorithms, such as the digital pulse compression. In this proposed method, the low-complexity design for the general mixed radix FFTs. For FFT processors, access data is the key point. The strategy in this paper makes the generation of access address simple and easy. As regards the hardware complexity, the address generation algorithm can reduce the hardware complexity and enhance the implementation efficiency at the same time.