Multiplierless fast Fourier transform architecture

M. Jiang, B. Yang, R. Huang, T. Y. Zhang and Y. Y. Wang

A multiplierless processor architecture is proposed for hardware implementation of fast Fourier transform. Distributed arithmetic is applied to simplify expensive butterfly operations and twiddle multiplications. The novel architecture can largely reduce area cost by replacing complex multipliers and adders with DA lookup tables. Both 8-bit and 16-bit 64-point FFT processors were designed, and the synthesis result shows the designs can attain much lower area cost while keeping real-time processing speed.

Introduction: Discrete Fourier transform (DFT) is intrinsically a cyclic convolution operation that is widely applied in OFDM-based communications, image processing, and other signal processing applications. The Cooley-Turkey algorithm [1] is a very popular fast Fourier transform algorithm which can reduce the computational complexity from $O(N^2)$ to $O(N\log_2 N)$. Based on the Cooley-Turkey FFT algorithm, radix-2 FFT, radix-4 FFT and radix-8 FFT have been proposed in many reports [2–6]. Although radix-8 FFT can have less twiddle multiplications, its bottleneck is the implementation of its basic 8-point FFT, which has a lot of butterfly operations as well as non-trivial complex multiplications with coefficients such as $\pm\sqrt{(2)}/2$. Therefore, radix-8 FFT has similar computation complexity as radix-4 [5]. Such kind of radix-8 FFT is also called radix-2^3 FFT because the radix-2 butterfly operation is still its basic operation. A number of algorithms have been applied to perform butterfly operations or complex multiplications, such as canonical signed digit (CSD) arithmetic [2], the CORDIC algorithm [3], and distributed arithmetic [4, 5]. DA-based architectures usually have lower area cost and lower power consumption compared with other methodologies. Distributed arithmetic can save butterfly operations by its LUT indexing. However, radix-4 FFT will have 8-input 256-word DA LUTs, and radix-8 FFT needs 16-input 256×256-word LUTs. Such an expensive hardware cost of ROMs and index addressing logic will prohibit distributed arithmetic from practical applications. Until now, distributed arithmetic was only used with radix-2 FFT architecture and radix-4 FFT architecture [4, 5].

In this Letter, a novel radix-8 FFT architecture is proposed for the FFT processor design. In the new architecture, the size of 16-input DA LUTs are reduced to only 64 words, and all complex multiplications and butterfly operations are implemented by DA lookup tables.

Cooley-Turkey fast Fourier transform: N^2-point DFT can be divided into two N-point FFTs according to the Cooley-Turkey FFT algorithm, which can be expressed as

$$\begin{aligned} F(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n_2=0}^{N-1} W_N^{n_2 k_1} \underline{\left\{ W_{N^2}^{n_2 k_2} \sum_{n_1=0}^{N-1} x(n_1 N + n_2) W_N^{n_1 k_2} \right\}}, \\ &= \sum_{n_2=0}^{N-1} W_N^{n_2 k_1} \hat{x}(n_2, k_2) \end{aligned} \tag{1}$$

where

$$n = n_1 N + n_2, \quad \text{and} \quad k = k_1 N + k_2 \tag{2}$$

Thus, an N^2-point FFT can be divided into two N-point FFTs. Accordingly, an N^r-point DFT can be divided into r-stage N-point FFTs. The decomposition with $N=2$ is called radix-2 FFT, and the decomposition with $N=8$ is called radix-8 FFT. For example, a 64-point FFT can be decomposed into two 8-point FFTs, shown as follows:

$$\begin{aligned} F(k) &= \sum_{n=0}^{63} x(n) W_{64}^{nk} \\ &= \sum_{n_2=0}^{7} W_8^{n_2 k_1} \left[W_{64}^{n_2 k_2} \sum_{n_1=0}^{7} x(8n_1 + n_2) W_8^{n_1 k_2} \right] \end{aligned} \tag{3}$$

Here, $n = n_1 N + n_2$, and $k = k_1 N + k_2$.

Proposed DA-based radix-8 FFT architecture: 8-point FFT is the basis of radix-8 FFT, which can be expressed as

$$\begin{aligned} \alpha(k) &= \sum_{m=0}^{7} x(m) W_8^{mk} \\ &= \left\{ \sum_{m=0}^{7} [x_{RE}(m)\cos(mk\pi/4) + x_{IM}(m)\sin(mk\pi/4)] \right. \\ &\quad \left. + j \sum_{m=0}^{7} [x_{RE}(m)\sin(mk\pi/4) - x_{IM}(m)\cos(mk\pi/4)] \right\} \end{aligned} \tag{4}$$

where, x_{RE} is the real part and x_{IM} is the imaginative part of $x(m)$. Because

$$\sin(\pi + \phi) = -\sin(\phi), \text{ and } \cos(\pi + \phi) = -\cos(\phi) \tag{5}$$

The above formula can be folded to 8-tap FIR format as

$$\begin{aligned} \alpha(k)_{RE} &= \sum_{m=0}^{3} [(x_{RE}(m) - x_{RE}(4+m))\cos(mk\pi/4) \\ &\quad + (x_{IM}(m) - x_{IM}(4+m))\sin(mk\pi/4)] \\ \alpha(k)_{IM} &= j \sum_{m=0}^{3} [(x_{RE}(m) - x_{RE}(4+m))\sin(mk\pi/4) \\ &\quad - (x_{IM}(m) - x_{IM}(4+m))\cos(mk\pi/4)] \end{aligned} \tag{6}$$

While k is odd number, we have $\cos(mk\ \pi/4) = 0$ while $m = 2$, and $\sin(mk\ \pi/4) = 0$ while $m = 0$. While k is even number, there are $\cos(mk\ \pi/4) = 0$ while m is odd number, and $\sin(mk\ \pi/4) = 0$ while m is even number. Therefore, the actual number of taps is reduced to six when k is odd number, and four when k is even number. By using the DA algorithm, the 8-point FFT computation can be done simply by 2^6-word or 2^4-word lookup tables.

Tap-merging of $(x_{RE}(m) - x_{RE}(4+\mathrm{m}))$ can be done by a sign exchange method

$$(x_{RE}(m) - x_{RE}(4+m)) = \left(x_{RE}(m) + \overline{x_{RE}(4+m)} + 1\right) \tag{7}$$

The above binary addition can be performed in bit-serial by a 1-bit 3:2 carry-save adder in the bit-serial DA architecture.

The 64-point FFT processor can be constructed with the basic 8-point FFT module, as shown in Fig. 1. The 64-point FFT is decomposed into two 8-point FFTs, and seven complex multiplications with twiddle factors W^{n2k2} need to be performed by twiddle multipliers. A transposed RAM is employed to store the intermediate results after the first 8-point FFT.

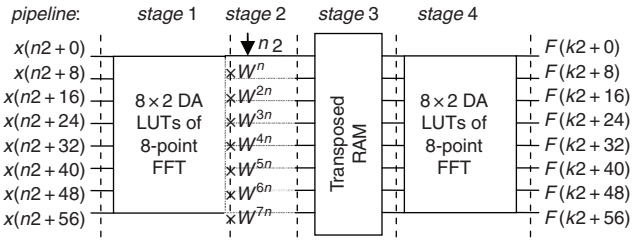

Fig. 1 *Proposed Radix-8 64-point FFT architecture*

The DA module can process eight m-bit input data in m cycles. When $m = 8$ bits, the DA-based 8-point FFT can process eight points in a real-time mode, i.e. one point per cycle. For a 16-bit FFT, we can use two 8-point FFT modules in the 8-point FFT stages to attain the real-time requirement.

DA-based twiddle multiplier: The twiddle multiplier is a bottleneck for FFT processor design. From Fig. 1 it is clear that $\alpha(k_2)$ from the first 8-point FFT will multiply with eight different twiddle factors W^{n2k2} to cover from $n_2 = 0$ to $n_2 = 7$. Usually there are three conventional ways to tackle this complex multiplication: Booth-Wallace multiplier, CORDIC multiplier, and CSD multiplier. CORDIC multiplier and Booth multiplier usually have large area cost, and it is also not easy for CSD arithmetic to tackle with constant twiddle factors of W^{nk}. Reference [2] supplied a shuffled CSD-based structure which puts all nine independent constants together. However, nine

CSD multipliers with a logic-complex shuffling multiplexer will apparently have more expensive area cost than one conventional array multiplier.

The complex multiplication of the twiddle factor W^{n2k2} is actually a 2-tap FIR, which are

$$X \times W = (X_{re} \times W_{re} - X_{im} \times W_{im}) + j(X_{re} \times W_{im} + X_{im} \times W_{re}) \quad (8)$$

According to distributed arithmetic, the above complex twiddle multiplication can be implemented by two 2^2-word lookup tables, as shown in Fig. 2. Each nonzero (n, $k \neq 0$) twiddle factor W^{nk} needs such a DA-based twiddle multiplier. In total, there are $7 \times 7 = 49$ DA-based twiddle multipliers in the 64-point FFT processor.

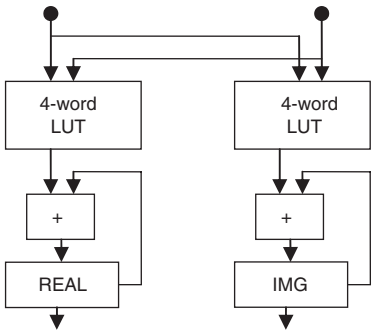

Fig. 2 *DA-based twiddle multiplier*

Results and discussion: Both 16-bit and 8-bit 64-point FFT processors were designed in VHDL and synthesised by Synopsis toolkits. All constant-based DA LUTs are implemented directly by logic gate circuits, rather than ROMs and addressing circuits [4, 5]. The result shows that the area cost of the architectures is much lower than existing designs, as listed in Table 1.

Table 1: Comparison of synthesis results

64-point FFT architectures		Gate count	Cycles per FFT
[6]	16-bit	~280 k	222
[2]	16-bit	~60 k	64
Ours	8-bit	8615	64
	16-bit	26 330	64

Because the processor speed depends on pipeline stages and implementation technology, the comparison of cycles per FFT should be more helpful to evaluate various FFT architectures than the running frequency. The processor can finish every 64-point FFT computation in 64 cycles, which ensures real-time processing of the input data. This can save a lot of buffer memory to store the input data. Another outstanding advantage of the proposed architecture is its multiplierless implementation by distributed arithmetic, which can save the power-hungry complex multipliers.

Conclusion: The proposed multiplierless DA-based radix-8 FFT algorithm can greatly save area cost without degrading processing speed. 64-point FFT processors were constructed with the proposed algorithm, and the synthesis shows it can have much lower area cost while keeping real-time processing ability.

Acknowledgments: The authors acknowledge the support of and governmental funding by the National Key Research Funding Project and the Advanced Technology Project of the Ministry of Science and Technology of China. The proposed FFT architecture in this Letter is under a patent protection held by Peking University.

14 October 2006
Electronics Letters online no: 20073049
doi: 10.1049/el:20073049

M. Jiang, B. Yang, R. Huang, T.Y. Zhang and Y.Y. Wang (*Institute of Microelectronics, Peking University, Beijing, People's Republic of China*)

E-mail: jiang@ime.pku.edu.cn

References

1 Cooley, J.W., and Turkey, J.: 'An algorithm for the machine calculation of complex Fourier series', *Math. Comput.*, 1965, **19**, pp. 297–301
2 Maharatna, K., Grass, E., and Jagdhold, U.: 'A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM', *IEEE J. Solid-State Circuits*, 2004, **39**, (3), p. 484
3 Hui, W., Ding, T.J., and McCanny, V.: 'A 64-point Fourier transform chip for video motion compensation using phase correlation', *IEEE J. Solid-State Circuits*, 1996, **31**, pp. 1751–1761
4 Shaditalab, M., Bois, G., and Sawan, M.: 'Self-sorting radix-2 FFT on FPGAs using parallel pipelined distributed arithmetic blocks'. IEEE Symp. on FPGAs for Custom Computing Machines, 1998, p. 137
5 Perez-Pascual, A., Sanaloni, T., and Valls, J.: 'FPGA-based radix-4 butterflies for HIPERLAN/2'. IEEE Int. Symp. on Circuits and Systems, Scottsdale, AZ, USA, 2002, p. 277
6 Chen, T., Sunanda, T.G., and Jin, J.: 'COBRA: a 100-MOPS single-chip programmable and expandable FFT', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1999, **7**, pp. 174–182