

基于 CORDIC 算法的高速可配置 FFT 的 FPGA 实现

文婧媛^{1,2}, 徐欣锋¹

(1 中国科学院 微电子研究所, 北京 100029; 2 南开大学 信息技术科学学院, 天津 300071)

摘 要: 论述了一种用于星载合成孔径雷达(SAR)星上数据实时自主处理系统中的高性能 FFT 的 FPGA 实现. 采用 CORDIC 算法实现复数乘法, 降低了系统的复杂性, 提高了运算速度, 并提出一种新型便捷的旋转因子产生方法, 无需额外的 ROM 资源. 采用块浮点的数据类型, 有效避免了大点数 FFT 的溢出问题. 运算点数可配置, 能够实现 64~32k 点, 实部、虚部均为 16bit 数据的 FFT 运算. 整体设计采用 16 点并行流水结构, 提出了适用于 16 通道并行读写的无冲突地址产生方法. 最高工作频率可达 118.89MHz, 100MHz 频率下, 1024 点 FFT 的计算时间仅为 4.48 μ s, 完全满足高速实时的运算要求.

关键词: CORDIC 算法; FFT; 旋转因子; 块浮点; 无冲突地址生成

中图分类号: TN911.7

文献标识码: A

文章编号: 1000-7180(2010)03-0024-05

A FPGA Realization of CORDIC Based High Speed Variable Point FFT

WEN Jing-yuan^{1,2}, XU Xin-feng¹

(1 Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029 China;

2 School of Information Technology and Science, Nankai University, Tianjin 300071, China)

Abstract: Present a FPGA realization of high performance FFT used in star-carried synthetic aperture radar real-time data processing system. The Coordinate Rotational Digital Computer (CORDIC) algorithm is adopted to realize complex multiplication, reducing the system complexity and saving hardware resource. A new convenient twiddle factor generator is proposed and don't need extra ROM resources. The block floating point data type effectively solves the overflow in long point FFT. The point can be varied from 64 to 32K and the real and imaginary parts of data are 16 bits. The whole design uses 16-point parallel pipeline structure. A 16-access parallel conflict free address generation is proposed. The highest clock frequency is 118.89MHz. The processor can compute a 1024-point FFT with 4.48 μ s, satisfies the requirement of high speed, real-time operation.

Key words: CORDIC algorithm; FFT; rotational factor; block floating point; conflict free address generation

1 引言

快速傅里叶变换(FFT)作为时域和频域转换的基本运算,已经成为数字谱分析的必要手段,广泛应用于雷达、通信、声纳和图像处理等领域.对于雷达信号处理系统,需要处理的数据量、运算量非常巨大,并要求达到准实时甚至全实时的成像处理速度,因此对 FFT 的性能也提出了较高的要求^[1].提高 FFT 运算速度的主要方法是提高工作频率和数据

的并行度.在 FFT 运算中,数据与旋转因子的复数乘法是制约工作频率的主要因素之一.文中采用 CORDIC 算法实现乘法运算,降低了系统设计的复杂性.整体设计采用 16 点并行流水结构,满足了高速实时的运算要求.

2 CORDIC 算法

CORDIC 算法(The Coordinate Rotational Digital Computer)从广义上讲是一种数值性计算逼近

收稿日期: 2009-03-13; 修回日期: 2009-05-16

基金项目: 中国科学院院长奖专项资金(07YY012001);中国科学院微电子研究所所长基金项目(06SB042002)

的方法.其基本思想是用一系列与运算基数相关的角度不断偏摆从而逼近所需旋转的角度.将一系列复杂的数学运算分解成一些简单的且在硬件中容易实现的基本算法,如加法、移位等^[2],从而降低设计的复杂性,节省硬件资源.这种算法是由 J. Volder 等人于 1959 年在美国航空控制系统的设计中首先提出的,1971 年 J. S. Walthers 将其推广到各种初等函数的统一计算中.可用其计算的函数包括乘、除、平方根、正弦、余弦、反正切、向量旋转(即复数乘法)以及指数运算等.

FFT 算法中序列与旋转因子 W_N^k 相乘为复数乘法,用乘法器直接相乘,需要进行四次实数乘法,两次加法,还需要一块 ROM 存放 $N/4$ 个旋转因子实部和虚部的值^[3],硬件开销很大,尤其当运算点数比较大时,所需硬件资源成倍增加,采用 CORDIC 算法则能够有效地解决这些问题^[4].

首先,设 $X = x(n) \cdot W_N^k$,将 $w_N^k = e^{-j\frac{2\pi}{N}nk}$, $x(n) = x_I + jx_Q$ 代入,得到

$$X = [\cos(-\frac{2nk\pi}{N}) + j\sin(-\frac{2nk\pi}{N})][x_I + jx_Q] \quad (1)$$

其实部和虚部分别为

$$\begin{aligned} X_I &= x_I \cos(-\frac{2nk\pi}{N}) - x_Q \sin(-\frac{2nk\pi}{N}) \\ X_Q &= x_Q \cos(-\frac{2nk\pi}{N}) + x_I \sin(-\frac{2nk\pi}{N}) \end{aligned} \quad (2)$$

因此,可以将复数序列和旋转因子相乘看作是将向量 $x(n)$ 旋转了 $\theta = -\frac{2nk\pi}{N}$ 度. CORDIC 算法的基本思想就是将 θ 分解成若干小角度 θ_n ,每次旋转 θ_n 度,同时将 θ 减去 θ_n ,再由 θ 的正负判断下一次旋转的方向,直到 θ 趋于 0,则可以认为向量在一定精度范围内旋转了 θ 度.向量旋转的迭代式可以写成^[5]:

$$\begin{aligned} x_{n+1} &= x_n \cos \theta_n - y_n \sin \theta_n \\ y_{n+1} &= y_n \cos \theta_n + x_n \sin \theta_n \end{aligned} \quad (3)$$

取合适的 θ_n 值,令 $\tan \theta_n = 2^{-n}$,即 $\theta_n = \arctan(2^{-n})$,整理得到

$$\begin{aligned} x_{n+1} &= \cos \theta_n (x_n - S_n y_n 2^{-n}) \\ y_{n+1} &= \cos \theta_n (y_n + S_n x_n 2^{-n}) \\ z_{n+1} &= z_n - S_n \arctan(2^{-n}) \end{aligned} \quad (4)$$

式中, $S_n = \pm 1$,为向量旋转的方向.可以看出每级旋转都是由输入向量的移位、加减运算完成的. Volder 已经证明这样对旋转进行迭代的方法是收敛

的.每一级迭代旋转中的因子 $\cos \theta_n$ 称为模校正因子,可以逐级提取出来最后根据需要决定是否乘上该因子,并且通过计算可以证明 $K = \prod_{n=0}^{\infty} \cos(\arctan 2^{-n}) = 0.607253$,因此该算法有一个约为 1.647 的增益,确切增益值则取决于迭代次数.

3 FFT 处理器的设计方案

3.1 FFT 处理器的基本结构

FFT 的整体结构如图 1 所示,为实现大点数、高速实时的 FFT 运算,这里采用 16 点并行的流水结构.内置两块双口 RAM,实现乒乓存储操作^[6].采用基 4-DIT 算法实现蝶形单元.

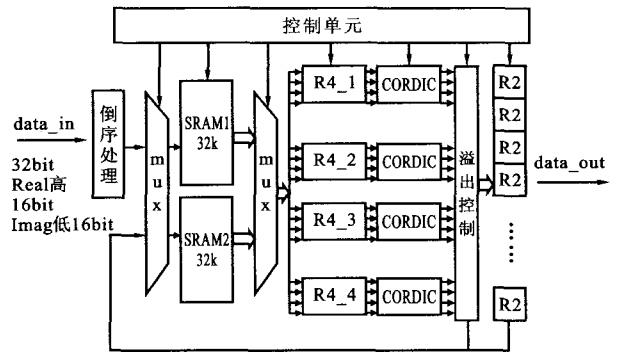


图 1 FFT 处理器整体结构图

当 SRAM1 中写满 N 个数据后,开始对 SRAM1 中的数据进行 FFT 运算,同时下一组 N 个数据继续逐一写入 SRAM2.由于采用 DIT 算法,所以在写入之前先进行倒序处理.运算部分由 4 个基 4 蝶形单元组成,能够同时计算 16 个数据.计算第一级时先输入第 1~16 个数据,每 4 个数据经过蝶形运算后进入 CORDIC 乘法器,由控制单元给出旋转角度.由于数据输入时已经完成倒序处理,计算结果原位返回 SRAM,即“原位计算”.在第二级运算中,将第 1、5、9、13 个数据输入第一个基 4 单元中,将第 2、6、10、14 个数据输入第二个基 4 单元中,以此类推.以后各级均类似实现,数据抽取由控制单元和地址产生模块给出.当点数 N 为 2^n 时,最后一级进行基 2 运算.全部运算完成后,依次将 SRAM1 中数据输出到外部接口.当 SRAM1 中数据输出结束后,SRAM2 中数据需要写入完毕,并开始对 SARM2 中数据进行 FFT 运算,同时将新一组 N 个数据继续写入 SRAM1,从而实现实时的数据处理过程,为此运算模块的时钟频率必须是输入数据采样频率的 2 倍以上.

3.2 CORDIC 处理器的设计

CORDIC 处理器采用流水线结构,运算速度快,满足高速实时的要求,如图 2 所示.

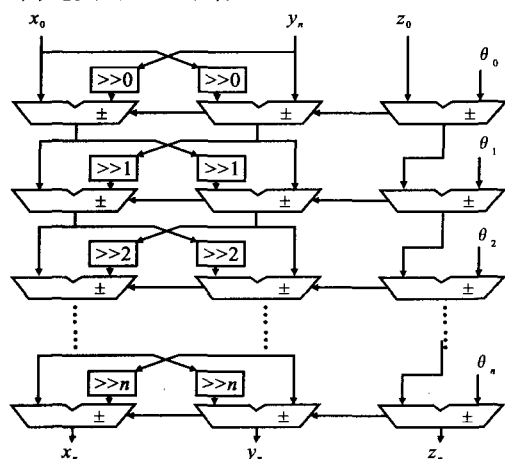


图 2 流水结构的 CORDIC 处理器

运算时只需要将数据的实部和虚部分别加到

x_0, y_0 通道,将旋转因子中的 $\frac{2nk\pi}{N}$ 作为角度从 z_0 输入,经过 n 次迭代即可得到复数相乘的结果.这里需要注意一点,对于旋转因子来说,真正的 $\theta_n = -\frac{2nk\pi}{N}$,若将 $\frac{2nk\pi}{N}$ 作为 z_0 输入时,迭代方程式应该变为

$$\begin{aligned} x_{n+1} &= x_n + S_n y_n 2^{-n} \\ y_{n+1} &= y_n - S_n x_n 2^{-n} \\ z_{n+1} &= z_n - S_n \arctan(2^{-n}) \end{aligned} \quad (5)$$

3.2.1 CORDIC 算法误差与旋转级数 n 值确定

由于 CORDIC 算法的迭代次数是有限的,所以运算只能得到近似旋转的结果. CORDIC 算法所能达到的精度与所选取的旋转级数和操作数位宽有关^[7]. 输出结果的误差主要由近似误差和舍入误差两部分组成. 近似误差是指实际旋转角度与理想旋转角度的误差值,主要是由旋转级数有限和用来表述角度的二进制位数有限引起的. 舍入误差是指每级计算中有限的计算精度引起的误差,主要是由有限的操作数宽度引起的. 根据文献^[7]的推导及仿真结果可以得出结论,对于实部、虚部均为 16bit 的数据宽度,选择旋转级数为 16 时,可以达到 10^{-4} 的数值精度. 此时对于硬件资源和计算精度都是比较合理的一种折中选择.

$\theta_n = \arctan(2^{-n})$ 是一系列的固定值,可以提前计算出来. 为保证 16 级旋转迭代,经计算, θ_n 的数据宽度最小为 16bit.

3.2.2 预旋转

在 CORDIC 乘法器中,还需要特别注意的一点是,当旋转角度超出一定的范围时,需要先进行预旋转,然后再进行迭代运算,否则旋转角度最终不收敛. 由于所能获得的最大旋转角度为 $z_{\max} = \lim_{n \rightarrow \infty} \sum_{i=1}^n \arctan(2^{-i}) = 1.7433 \approx 99.911^\circ$, 所以将旋转范围确定为 $[0, \frac{\pi}{2}]$, 超出该范围的角度,通过简单的三角函数变换映射到该区域内,具体操作如表 1 所示.

表 1 旋转角度预处理

旋转角度	预旋转角度	x_{out}	y_{out}
$[0, \pi/2]$	0	x_{in}	y_{in}
$[\pi/2, \pi]$	$\pi/2$	y_{in}	$-x_{\text{in}}$
$[\pi, 3\pi/2]$	π	$-x_{\text{in}}$	$-y_{\text{in}}$
$[3\pi/2, 2\pi]$	$3\pi/2$	$-y_{\text{in}}$	x_{in}

3.3 旋转因子的产生方法

旋转因子生成的常用方法是将角度的正弦余弦值预先存储在 ROM 中,然后通过控制电路送入蝶形单元中. 文中提出一种简单易行的产生方法,不需要进行 ROM 存储,节省了硬件资源,对于大点数的 FFT 来说尤其有益. 可配置点数的 FFT 尽管参加运算的点数和所需的级数不同,但对不同点数的同一级(stage)来说,其旋转因子有着相同的规律可循. 对于第一级(stage = 0)蝶形运算,不需要乘以旋转因子,因而将存储器中的原始数据直接输入到蝶形单元中进行运算. 在第二级(stage = 1)运算中,数据需要乘以的旋转因子均为 W_N^{nk} 的形式,其中 $n = 0, 1, 2, 3, k = 0, 1, 2, 3$, 每 16 个数据可以看作一组,各组旋转因子相同. 第三级(stage = 2)的旋转因子形式均为 W_{64}^{nk} ,其中 $n = 0, 1, 2, 3, k = 0, 1, 2, \dots, 15$, 每 64 个数据可看作一组,依次类推. 可见旋转因子的分母可由级数确定. 由于 z_0 和 θ_n 实质上只需要比较大小以确定旋转方向,而不关心最终结果,所以可以将 z_0 和 θ_n 提取出因子 $\frac{\pi}{M}$, 这样只需要关心分子中的 n 和 k 即可得到旋转因子的值. n 和 k 的值与数据在 SRAM 中存储的序号相关,尤其对于原位计算来说,这种产生方法非常简单方便,具体关系如表 2 举例所示. 对于 N 点 FFT,将 N 个数据的存储序号写成二进制形式,其中各级旋转因子的分母(如 16, 64...) 减 1 对应的 bit 位数的高两位颠倒次序即为该数据对应的 n 值,其余低位表示的即为 k 值.

表 2 旋转因子的产生

Stage	Index	k	n	W_N^{nk}
1	$b_{14}b_{13}\dots b_3b_2b_1b_0$	b_1b_0	b_2b_3	
	00...0000	00	00	W_{16}^0
	00...0001	01	00	W_{16}^0
	00...0010	10	00	W_{16}^0
	00...0011	11	00	W_{16}^0
	00...0100	00	10	W_{16}^0
	00...0101	01	10	W_{16}^0
	00...0110	10	10	W_{16}^0
	00...0111	11	10	W_{16}^0
	00...1000	00	01	W_{16}^0
	00...1001	01	01	W_{16}^0
	00...1010	10	01	W_{16}^0
	00...1011	11	01	W_{16}^0
	00...1100	00	11	W_{16}^0
	00...1101	01	11	W_{16}^0
	00...1110	10	11	W_{16}^0
	00...1111	11	11	W_{16}^0

3.4 块浮点与溢出控制

对于大点数 FFT 来说需要经过多级乘、加运算得到最终结果,每一级蝶形运算都会使输出数据的模值增大,因此在运算中如何保证精度和不溢出是一个十分重要问题.

FFT 的运算数据可分为浮点、块浮点和定点 3 种类型.定点数运算实现简单、速度快,但动态范围小运算精度也不高.浮点 FFT 需要将输入转换成由符号位、指数位和尾数位组成的浮点表示形式.只要指数位数足够大,浮点 FFT 计算是不会溢出的,且运算精度高.但其主要缺点是电路实现复杂且速度较慢.块浮点表示法兼有定点和浮点的某些优点,在一个数据块上实现浮点,一组数据共享一个指数.在对数据块执行加法和乘法运算时,仅对尾数进行加法和乘法运算即可,与定点运算一样方便快捷.本设计采用块浮点,在每级的数据输入端,对符号位进行扩充;在每级的数据输出端,对最高 3 位进行检测,以确定是否存在溢出以及溢出的位数,再做出相应的移位处理.在一个基 4 蝶形运算单元中,由于旋转因子实部和虚部的绝对值总是不大于 1 的,故乘法运算不会引起输出数据位数的增加,而输入数据的加减也最多只有两次进位.

块浮点 FFT 算法以定点运算的速度达到了浮点运算的精度^[8],兼有两者的优点,且耗费资源与定点运算相当,对于应用于雷达 SAR 信号处理系统的 FFT 来说,是一种经济实用的算法.

3.5 并行无冲突地址生成方法

本设计中用两块 SRAM 来存储输入数据及中间运算结果,这两块 SRAM 采用配置 FPGA 内置 RAM 资源的方式实现.内置 RAM 为双端口,每次只能对一个数据进行读写操作,对于 16 个数据并行处理的结构就需要将每块 SRAM 分成 16 个子存储体,每个子存储体的深度为 2048.数据开始输入系统时不能依次存储,而是要保证每次读取的 16 个数据存储在不同的子存储体中.图 3(a)所示为按数据倒序后依次存储,第一级运算中蝶形运算的抽取间隔为 1,第二级抽取间隔为 4,不存在地址冲突的问题,但第三级抽取间隔为 16,即第 0、16、32、48 个数为一组进行蝶形运算时,则发生读取地址冲突.因此,为保证高速并行地访问存储器,本文研究了一种适合 16 点并行的无冲突地址产生方法.由蝶形单元的抽取方式不难发现,无论抽取间隔是多少,每相邻的四个数,即(0、1、2、3),(4、5、6、7),...,总是同时被读出,可视为一组,需要同时读出的组要放在不同的子存储体中,如图 3(b)所示.数据所存入的子存储体做为块地址,可以用 4bit 二进制数表示,根据以上分组的思想不难找出块地址的产生规律,可以用图 3(c)中的加法运算来实现.由于在存储体内的点地址是按照数据顺序号 index,每 16 个数加 1,因此点地址即为 $\text{index}[14:4]$.

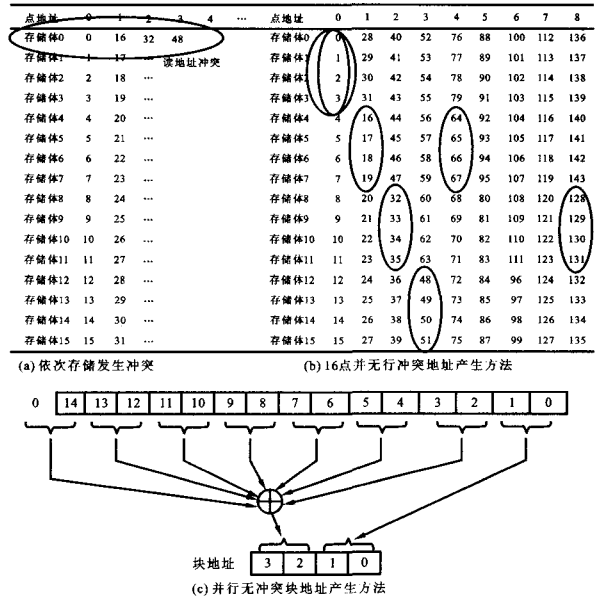


图 3 16 点并行无冲突地址产生方法

4 功能验证与 FPGA 实现

本设计全部由 Verilog 语言实现,用 Modelsim

进行功能仿真,如图 4(a)所示,计算结果与 Matlab 结果进行比较显示功能正确.采用 Altera Stratix II EP2S180F 1020C4 FPGA 进行实现,两块 SRAM 采用配置 FPGA 内置 RAM 资源的方式实现.经 Quartus II 6.0 仿真综合,整个系统占用 51 181 个 ALUTs,占总数的 36%,综合结果显示系统最高可达 118.89MHz,仿真结果如图 4(b)所示.

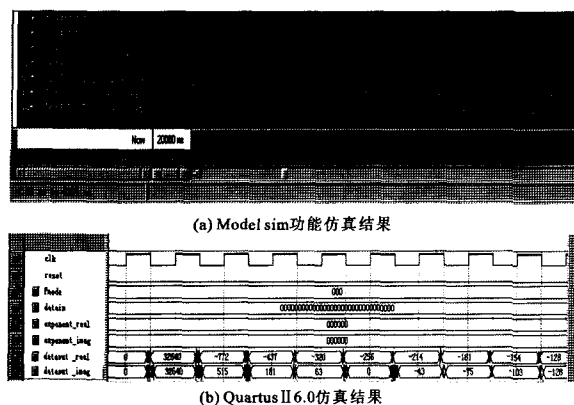


图 4 FFT 处理器仿真结果

5 结语

文中研究了一种基于 CORDIC 算法的高速可配置块浮点型 FFT 处理器的 FPGA 实现.能够实现 64~32k 点,实部、虚部均为 16bit 数据的 FFT 运算,既可以用于 SAR 的数据处理系统,也可以作为通用 FFT 处理器使用.提出一种新型便捷的旋转因子产生方法,节省了大量 ROM 资源.无冲突地址产生方法允许 16 点并行处理,大大提高了系统的并行度和运算速度.在 100MHz 频率下,1024 点 FFT 的计算时间仅为 $4.48\mu\text{s}$, 32k 点 FFT 的计算时间为 $65.7\mu\text{s}$,完全满足高速实时的运算要求.

参考文献:

- [1] 李伟,孙进平,王俊,等.一种基于 FPGA 的超高速 32k 点 FFT 处理器[J].北京航空航天大学学报,2007,33(12):1440-1443.
- [2] Yu C Y, Chen S G, Chih J C. Efficient CORDIC designs for multi-mode OFDM FFT[C]//International Conference on Acoustics, Speech and Signal Procession (ICASSP2006). Toulouse, IEEE, 2006: III/14-III/19.
- [3] 董惠,卫铭斐,江丽,等.基于 FPGA 的 FFT 处理器的设计与仿真[J].微电子学与计算机,2008,25(11):117-120.
- [4] Garrido M, Grajal J. Efficient memoryless CORDIC for FFT computation[C]//International Conference on Acoustics, Speech and Signal Procession (ICASSP2007). Honolulu, IEEE, 2007: II/113-II/116.
- [5] Liu G H, Feng Q Y. ASIC design of low-power reconfigurable FFT processor[C]//ASICON'07. China: Shanghai, IEEE, 2007: 44-47.
- [6] 元中瑞,张浩,邱昕,等.802.16e 系统变长可配置 FFT 的设计与实现[J].微电子学与计算机,2008,25(11):1-4.
- [7] 韩芳,初建朋,赖宗声.一种 CORDIC 算法的精度分析及其在 FFT 设计中的应用[J].微电子学与计算机,2004,21(7):14-16.
- [8] Kim H G, Youn J S, Yoon K T, et al. 8K-point Pipelined FFT/IFFT with compact memory for DVB-T using block floating-point scaling technique[C]//Information and Communications Technology. (ICICT2007). Melbourne, IEEE, 2007: 41-44.

作者简介:

文婧媛 女,(1983-),硕士研究生.研究方向为数字集成电路设计.

徐欣锋 男,(1977-),副研究员.研究方向为嵌入式计算和数字集成电路 ESL 设计.