现代雷达中的高速FFT设计

吴　伟1，　唐　斌1，　杜东平1，　杨宝强2

(1. 电子科技大学电子工程学院，四川成都 610054；2. 空军工程大学训练部，陕西西安 710051)

摘　要：针对FFT专用处理器无法满足现代雷达高速实时信号处理的要求，提出了四种高速FFT的设计方案。方案在分析比较各种FFT算法的基础上，兼顾速度、资源和复杂度三个方面，选用基4算法，利用CORDIC算法产生旋转因子，点数和字长均可灵活配置，工程可实现性强。设计方案的性能分析和硬件实现验证了设计方案的有效性，适应现代雷达的不同处理要求。

关键词：FFT. 基4算法；CORDIC算法；旋转因子

中图分类号：TN957　　文献标识码：A　　文章编号：1009-3516(2005)05-0053-03

FFT作为数字谱分析的必要手段，广泛应用于雷达、通信、声纳和图像处理等众多领域。目前，FFT专用处理器有了较大的发展，常见的FFT专用模块可以达到的速度数量级普遍为1024点16位字长定点、块浮点、浮点运算在几十到数百μs量级[1]，但是其功能单一，设计固定，外围引脚较多且需要外部RAM配置，操作复杂，无法针对实际要求进行修改，性价比较低。在现代雷达信号处理中，FFT面临的数据率从几kHz到数百MHz，处理的实时性要求高，处理速度快。面对如此复杂的高速信号处理环境，FFT专用处理器就无能为力了。在各种FFT算法中，Winograd算法和素因子算法[2]在运算量上占优，用的乘法器比Cooley－Tukey算法[3]和分裂基算法[4]等递归型算法少，但控制复杂，控制单元的实现相当麻烦。分裂基算法综合了基4和基2算法的运算特点，但其L型蝶形运算结构在控制上要复杂一些。基4与基2算法相比，节省约25%乘法次数，并且可以使存储器和运算部件间的必要通讯减少一半，并行性提高4倍，同样时钟频率下的运算速度是基2的2倍，并且可以显著的改善数值精度[5~6]。

1　高速FFT设计方案

1.1　采用CORDIC算法的设计方案

实现FFT运算的一个难点是旋转因子的产生，若是已经确定FFT运算的点数和字长，则可以预先产生旋转因子，再转换成二进制定点数存储在ROM中。当FFT的点数较大时，需要存储的旋转因子的数量将迅速增长，让人难以接受，同时，为使设计更具通用性，运算点数和字长也应具备可变性。所以，预先存储旋转因子的方法就无能为力了。利用CORDIC算法产生旋转因子的方法具有实现简单，点数和字长可配置的特点，采用算法的FFT设计方案见图1，方案中共同复用一个基4蝶形运算单元，耗费资源最少。

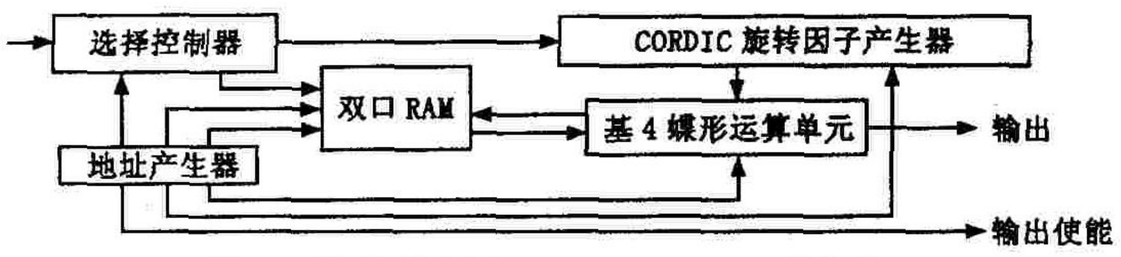

图1　采用CORDIC算法的FFT设计方案

设完成一次4输入4输出的基4蝶形运算的时间为T_b，则完成4^N点复数FFT运算需要进行N级基4

收稿日期：2005－01－13

作者简介：吴　伟(1979－)，男，四川资阳人，博士生，主要从事雷达数字接收技术研究；
唐　斌(1964－)，男，四川成都人，教授，博士生导师，主要从事雷达、通信与信息对抗等研究.

蝶形运算,每一级中包括 4^{N-1} 次基 4 蝶形运算。因此,在复用一个基 4 蝶形运算单元的情况下,采用 CORDIC 算法的 FFT 一次运算时间为 $T_{FFT1}=N\times4^{N-1}T_b+T_{S1}+T_{C1}$。式中:$T_{S1}$ 为该设计方案下运算处理数据在双口 RAM 中的存取时间;T_{C1} 为运算处理的控制时间,包括产生每一级每一次蝶形运算数据存取地址的时间。

输入数据速率匹配下,CORDIC 算法的 N 次连续 FFT 运算时间为 $T_{N\_FFT1}=N(N\times4^{N-1}T_b+T_{S1}+T_{C1})=M^2\times4^{N-1}T_b+NT_{S1}+NT_{C1}=N^2\times4^{N-1}T_b+T_{N-S1}+T_{N-C1}$。式中:$T_{N\_S1}=NT_{S1}$ 为 N 点 FFT 的总数据存取时间;$T_{N\_C1}=NT_{C1}$ 为总控制时间。

1.2 采用并行结构和 CORDIC 算法的设计方案

采用并行结构是提高 FFT 运算速度的一个重要技术途径,它利用资源来换取时间,使得每一级 4^{N-1} 次基 4 蝶形运算同时进行,仅耗费一次基 4 蝶形运算时间。采用并行结构和 CORDIC 算法 FFT 方案见图 2。

在 4^N 点复数 FFT 运算中,基 4 DIF 运算分为 N 级,每一级包括 4^{N-1} 个基 4 蝶形运算单元。因此,完成一次 FFT 的运算时间为 $T_{FFT2}=NT_b+T_{S2}+T_{C2}$。式中:$T_{S2}$ 为该设计方案下 N 级运算处理数据在存储器中的存取时间,T_{C2} 为运算处理的控制时间,包括产生每一级蝶形运算数据存取地址的时间。

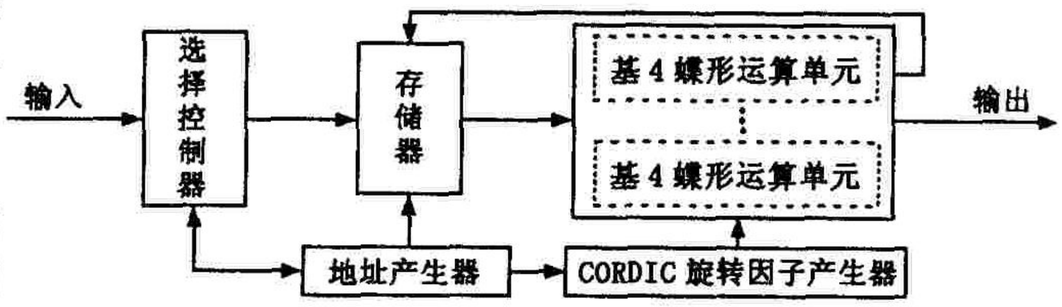

图 2 采用并行结构和 CORDIC 算法的 FFT 设计方案

输入数据速率匹配下,采用并行结构和 CORDIC 算法的 N 次连续 FFT 运算时间为 $T_{N\_FFT2}=N(N+T_{S2}+T_{C2})=M^2T_b+NT_{S2}+NT_{C2}=N^2T_b+T_{N-S2}+T_{N-C2}$。式中:$T_{N\_S2}=NT_{S2}$ 为 N 点 FFT 的总数据存取时间;$T_{N\_C2}=NT_{C2}$ 为总控制时间。

1.3 采用流水线结构和 CORDIC 算法的设计方案

采用流水线结构是提高 FFT 运算速度的另一个重要技术途径,保证 FFT 中每一级运算能够同时进行,当输入数据速率匹配时,总系统运算时间仅为一级流水线的时间。采用流水线结构和 CORDIC 算法的 FFT 设计方案见图 3,每一级运算复用一个基 4 蝶形运算单元。

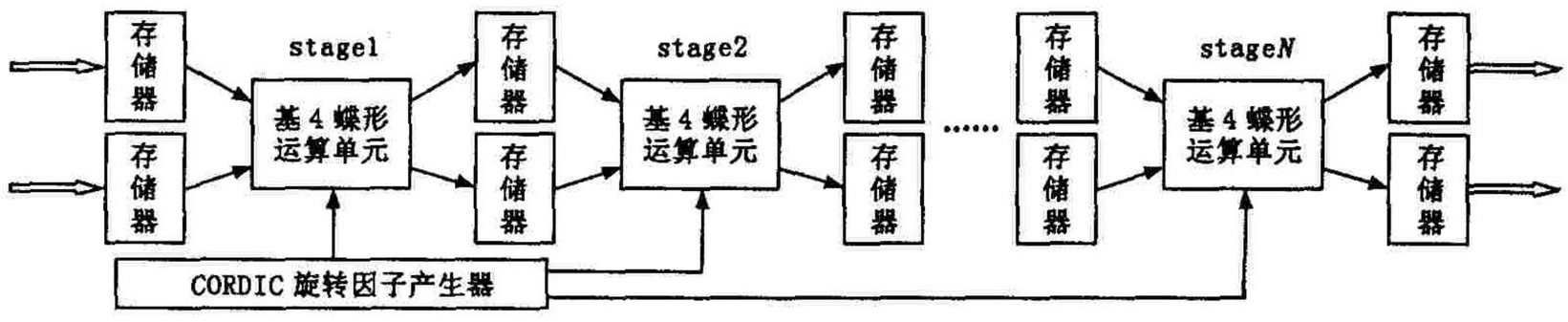

图 3 采用流水线结构和 CORDIC 算法的 FFT 设计方案

在 4^N 点复数 FFT 运算中,基 4 DIF 运算分为 N 级流水运算,共包括 N 个基 4 蝶形运算单元和 $N+1$ 组存储 RAM 单元。其中,每一级流水结构只包括一个基 4 蝶形运算单元,在运算控制器的控制下进行时分复用,每一级需要进行 4^{N-1} 次基 4 蝶形运算,因此,采用流水线结构和 CORDIC 算法设计方案的 FFT 一次运算时间为 $T_{FFT3}=N\times4^{N-1}T_b+T_{S3}+T_{C3}$。式中:$T_{S3}$ 为该设计方案下 N 级运算处理数据在存储器中的存取时间;T_{C3} 为运算处理的控制时间,包括产生每一级每一次蝶形运算数据存取地址的时间。

在输入数据速率匹配的情况下,采用流水线结构和 CORDIC 算法设计方案的 N 次连续 FFT 运算时间为 $T_{N\_FFT3}=(N\times4^{N-1}T_b+T_{S3}+T_{C3})+(N-1)\times4^{N-1}T_b+(N-1)/N(T_{S3}+T_{C3})=(2N-1)\times(4^{N-1}T_b+(2N-1))/N(T_{S3}+T_{C3})=(2N-1)\times(4^{N-1}T_b+T_{N\_S3}+T_{N\_C3})$。式中:$T_{N\_S3}=(2N-1)/NT_{S3}$ 为 N 点 FFT 的总数据存取时间;$T_{N\_C3}=(2N-1)/NT_{C3}$ 为总控制时间。

1.4 采用流水线和并行结构的设计方案

在资源足够的情况下,可以同时采用流水线结构和并行结构,见图 4,这样能够最大程度的提高 FFT 运算速度。当输入数据速率匹配时,总系统运算时间仅为一级流水线的时间,每一级有 4^{N-1} 个基 4 蝶形运算单元并行同时运算,时间仅为一个基 4 蝶形运算时间。

在 4^N 点复数 FFT 运算中,基 4 DIF 运算分为 N 级,N 级流水运算,$N+1$ 组存储 RAM 单元,每一级流水结构包括 4^{N-1} 个基 4 蝶形运算单元。因此,完成一次 FFT 的运算时间为 $T_{FFT4}=NT_b+T_{S4}+T_{C4}$。式中:$T_{S4}$ 为该设计方案下 N 级运算处理数据在存储器中的存取时间;T_{C4} 为运算处理的控制时间,包括产生每一级蝶

形运算数据存取地址的时间。

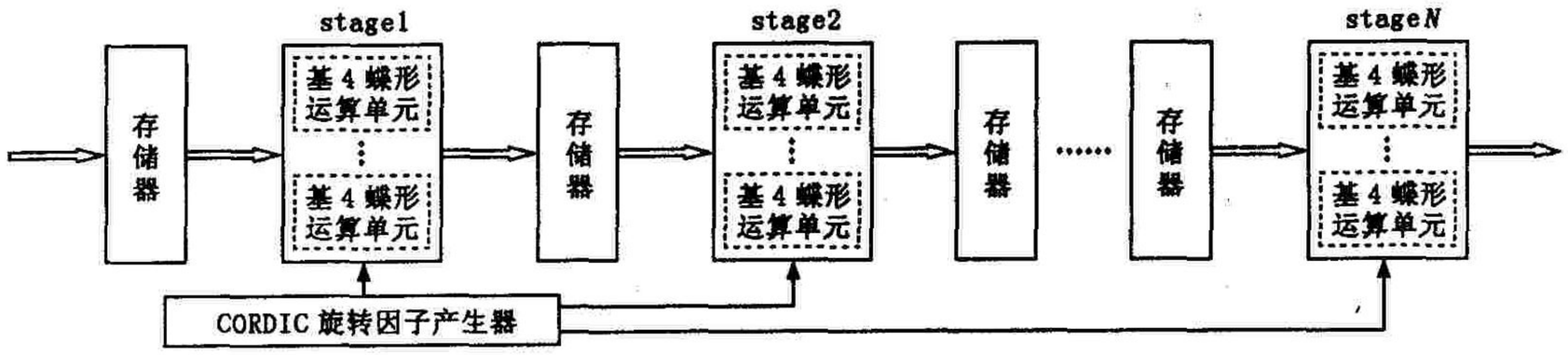

图 4　采用流水线和并行结构的 FFT 设计方案

在输入数据速率匹配的情况下,采用流水线和并行结构设计方案的 N 次连续 FFT 运算时间为 $T_{N\_FFT4} = (NT_b + T_{S4} + T_{C4}) + (N-1)T_b + (N-1)/N(T_{S4} + T_{C4}) = (2N-1)T_b + (2N-1))/N(T_{S4} + T_{C4}) = (2N-1)T_b + T_{N\_S4} + T_{N\_C4})$。式中:$T_{N_S4} = (2N-1)/NT_{S4}$为 N 点 FFT 的总数据存取时间,$T_{N_C4} = (2N-1)/NT_{C4}$为总控制时间。

2　性能分析

上述四种高速 FFT 设计方案在速度、资源和复杂度方面各具优势,分别适合不同情况下的处理要求。这些方案均可采用 VHDL 完成设计,并通过仿真、综合、布局和布线,分别在不同系统中满足各自的要求。

下面通过在速度、资源和复杂度等方面进行比较,表 1 给出这四种高速 FFT 设计方案的性能分析。

表 1　高速 FFT 设计方案性能比较

		CORDIC	并行 + CORDIC	流水线 + CORDIC	并行 + 流水线
时间	1 次	$N\times4^{N-1}T_b$	NT_b	$N\times4^{N-1}T_b$	NT_b
	N 次	$N^2\times4^{N-1}T_b$	N^2T_b	$(2N-1)\times4^{N-1}T_b$	$(2N-1)T_b$
资源		S	$4^{N-1}S$	NS	$N\times4^{N-1}S$
复杂度		4	2	3	1

表 1 中,忽略相对较短的数据存取时间和控制时间,以最耗资源的一个基 4 蝶形运算单元和一组存储器为一个资源计算单元,运算控制的复杂度以 1 为最简单,4 为最复杂。可以看出,采用 CORDIC 算法的设计方案耗费的资源最少,然而却是以牺牲速度为代价的,耗时最长,其运算控制也最复杂;采用流水线和并行结构的设计方案正好相反,无论哪种情况下其速度总是最快的,运算控制也最简单,然而却是以牺牲资源为代价,耗费的资源最多。另外两种方案则为二者的折中,兼顾了提高速度、节省资源和降低复杂度三个方面。

3　结论

上述分析表明,本文给出的四种高速 FFT 设计方案,在速度、资源和复杂度方面具有各自的优势,分别已经在不同的实际系统中得到实现验证。因此,根据不同的系统要求,选择恰当的高速 FFT 设计方案,借助可编程器件配置方便灵活的优势,可以设计出兼顾速度、资源和复杂度的高速 FFT 处理器。

参考文献:

[1]　韩泽耀, 韩雁, 郑为民. 一种高速实时定点 FFT 处理器的设计[J]. 电路与系统学报, 2002, 7(1): 18 - 22.

[2]　蒋增荣, 曾泳泓, 余品能. 快速算法[M]. 长沙: 国防科技大学出版社, 1993.

[3]　Duhamel P. Implementation of Split - Radix FFT Algorithms for Complex, Real and Real Symmetric Data[J]. IEEE Trans. ASSP, 1986, 34(2): 285 - 295.

[4]　胡广书. 数字信号处理:理论、算法与实现(第二版)[M]. 北京:清华大学出版社, 1997.

[5]　Chang Y N, Parhi K K. An Efficient pipelined FFT Architecture[J]. IEEE Transactions on Circuits and Systems II, 2003, 50(6): 322 - 325.

(下转第 58 页)