摘　要

论文阐述了一种自适应抗干扰阵列信号处理系统的高速实现，算法采用基于 QR 分解的递归最小二乘算法(QRD-RLS)，在系统的设计中采用了超前处理技术。超前处理技术通过引入并行机制把串行的自适应阵列信号处理算法转换成了并行的自适应阵列信号处理算法。通过对超前处理技术的采用，使基于 QR 分解的最小二乘算法具有优良的流水粒度。阐述了系统的基于 FPGA 的硬件构架，该构架的宏单元由能够完成 Givens 旋转的 CORDIC 运算宏单元来充当，而且可以在不损失收敛速度的前提下达到很高的采样速率。设计了承载此系统的硬件平台。本文第一章概述了高速实时自适应抗干扰阵列处理技术及其研究进展。第二章研究了适于引入并行流水机制的基于 QR 分解的递归最小二乘算法(QRD-RLS)，并对此算法的拓扑结构和运算宏单元进行了阐述。第三章对上一章所阐述的系统构架进行改进，引入了更强的并行机制，给出了应用超前处理技术的系统硬件结构。第四章进行了基于 FPGA 的系统设计，其后搭建了系统的硬件平台。

关键词：QRD-RLS；超前处理技术；并行结构；流水处理；CORDIC

ABSTRACT

The paper presents the realization of adaptive least squares array signal ssing algorithm based on QR decomposition adopting annihilation-reorder -ahead technology. The annihilation-reorder look-ahead technology asforms a sequential adaptive array signal processing algorithm into an quivalent concurrent adaptive array signal processing algorithm by creating additional concurrency in the algorithm. The annihilation-reorder look-ahead is employed to develop fine-grain pipelined QR decomposition-based RLS adaptive algorithm. The proposed architectures can be operated at arbitrarily high sample rate without degrading the convergence behavior and consist of only Givens rotations, which can be scheduled onto CORDIC arithmetic-based processors based on FPGA. In the end, the hardware plat for the adaptive anti-jamming array system is designed. Chapter 1 addresses the present conditions and development of the high-speed and real-time adaptive anti-jamming arrays technology. Chapter 2 deals with recursion least-squares algorithms based on QR department that is adopt to be implement intercurrent and describes the topology and processors than can be used to realize the system.Chapter 3 studies the annihilation-reorder look-ahead technology that is employed to develop fine-grain pipelined QR decomposition-based RLS adaptive algorithm and proposes the updated algorithm with the annihilation-reorder look-ahead technology. Chapter 4 discusses the design of the adaptive anti-jamming arrays system based on FPGA and the creation of the hardware plat on which the system runs.

Key words : QRD-RLS , Annihilation-reorder Look-ahead , Parallel Architectures, Pipeline Processing, CORDIC

哈尔滨工程大学

学位论文原创性声明

本人郑重声明：本论文的所有工作，是在导师的指导下，由作者本人独立完成的。有关观点、方法、数据和文献的引用已在文中指出，并与参考文献相对应。除文中已注明引用的内容外，本论文不包含任何其他个人或集体已经公开发表的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者（签字）：徐法绿

日期：2004 年 3 月 2 日

第1章 绪 论

1.1 高速实时自适应抗干扰阵列处理技术

自适应阵列处理是新一代电子系统中的实用技术，在众多领域有广泛的应用前景，这些领域包括通信、雷达、导航、声纳、红外及激光传感系统、地震预报、摇感和遥测等。随着电子信息技术的发展，电磁环境越来复杂，对自适应阵列提出了更高的要求。

当前，在雷达，声纳和通信等系统中，作为抑制干扰和降低噪声损害的一种有效手段，自适应阵一直被人们广泛采用和深入研究的重要课题。之所以对自适应阵系统感兴趣，主要因为这种系统既具有自动感知扰源存在并抑制其影响的能力，又具有增强所需接收信号的能力，在此过程中无需知道关于所需信号和干扰环境的先验信息。另外，还可以通过合理设计自适应阵，来辅助其它抗干扰技术，比单纯依靠一种常规手段(如采用扩展频谱技术，或者采用强定向性传感器)，所获得的实际抗干扰能力更大。

自适应阵是一个由传感器阵和实时自适应信号接收机一处理机所组成的系统，它能够自动高速调整传感器的灵敏度方向图，使得阵性能质量得到改善，自适应阵比起常规阵更可靠，把方向图波束的零陷适当调向干扰源方向，并降低边波束电平，即可做到抑制干扰信号：与此同时保持合乎要求的主波束(信号波束）特性，即可保证接收所需信号。自适阵系统正是紧紧依靠这种空间特性改进了输出信噪比。灵敏度方向图可以在窄频带内形成很深的零陷，所以有效抑制很强的干扰。

随着无线通信技术的发展，信号的带宽在不断的提高，信号处理的计算量在急剧增加，这就产生了高速实时自适应阵列处理技术。首先，微波技术和数字信号处理器件的发展，微波技术的发展主要体现在砷化镓单片微波集成电路(MMIC)的广泛应用，以 MMIC 发/收组建构成固态有源阵列，将会大大提高阵元的可靠性，同时，有源阵列便于实现数字波束形成(DBF)和与其相关联

的阵列信号处理。数字信号处理器件的发展主要体现在超大规模集成电路(VLSI)技术的发展，DSP、FPGA、MCU 的集成度和运算速度有了极大的提高。其次，算法和系统构架的发展，在算法方面，出现了很多收敛速度大、计算复杂度低、数值稳定性好、适于并行流水实现的高速算法[1][2]。在系统构架方面，出现了脉动阵列(Systolic)[3]和坐标旋转计算技术 (CORDIC)[4][5][6]；脉动阵列的各个单元间采用局部数据通信，是一个局部串行全局并行[3]的拓扑结构，因此具有良好的并行流水性。坐标旋转计算技术可以将乘、取模、取幅角等运算用加和移位来完成，大大简化了设计难度，提高了运算速度。

1.2 自适应抗干扰阵列信号处理基础[2]

在图 1.1 中有 M 个阵元，设接收的信号 $x_i(n)$， $i=1,2,\cdots,p$ 是窄带信号，在 n 时刻的输出 $y(n)$ 可写成

$$y(n)=\sum_{i=1}^{p} w_i^* x_i(n)=w^H x(n) \tag{1-1}$$

式中， w 是加权系数 w_i， $i=1,2,\cdots,p$ 的矢量， $w=\begin{bmatrix} w_1 & w_2 & \cdots & w_p \end{bmatrix}^T$；

$x(n)=\begin{bmatrix} x_1(n) & x_2(n) & \cdots & x_p(n) \end{bmatrix}^T$ 为信号矢量，"H"符号表示共扼转置。

在许多情况下，阵列的接收机是由正交通道组成(即 I 和 Q 两通道)，所以这里假设信号和加权系数都为复数，也就是输入信号为复平面波，波的到达方向为 θ。为方便起见，设第一阵元的接收信号的相位为零，即

$$x_i(n)=a_i \exp(jwn-\Delta_i\theta)，\ \Delta_1=0 \tag{1-2}$$

式中，当 $i=2,3,\cdots$ 时， $\Delta_i>0$。为简单起见，令 $a_i=1$，则波束形成器的输出 $y(n)$ 由式(1-1)和(1-2)写成：

$$y(n)=\left(\sum_{i=1}^{p} w_i^* e^{-j\Delta_i\theta}\right)e^{jwn} \tag{1-3}$$

其中， $\Delta_1\theta=0$。因此，波束形成器的响应被定义为输出的幅度和相位，即响应为：

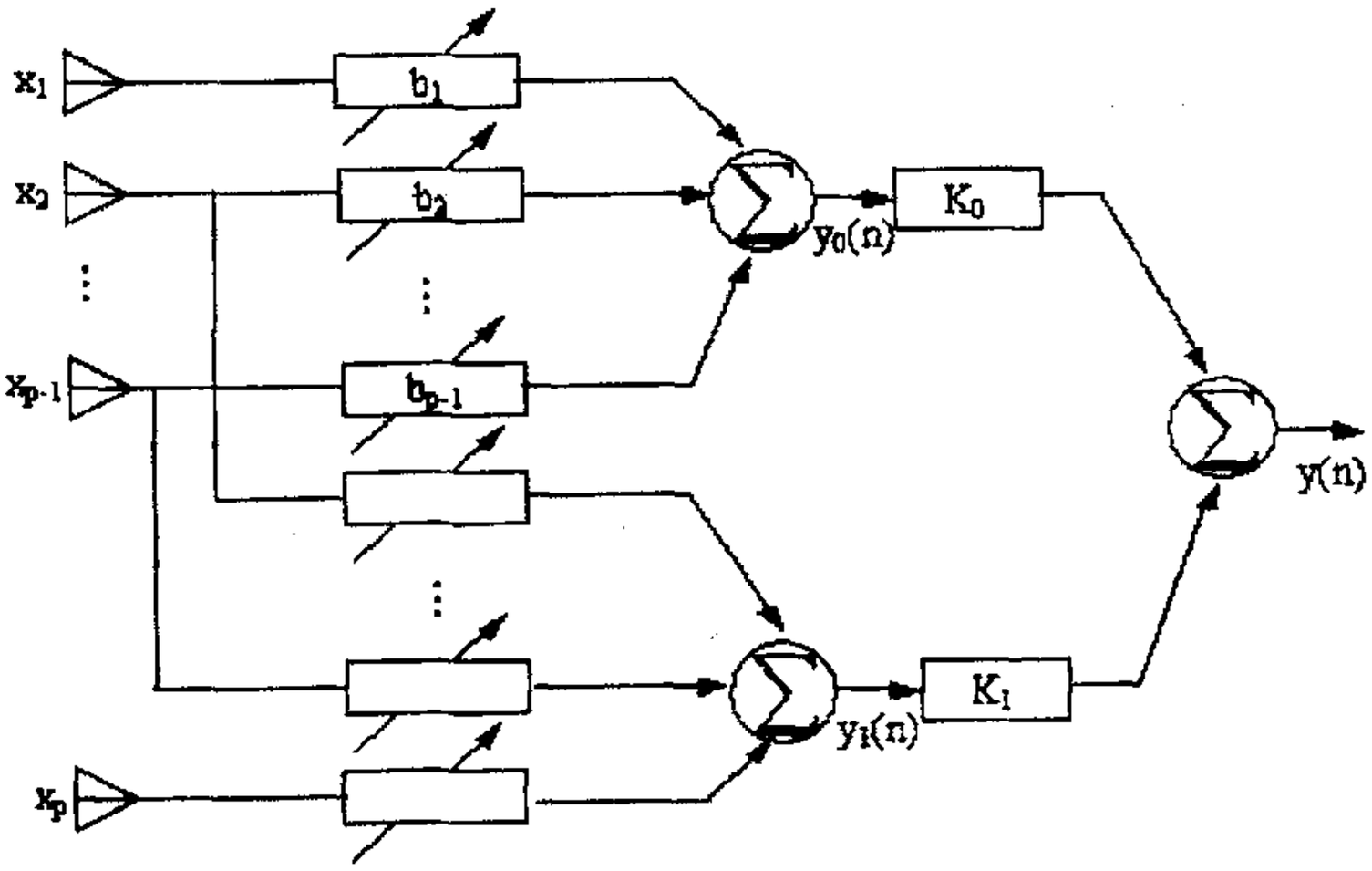


图 1.1 波束形成器示意图

$$r(\theta,w)=w^{H}a(\theta,w) \tag{1-4}$$

式中，矢量$a(\theta,w)$等于

$$a(\theta,w)=\begin{bmatrix}1 & e^{-j\Delta_2\theta} & \cdots & e^{-j\Delta_M\theta}\end{bmatrix}^{T} \tag{1-5}$$

通常，称$a(\theta,w)$为信号的导引矢量(Steering Vector)，它是一种用来描述一个复平面波以θ方向到达阵列的，出现在每个阵元上的信号矢量。图 1.2 表示一种自适应波束形成器，它由引入导向矢量与阵元上信号矢量作用于自适应控制算法，来调节波束形成器的权系数。

当传感器阵列中阵元间距相等时，由图 1.3 可见，空间延迟的$d\sin\theta$，导引矢量$a(\theta,w)$可写成

$$a(\theta,w)=\left[1 \quad \exp(-\frac{2\pi d}{\lambda}\sin\theta) \quad \cdots \quad \exp(-\frac{2\pi(p-1)d}{\lambda}\sin\theta)\right]^{T} \tag{1-6}$$

其中，d为阵元间距；λ为信号波长。

通常用$r(\theta,w)$的平方定义阵列的方向图，式(1-4)的矢量表示法使我们容易地用矢量空间理论来解释波束的形成。这一观点在波束形成器的设计和分析上特别有用。权矢量w及方向矢量$a(\theta,w)$可以看作 M 维空间中的矢量，两

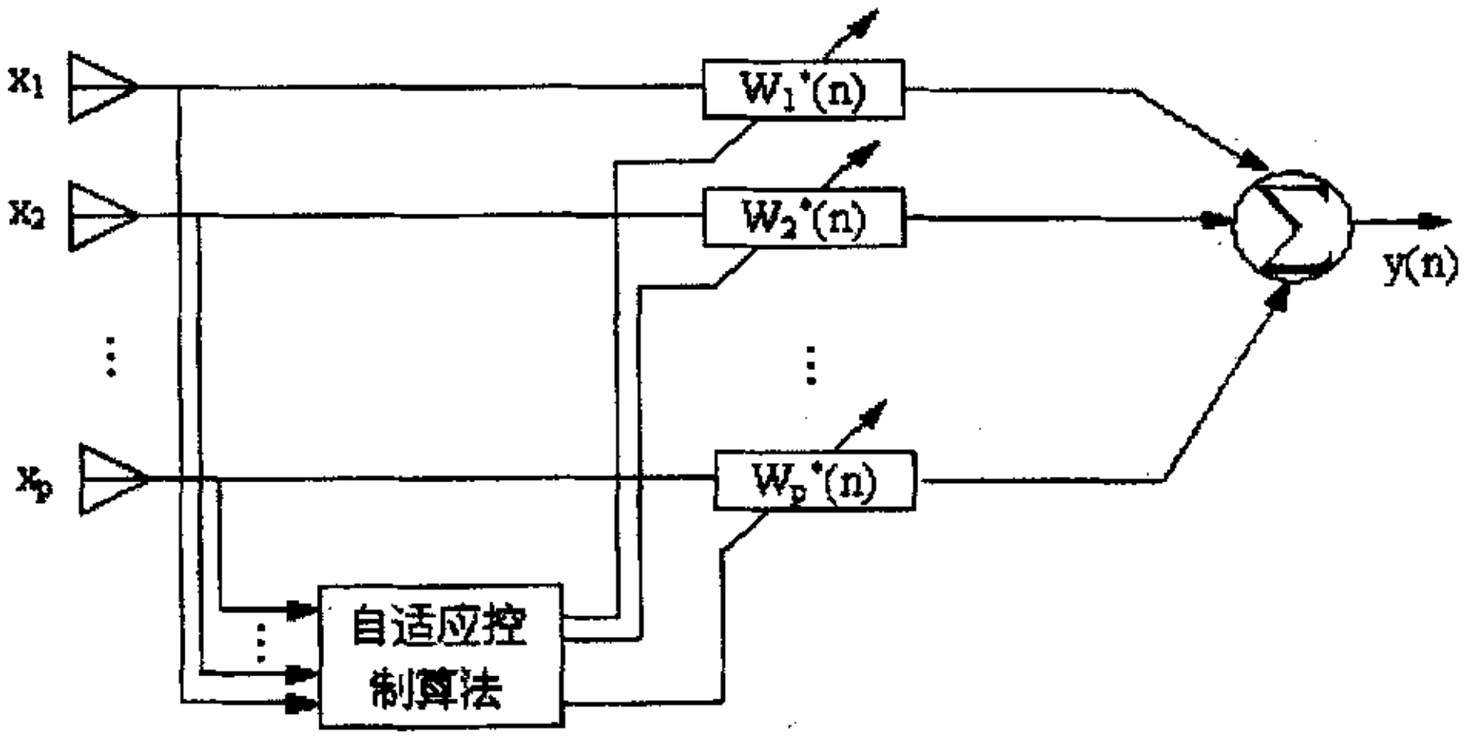


图 1.2 自适应波束形成器的一种方案

个矢量之乘积确定了波束形成器的响应。换句话说，响应$r(\theta,w)$ 取决于两矢量w与$a(\theta,w)$之间的夹角，当w与$a(\theta,w)$两矢量正交，夹角为 90°时，响应则为零;当两矢量之间夹角为 0° 时，则响应幅度为最大。因此，不同位置(θ_1,w_1)和(θ_2,w_2)上信号源的区分能力就取决于$a(\theta_1,w_1)$和$a(\theta_2,w_2)$之间的夹角。

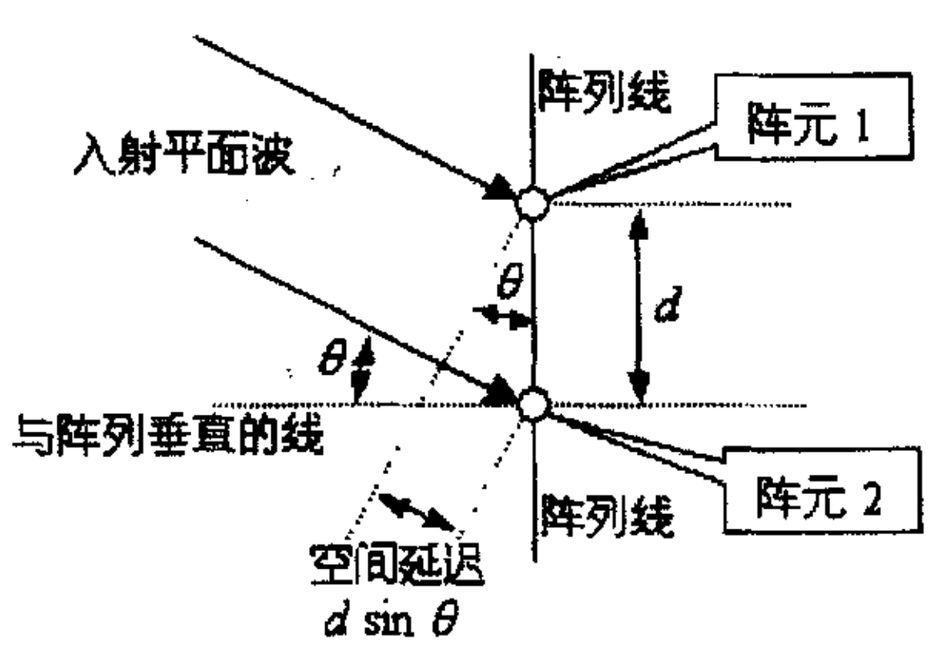


图 1.3 线性阵列空间延迟

与时域滤波器相似，波束形成器是一种空域滤波器，也存在空间混迭问题。在空间中出现混迭，意味着$a(\theta_1,w_1)$与$a(\theta_2,w_2)$之间有重叠，这是阵元间距太大所引起的。为了避免这一情况，一般取$d \le \frac{1}{2}\lambda$。当然，如果阵元间距太小，将造成响应矢量在p维空域内不能很好地分开。

在波束形成器的分析与设计中，往往会涉及功率及方差，即数据的二阶统计量起着非常重要的作用。假设阵元接收到的信号是零均值，则波束形成器的方差或期望功率由下式决定：

$$E\left[(Y)^2\right]=w^H E\left(xx^H\right)w=w^H R_{xx}w \tag{1-7}$$

如果数据是广义平稳过程，则 R_{xx} 与时间无关。假设信号矢量 x 表示来自方位角为 θ 的源，则得到

$$R_{xx}=\int_{w_a}^{w_b}S(w)a(\theta,w)a^H(\theta,w)dw \tag{1-8}$$

式中，$S(w)$ 为信号的功率谱；w_a,w_b 为信号频带宽的边界角频率。如果 R_{xx} 能表示秩为 1 的外积，则有

$$R_{xx}=\sigma_s^2 a(\theta,w_0)a^H(\theta,w_0) \tag{1-9}$$

这里 σ_s^2 表示信号源的功率，w_0 表示信号源为窄带的带宽。

当有 M 个信源存在时，则输入信号矢量 x 将由它们各自的作用之和给出：

$$x=x^1+x^2+\cdots+x^M \tag{1-10}$$

如果所有的信号源之间是不相关的，则得到

$$R_{xx}=E[(x^1+x^2+\cdots+x^M)(x^1+x^2+\cdots+x^M)^T]$$

$$=E\left[(x^1)(x^1)^H\right]+\cdots+E\left[(x^M)(x^M)^H\right] \tag{1-11}$$

如果再考虑到接收机的噪声，实际上得到的数据是

$$x=x^1+x^2+\cdots+x^M+v \tag{1-12}$$

这里，$v=\left[v_1+v_2+\cdots+v_p\right]^T$ 是 p 个接收机的复高斯噪声，可以认为他们是相互独立的，方差同为 σ_v^2，所以得到

$$R_{xx}=E\left[(x^1)(x^1)^H\right]+\cdots+E\left[(x^M)(x^M)^H\right]+\sigma_v^2 I \tag{1-13}$$

式中，I 为单位阵。

1.3 自适应阵列信号处理的研究进展

自适应阵列信号处理技术对雷达、通信、声纳和地震勘探等领域都具有重要应用价值，不同的应用领域，对处理方式要求有所不同，但工作原理是一致的，都是基于维纳滤波理论。自适应算法则要将这种理论在实际的信号处理器或计算机上予以实现。

最早的自适应算法是闭环梯度型算法。在六十年代初，Howells 和 Applebaum[7]首次提出了最大信噪比准则下，由模拟电路实现的闭环梯度算法，与此同时，Windrow 等人[8]提出了在最小均方差准则下，数字实现的 LMS 算法，两种算法实现不同，原理极其相似，最终都收敛于最佳维纳解。Frost[9]和 Griffiths[10]又进一步将 LMS 算法推广应用于线性约束自适应波束形成。梯度型算法实现简单，性能可靠，不需数据存贮。其主要缺点是收敛于最佳权的响应时间取决于数据特征值分布，在某些干扰分布情况下，算法收敛速度很慢。因此，对很多要求具有快速响应的实时应用场合，梯度型算法是不适宜的。

近十多年来，人们把兴趣更多地集中在开环算法的研究上，以适应日益增长的需要。开环算法是在最小平方误差准则下，将自适应阵列信号处理化为求解超定线性方程组的最小二乘解问题。开环算法的优越性在于，只需很少的数据就能够准确地描述外部环境，从而提供在较宽动态范围内抑制干扰的方向图，而与特征值分布无关。事实上，开环算法可以认为是实现自适应处理的最佳途径。

从数学上可以知道，求解最小二乘问睡的途径主要有两条，一是正规方程求解方法，再是基于系数矩阵 QR 分解的求解方法，后者的运算量几乎是前者的两倍，但数值特性要好得多。自适应波束形成开环算法无外乎也都基于这两条基本途径，分别称为均方域算法和数据域算法。

Reed 等人[11]最早提出的著名的采样协方差矩阵求逆(SMI)算法，则属于均方域算法。SMI 算法还有其他的实现形式，象逆矩阵递推算法[12]，权矢量递推算法[13]等。均方域算法需显式计算采样协方差矩阵，相当于对采样数据进行平方运算，大大增加了数据动态范围，使得在许多实际情况中协方差矩阵出现病态，严重影响算法数值特性。此外，均方域算法结构上不利于并行

实现，采用传统的单处理器顺序计算实现方式，算法运算量严重限制自适应权矢量的更新速率，使得自适应处理阶数不可能很高。

由于均方域算法存在的不足，近年来，数据域算法成为研究的热点。数据域算法的主要优点在于，不需显式计算采样协方差矩阵，因而数值特性远优于均方域算法;算法结构利于并行实现,而与现代数字处理技术的发展潮流相适应。目前，数字信号处理器性能日益提高，运算速度已接近或达到每秒一亿次浮点数运算，但仍不能充分满足当前实时信号处理的要求，因此普遍认为，计算处理能力要有真正大的提高，则要求有高度并行的算法和并行处理结构。由于价格便宜的大规模集成电路和分存式结构都趋于成熟，计算和设计现在已进入并行处理时代。算法的并行性成为衡量算法性能的一个重要方面。

数据域的高速算法的硬件实现是这样一步一步发展起来的：继 1979 年 Kung 和 Leiserson 提出脉动阵列处理技术后，1981 年 Gentleman 和 Kung 提出了解很多自适应滤波算法所依赖的最小二乘问题的脉动阵列算法[14]。1983 年 Micwhirter 在此基础上提出了一个不求权系数而能直接得到残差的递推最小二乘脉动三角阵算法[15]。1984 年，ward 等人把这个三角阵用于自适应波束形成[16]。1959 年，Volde 提出了坐标旋转计算(CORDIC) [6]，后来，Walthe 和 Hu 又对其进行了深入的研究和应用[4]、[6]、[17]。在前人的基础上，Liu 和 Cioffi 等人提出了基于 CORDIC 的快速 QR 分解递推最小二乘算法[18]、[19]、[20]、[21]。在另一方面，Parhi 等人很好的将流水和并行处理机制应用到了自适应数字滤波器中[22]、[23]；Raghunath 和 Parhi [24]提出了应用正切旋转(STAR)的流水 RLS 自适应滤波技术，应用 STAR 比应用 CORDIC 的硬件量能够降低，但滤波性能有所下降。2000 年，Ma、Parhi 和 Gao [25]、[26]把快处理技术应用到了自适应滤波中，大大提高了系统的采样速率。

总的来说，有两种类型的 QRD-RLS 算法，一种是不计算权向量的[1]、[15]，叫权向量隐式算法，这种算法适用于自适应波束形成，在这种算法中只是计算出残差向量，而不计算权向量，本文讨论的就是这种类型；另一种是需要计算出权向量的值[27]-[30]，叫权向量显式算法，这种算法适用于信道均衡。

在实现方面，欧洲通信委员会(CEC)在 1995 年初开始现场试验。实验评测了采用 MUSIC 算法判别用户信号方向的能力，采用的自适应算法有

NLMS(Normalized Least Mean Squares)算法和 RLS(Recursive Least Square)算法。同时，通过现场测试，表明圆环和平面天线适于室内通信环境使用，而像市区环境则采用简单的直线阵更合适。CEC 准备继续进行研究，具体问题集中于以下方面：最优波束形成算法、系统协议研究与系统性能评估、多用户检测与自适应天线结构、时空信道特性估计及微蜂窝优化与现场试验。ArrayComm 公司和中国邮电电信科学研究院信威公司研制出应用于无线本地环路(WLL)智能天线系统。ArrayComm 产品采用可变阵元配置，有 12 元和 4 元环形自适应阵列可供不同环境选用。在日本进行的现场实验表明，在 PHS 基站采用该技术可以使系统容量提高四倍。信威公司智能天线采用八阵元环形自适应阵列，射频工作于 1785MHz～1805MHz，采用 TDD 双工方式，收发间隔 10ms，接收机灵敏度最大可提高 9dB。德州大学奥斯汀 SDMA 小组建立了一套智能天线试验环境，着手理论于实际系统相结合。加拿大 McMaster 大学研究开发了 4 元阵列天线，采用恒模(CMA)算法。国内部分大学也正在进行相关的研究。

1.4 本文主要工作

本文给出了一个高速实时自适应抗干扰阵列系统的设计及硬件实现，具体工作如下：

1. 对几种算法进行比较，选择一种适合于硬件实时实现的高速算法。
2. 对采用此算法的自适应抗干扰阵列系统的拓扑结构和运算宏单元进行设计，其中引入超前处理技术。
3. 在 FPGA 上实现系统。
4. 搭建系统的硬件平台。

第2章 基于 QR分解的递归最小二乘算法

2.1 引言

最主要的自适应算法是最小均方差准则下的 LMS 算法和最小平方误差准则下最小二乘算法。LMS 算法实现简单，性能可靠，不需数据存贮，但收敛于最佳权的响应时间取决于数据特征值分布，在某些干扰分布情况下，算法收敛速度很慢。最小二乘算法只要很少的数据就能够准确地描述外部环境，从而提供在较宽动态范围内抑制干扰的方向图，而与特征值分布无关，而且算法的收敛速度快。

求解最小二乘问题的途径主要有两条[31]，一是正规方程求解方法，称为均方域算法；再是基于系数矩阵 QR 分解的求解方法，称为数据域算法。均方域算法需显式计算采样协方差矩阵，相当于对采样数据进行平方运算，大大增加了数据动态范围，使得在许多实际情况中协方差矩阵出现病态，严重影响算法数值特性。此外，均方域算法结构上不利于并行实现，采用传统的单处理器顺序计算实现方式，算法运算量严重限制自适应权矢量的更新速率，使得自适应处理阶数不可能很高。数据域算法不需显式计算采样协方差矩阵，因而数值特性远优于均方域算法，算法结构利于并行实现，与现代数字处理技术的发展潮流相适应。

基于以上的原因，对比了[32]、[33]、[34]中的算法，选择了基于 QR 分解的递归最小二乘算法来设计高速实时的自适应阵列信号处理系统。

2.2 脉动阵列处理器

Systolic 一词源于生理学，它的原意是指心脏有节奏地作周期性跳动，并把血液以脉动方式送往身体各部位，美国 Carnegie-Mellon 大学的孔祥重(H.T.Kung)教授等人借用这一概念，提出 Systolic(脉动)阵处理技术，并把它作为以高度并行计算变换到 VLSI 计算结构的一种通用方法[4][5]，其基本思

想是把特定问题算法分解为简单而有规律的基本操作，并用适于 VLSI 的专用芯片来实现。它寻求从算法到实现的统一，以及处理过程的高速性和灵活性的统一。

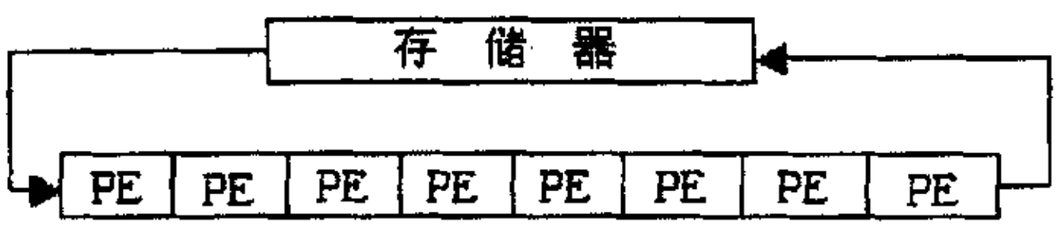


图 2.1　脉动阵列基本结构

脉动阵列的基本原理如图 2.1 所示。在脉动阵列系统中，数据以流水的方式通过由多个胞元构成的阵列而得到所需结果，它采用简单而规则的通信和控制结构，由简单而重复的处理单元(PE)组成阵列。每个处理单元能执行固定的简单操作。一个处理单元只与相邻的处理单元有规则地互连，处理单元之间的数据用流水线方式传递，整个阵列按同步方式有节奏地工作，因此它具有规整化、模块化、局部数据传输和高效率的并行运算等优点，非常适于 VLSI 实现[6]。脉动阵列集中了流水线结构和 SIMD(单指令/多数据流)阵列结构的优点，适合于开发时间和空间的高度并行性。它在流水中实现阵列运算，同时允许存在多条不同方向、不同流速的流水线，它们有机的结合使得脉动阵列可以获得相当高的计算吞吐率，脉动阵列是算法的硬件直接实现，其结构依赖于算法，而且与算法有着良好的匹配关系，它把算法中蕴含的并行性用在阵列入口的等距线上的处理单元同时工作来实现，而算法中连续执行的相同操作则用阵列中接成流永线的一串处理器来实现。在功能上，这样的阵列相当于软件中的循环语句过程，由于算法被阵列固化了，因此可以把整个阵列作为一个功能单位，这对于实时信号处理有很大的好处，下面我们要介绍的递归最小二乘脉动阵列就是很好的这样一个例子，它是脉动阵列的出色应用，可以说集中了脉动阵列的所有特点和优点。

2.3 最小二乘法

给定 $n\times p$ 维阵元输出数据构成的矩阵 $X_{n\times p}$ 和 n 维期望向量 y，找到一个 p 维向量 w，使 $Xw-y$ 的欧式范数最小，这就是最小二乘问题。下面来解决这个问题：

定义一个残差向量：

$$e = Xw - y \tag{2-1}$$

要使得欧式范数最小，需要：

$$\begin{aligned}\varepsilon &= \|e\|_2^2 \\ &= e^T e \\ &= (Xw-y)^T(Xw-y)\end{aligned}$$

最小。

上式两边对 w 求偏导，令右边部分等于零，可以得到线性方程组：

$$X^T Xw = X^T y$$

便得到最小二乘解：

$$w_{opt} = (X^T X)^{-1} X^T y$$

以上介绍的求最小二乘解的方法不仅会遇到数据多难于处理的问题，而且，方程组的相关矩阵 $X^T X$ 包含基本数据阵 X 元素的平方，其有限字长的计算往往会出现奇异阵，造成严重病态，数值性能很不稳定。基于 QR 分解的最小二乘算法可以避免构造方程组，具有良好的数值特性。

2.4 递归最小二乘的QR分解算法及其脉动阵列实现

假定对于矩阵 $X_{n\times p}$ 存在正交矩阵 $Q_{n\times n}$，使得矩阵 X 三角化：

$$QX = \begin{bmatrix} R \\ O \end{bmatrix} \tag{2-2}$$

其中，R 为 p 阶上三角阵，O 为 $(n-p)\times p$ 维零矩阵，同时有：

$$Qy = \begin{bmatrix} b \\ r \end{bmatrix} \tag{2-3}$$

其中，b 为 p 维向量，r 为 $(n-p)$ 维向量。这样：

$$\begin{aligned}Qe &= QXw - Qy \\ &= \begin{bmatrix} R \\ O \end{bmatrix} w - \begin{bmatrix} b \\ r \end{bmatrix}\end{aligned}$$

$$= \begin{bmatrix} Rw-b \\ r \end{bmatrix}$$

因为Q是正交矩阵，所以：

$$\|e\|^2 = \|Qe\|^2 = \|Rw-b\|^2 + \|r\|^2$$

如果选择w，使得：

$$Rw-b=0$$

则$\|e\|^2$将达到极小值$\|r\|^2$，且：

$$e = Q^T \begin{bmatrix} 0 \\ r \end{bmatrix} \tag{2-4}$$

最小二乘解w_{opt}很容易由回代：

$$Rw = b \tag{2-5}$$

得到。

这种 QR 分解是可递推的，当新的一行数据x_{n+1}进入后，可以用原上三角阵对这一行数据消 0，从而得到新的上三角阵及新的最小二乘解：

$$\begin{bmatrix} e_1 \\ \vdots \\ e_n \\ -- \\ e_{n+1} \end{bmatrix} = \begin{bmatrix} R_n \\ O \\ -- \\ x_{n+1} \end{bmatrix} w_{n+1} + \begin{bmatrix} p_n \\ v_n \\ -- \\ y_{n+1} \end{bmatrix}$$

$$\begin{bmatrix} e'_1 \\ \vdots \\ e'_n \\ -- \\ e'_{n+1} \end{bmatrix} = Q_{n+1} \begin{bmatrix} e_1 \\ \vdots \\ e_n \\ -- \\ e_{n+1} \end{bmatrix} = \begin{bmatrix} R_{n+1} \\ O \\ -- \\ 0 \end{bmatrix} w_{n+1} + \begin{bmatrix} p_{n+1} \\ v_n \\ -- \\ a_{n+1} \end{bmatrix} \tag{2-6}$$

于是，新的最佳权系数矢量可递推得到：

$$R_{n+1} w_{n+1} + b_{n+1} = 0 \tag{2-7}$$

以上的正交变换Q可以是 Gram-Schmidt 正交变换[13]、[38]、[39]，Householder 变换以及 Givens 变换等，其中，Givens 变换数值性好，方便用脉动阵列实

现。

直角坐标旋转变换矩阵是一个正交矩阵，旋转公式：

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} r' \\ x' \end{bmatrix}$$

当旋转角

$$\theta = -Arc\tan\left(\frac{x}{r}\right)$$

时，有 $r' = 0$。

利用这个特性，可以对新进的数据进行这种变换来消 0，从而得到新的上三角阵。采用 Givens 变换，其 $Q(n)$ 矩阵是这样得到的：

$$Q(n) = Q_p(n)Q_{p-1}(n)\cdots Q_1(n)$$

其中，

$$Q_i(n) = \begin{bmatrix} I_{i-1} & & & O & & \\ & \cos\theta_i(n) & 0 & \cdots & 0 & \sin\theta_i(n) \\ O & & & I_{p-i} & & \\ & -\sin\theta_i(n) & 0 & \cdots & 0 & \cos\theta_i(n) \end{bmatrix}$$

新进的数据 $x(n)$ 是这样被消 0 的：

首先，前一时刻上三角阵的左上角的元素 r_{11} 和 $x_1(n)$ 依照 $Q_1(n)$ 旋转，把 $x_1(n)$ 消掉，然后，第一行和最后一行的新进元素的对应列也依照 $Q_1(n)$ 旋转。接下来是第二行与最后一行进行同样的操作，消掉 $x_2(n)$。以此类推，直至得到新的上三角阵。这种过程可以映射到一个脉动阵列中去，以 $p = 4$ 为例示出了其脉动阵列的结构(图 2.1)：

其中，边界单元产生 Givens 旋转角度、消掉新进的数据，内部单元依照边界单元提供的角度进行旋转。这样就得到了新一时刻的上三角阵。

以上脉动阵列的高度的并行流水运算，加快了运算速度。如果在此阵列的基础上在右边加一列单元对 y 进行同样的旋转，就可以完成式(2-3)的运算。再加上回代单元，就可以以流水的方式进行回代运算，从而就可形成最小二乘脉动阵列，从阵列中可以直接输出最小二乘解 w [40]。由于本文对 w 并不很感兴趣，所以对此就不作详细介绍了。

在本文涉及的应用中，没有必要对权系数知道得很清楚，而只需求当前

的残差来进行实时处理。递归最小二乘脉动阵列算法正是为此而设计的。它

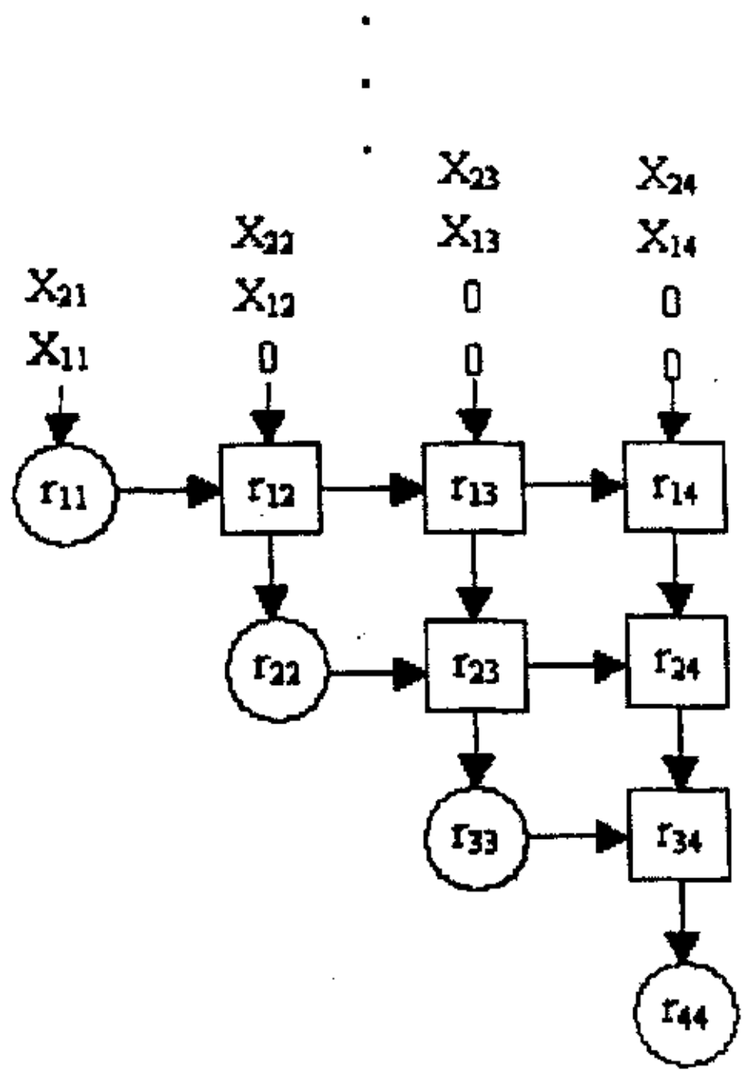


图 2.1　采用 Givens 旋转实现 QR 分解的矩阵三角化脉动阵列

在以上递推 QR 算法的基础上进行改进，使得不计算权系数而直接从阵列中输出残差。

基于(2-4) (2-6) (2-7)及矩阵的正交性，可得：

$$\begin{bmatrix} e_1 \\ \vdots \\ e_n \\ -- \\ e_{n+1} \end{bmatrix} = Q^T{}_{n+1} \begin{bmatrix} e'_1 \\ \vdots \\ e'_n \\ -- \\ e'_{n+1} \end{bmatrix} = Q^T{}_{n+1} \begin{bmatrix} 0 \\ v_n \\ -- \\ a_{n+1} \end{bmatrix}$$

这样 $n+1$ 时刻的最小二乘残差 e_{n+1} 不难得到[3]：

$$e_{n+1} = a_{n+1} \prod_{i=1}^{p} \cos\theta_i$$

a_{n+1} 可以伴随矩阵的三角化过程得到，为了得到 $\prod_{i=1}^{p} \cos\theta_i$，可以在式(2-6)中加上一列，让这一列与其他列同时旋转，旋转完成后可以得到 $\prod_{i=1}^{p} \cos\theta_i$。

考虑到新进的数据对残差的影响比以前的数据大，因此在以前的数据上乘一个遗忘因子 β（$\beta<1$），提高系统的性能。综上，得到递归最小二乘脉动阵列的表达式(式 2-8)和阵列结构(图 2.4)。

$$\begin{bmatrix} R(n+1) & y_p(n+1) & s(n+1) \\ 0^T & \alpha_{n+1} & \gamma_p \end{bmatrix} = Q(n) \begin{bmatrix} \beta R(n) & \beta y_p(n) & 0 \\ x_{n+1}^T & y(n+1) & 1 \end{bmatrix}$$

$$e_{n+1} = \gamma_p \alpha_{n+1} \tag{2-8}$$

图中的最后两列分别得到 a_{n+1} 和 $\prod_{i=1}^{p} \cos\theta_i$，经过末端的乘法器就可以得到 $n+1$ 时刻的残差 e_{n+1}。

依照式(2-8)，进行了 MATLAB 仿真，结果如图 2.2。还对未采用 QR 分解的均方域的最小二乘解法进行了仿真(图 2.3)。通过比较可以看出基于 QR 分解算法的数值稳定性。

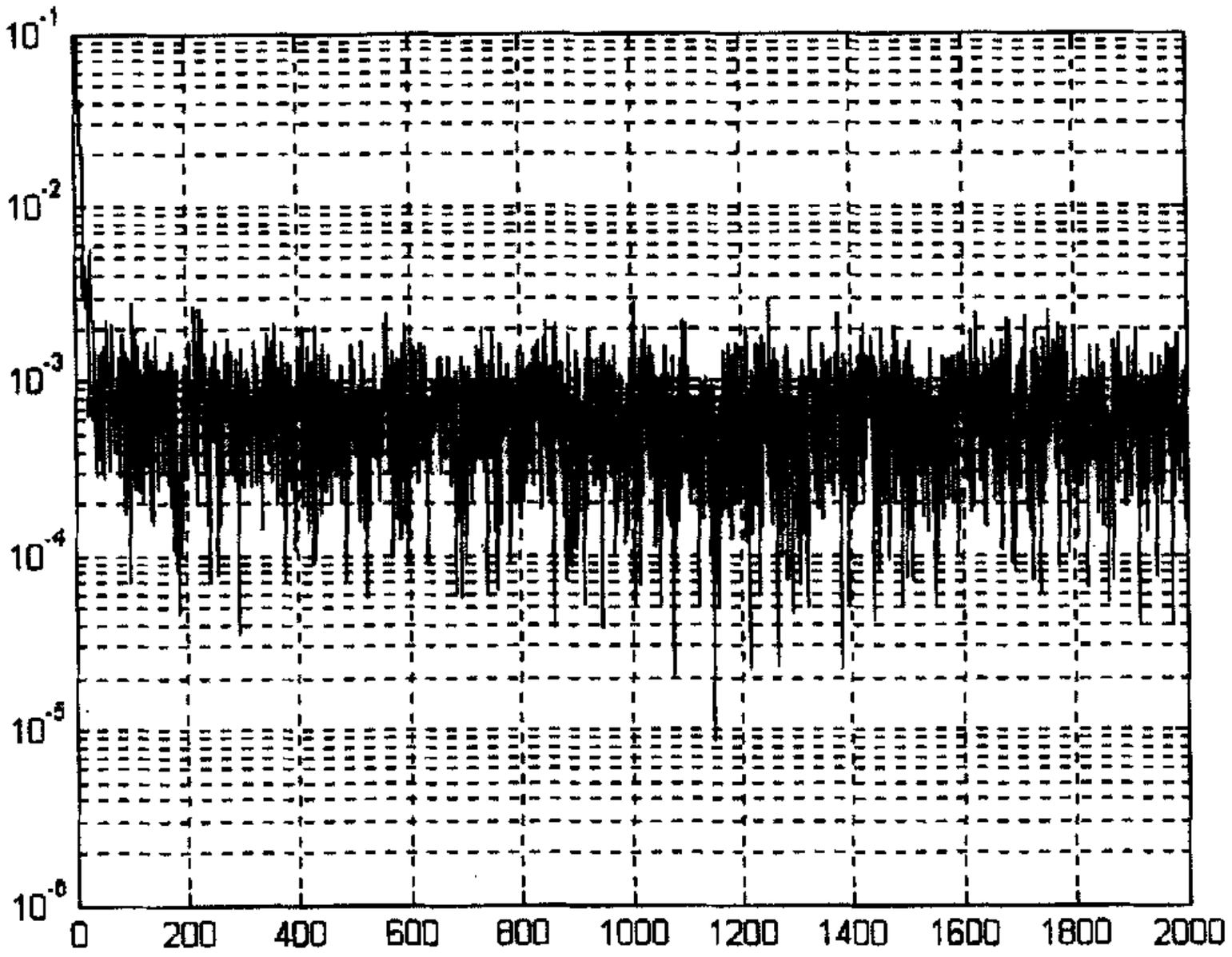


图 2.2 基于 QR 分解的最小二乘算法的残差曲线

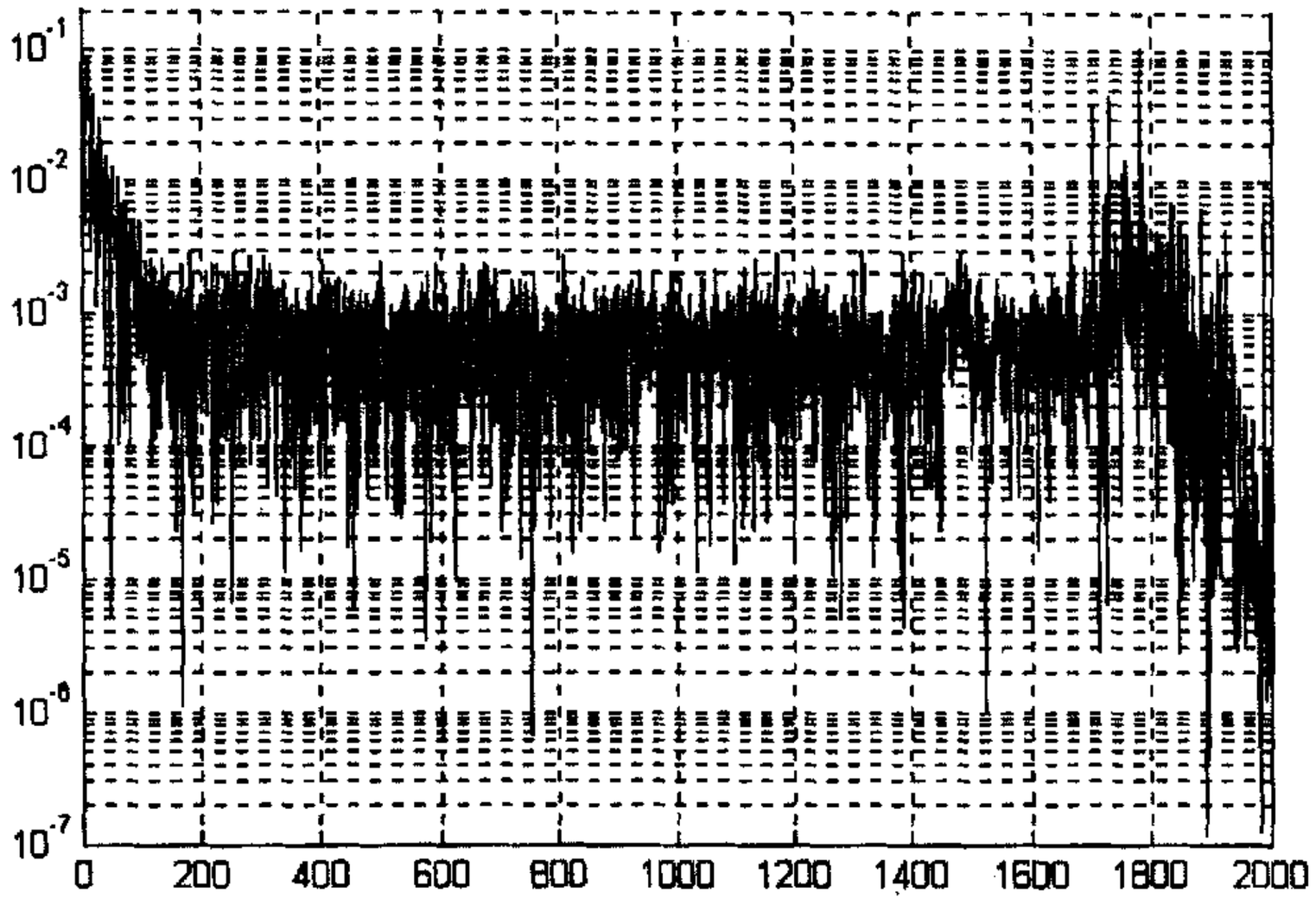


图 2.3 最小二乘算法的残差曲线

前面介绍了递归最小二乘脉动阵列，这个阵列具有收敛快、稳定性强、实时跟踪的优点，可以以较好的性能用于实时信号处理中，该阵列的主要原理是用一级一级的 Givens 旋转，使输入数据矩阵三角化同时在末端单元中直接输出残差。但从其运算内容来看，需要进行乘、除、开方等复杂运算，直接实现比较复杂，为此我们采用 CORDIC 算法来实现 Givens 旋转和其他运算，下面我们将看到 CORDIC 算法实现这些内容不仅运算简单，而且硬件实现也方便。

2.5 最小二乘的QR分解算法的CORDIC实现[59]

2.5.1 CORDIC 算法

1956 年，Volder 开发了一类计算三角函数、双曲函数的算法，其中包括指数和对数运算，1959 年，Volder 提出了 CORDIC 算法并用于导航系统[23]，使得矢量的旋转和定向运算不需三角函数表及乘、除、开方、反三角函数等

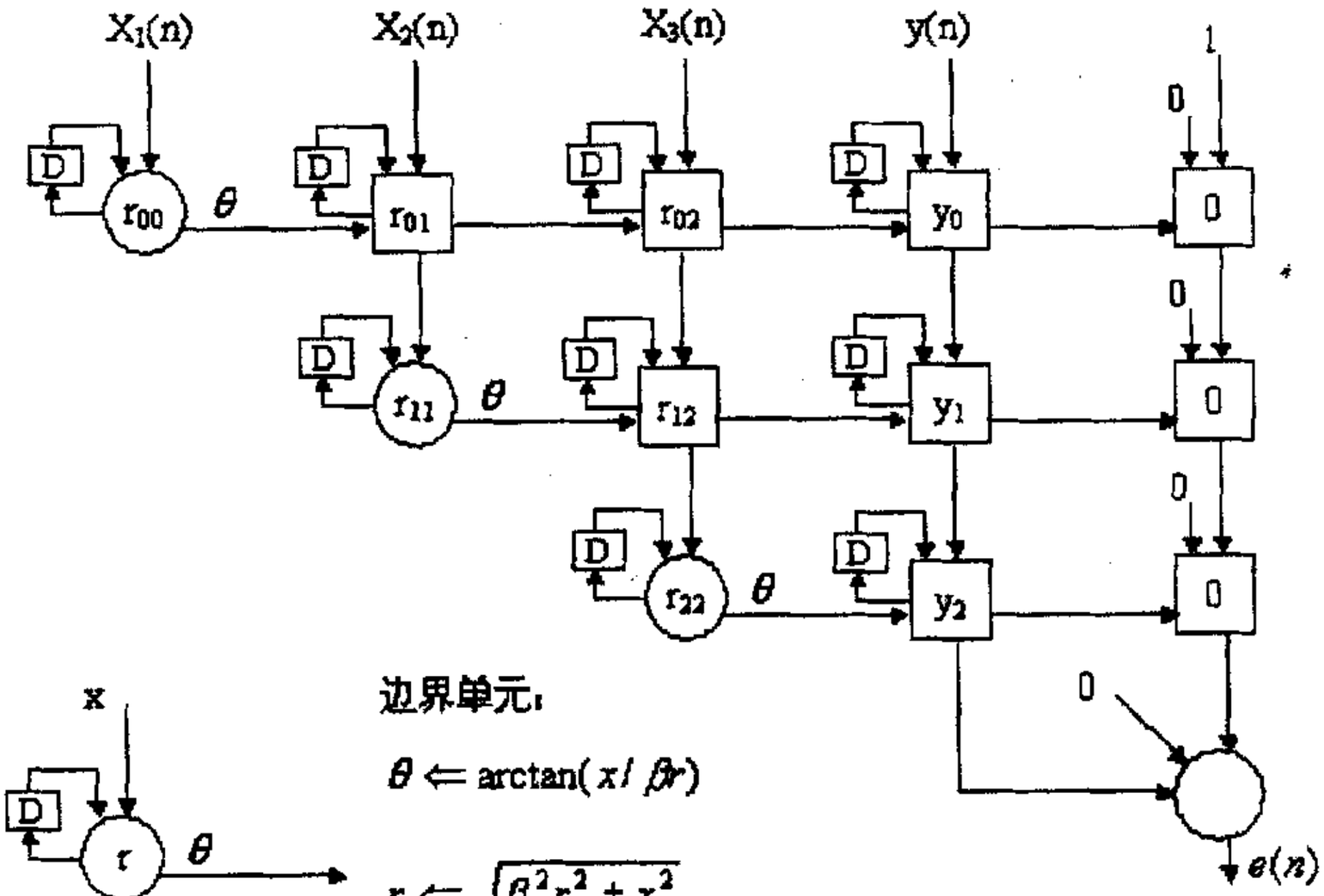


图 2.4 递归最小二乘脉动阵列

复杂运算，其基本思想是用一系列固定的与运算基数相关的角度θ_i（对于$\theta_i = Arc\tan 2^{-i}$，$i = 0,1,...,N-1$）不断偏摆从而逼近所需旋转的角度$\theta$。

$$\theta \leftarrow z = \sum_{i=0}^{N-1} \xi_i \theta_i \tag{2-9}$$

其中$\xi_i \in \{1,-1\}$描述偏摆过程。它代表第i次偏摆或旋转的方向。

当$\xi_i = 1$时，往逆时针方向偏摆。

当$\xi_i = -1$时，往顺时针方向偏摆。

于是对于整个θ角的旋转，运算过程可如下：

$$\begin{bmatrix} r' \\ x' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix}$$

$$=\prod_{i=0}^{N-1}\begin{bmatrix}\cos\xi_i\theta_i & -\sin\xi_i\theta_i\\ \sin\xi_i\theta_i & \cos\xi_i\theta_i\end{bmatrix}\begin{bmatrix}r\\ x\end{bmatrix}$$

$$=\prod_{i=0}^{N-1}\cos\theta_i\prod_{i=0}^{N-1}\begin{bmatrix}1 & -\xi_i\tan\theta_i\\ \xi_i\tan\theta_i & 1\end{bmatrix}\begin{bmatrix}r\\ x\end{bmatrix}$$

$$=K_N\prod_{i=0}^{N-1}\begin{bmatrix}1 & -\xi_i 2^{-i}\\ \xi_i 2^{-i} & 1\end{bmatrix}\begin{bmatrix}r\\ x\end{bmatrix} \tag{2-10}$$

其中，

$$K_N=\prod_{i=0}^{N-1}\cos\theta_i=\prod_{i=0}^{N-1}\frac{1}{\sqrt{1+2^{-2i}}}$$

于是，旋转运算可以由一下迭代进行：

$$\begin{cases}r_{i+1}=r_i-\xi_i x_i 2^{-i}\\ x_{i+1}=x_i+\xi_i r_i 2^{-i}\end{cases}\qquad(r_0=r,x_0=x) \tag{2-11}$$

$$\begin{cases}r'=K_N r_N\\ x'=K_N x_N\end{cases} \tag{2-12}$$

式(2-11)就是基本的 CORDIC 迭代，可以说它是连接旋转运算与其具体硬件实现的桥梁，它一方面描述如图 2.4 所示的垂直性偏摆，另一方面也是其硬件实现的描述(移位和加法)。正因为其中只有移位和加法，才是硬件实现非常简单。

但象图 2.4 所示的垂直性偏摆。它每偏摆一次模值要扩大$\sqrt{1+2^{-2i}}$倍因此整个迭代完成后，要进行式(2-12)的模值校正，使得旋转归一化。

对于式(2-11)的迭代对ξ_i取不同的控制方式，可以得到不同的功能。

如已知向量$[r_0 \quad x_0]$和旋转角θ，要求旋转后的向量。ξ_i是这样控制的：

$$\begin{cases}\xi_i=Signz_i\\ z_0=\theta\\ z_{i+1}=z_i-\xi_i\theta_i\end{cases} \tag{2-13}$$

这样的迭代使

$$z_N=z_0-\sum_{i=0}^{N-1}\xi_i\theta_i\to 0$$

从而使式(2-13)迭代完成后旋转的角度为：

$$\sum_{i=0}^{N-1}\xi_i\theta_i = z_0 = \theta$$

其模经校正后即为所要的向量。

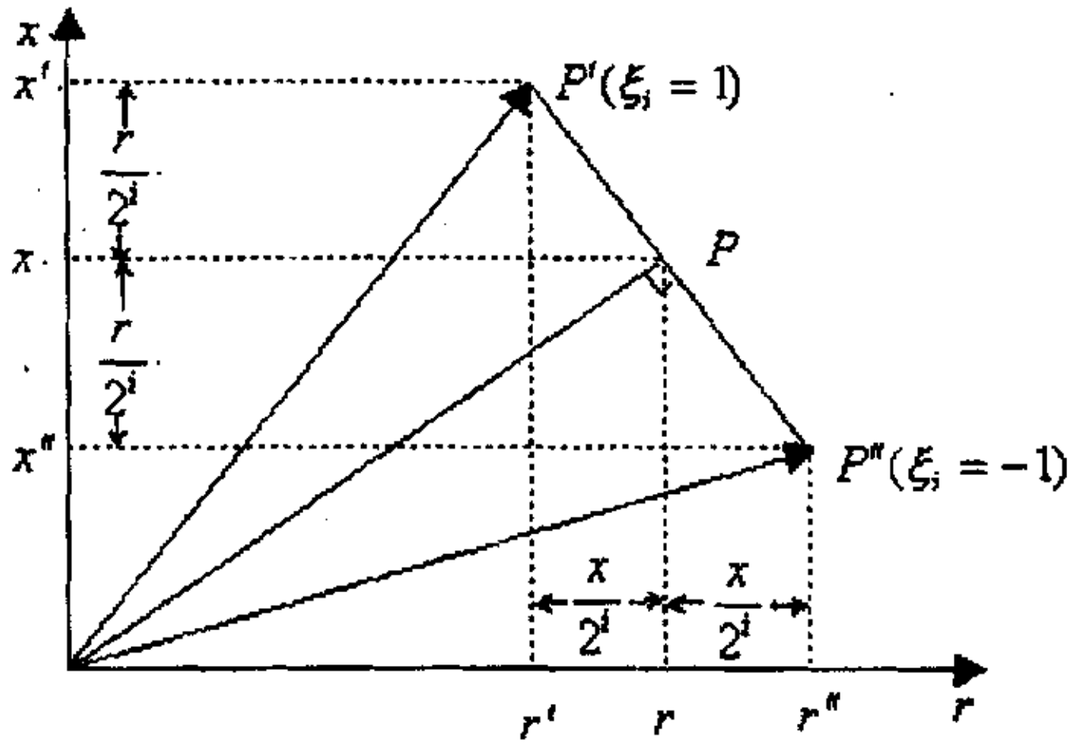


图 2.4 CORDIC 的几何解释

再如已知向量$[r_0 \quad x_0]$要求其模值和幅角，让向量旋转至横轴，则旋转后的横坐标即是模值，旋转的角度即是幅角。这样，CORDIC 迭代需保证$x_n \to 0$，这只需要取迭代中的ξ_i控制为：

$$\xi_i = -Signx_i \tag{2-14}$$

这样x_i正了就往负偏，负了就往正偏，而每次的偏移量越来越小，这就可以保证旋转逼近横轴。如果再加上迭代：

$$\begin{cases} z_{i+1} = z_i - \xi_i\theta_i \\ z_0 = 0 \end{cases}$$

则可的其幅角

$$\theta = -z_n = -\sum_{i=0}^{N-1}\xi_i\theta_i$$

这样，就代替了一般的用平方和开方求模值，用反三角函数求幅角的运算，显然简单多了。人们把它当作信号处理的算术单元[41]，广泛应用于信号处理各领域。诸如离散傅里叶变换[42][43]、语音信号分析[44]、矩阵分析[45][46]、三角函数发生器[47][48]等。与此同时，CORDIC 算法本身的硬件实现也得到不断

改进和发展。研制出了专门的 CORDIC 算法芯片[44][49]。

2.5.1 算法的 CORDIC 实现

递归最小二乘脉动阵列的边界单元和内部单元进行的运算是 Givens 旋转，末端单元进行的是乘法运算。首先来看边界单元和内部单元的 CORDIC 算法实现。

边界单元和内部单元进行的是下列运算：

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} r(1) & \cdots & r(k) & \cdots \\ x(1) & \cdots & x(k) & \cdots \end{bmatrix}\rightarrow\begin{bmatrix} r'(1) & \cdots & r'(k) & \cdots \\ 0 & \cdots & x'(k) & \cdots \end{bmatrix} \tag{2-15}$$

其中，$r(1),x(1),r'(1)$ 对应于边界单元的更新前后的对角元素和新进的数据；$r(k),r'(k),x(k),x'(k)$ 对应于内部单元的更新前后的对角元素和新进的数据。

式(2-15)中不包含计算残差所需的余弦函数的连乘运算。如果把乘余弦的运算看成旋转运算的结果，就可以用式(2-15)来实现它，在式(2-15)中加上一列$[0,\ \gamma_{in}]$，则运算后结果为：

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ \gamma_{in} \end{bmatrix}\rightarrow\begin{bmatrix} -\gamma_{in}s \\ \gamma_{in}c \end{bmatrix}\rightarrow\begin{bmatrix} d \\ \gamma_{out} \end{bmatrix} \tag{2-16}$$

这样就得到了计算残差所需的余弦函数的连乘运算：

$$\gamma_{out} \leftarrow c\gamma_{in}$$

于是，重点看式(2-15)的 CORDIC 算法实现。

首先考虑 $\beta=1$ 的情况，这时式(2-15)就是一个 Givens 旋转算子运算：

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} r(1) & \cdots & r(k) & \cdots \\ x(1) & \cdots & x(k) & \cdots \end{bmatrix}\rightarrow\begin{bmatrix} r'(1) & \cdots & r'(k) & \cdots \\ 0 & \cdots & x'(k) & \cdots \end{bmatrix} \tag{2-17}$$

它可以被看成是一系列式(2-10)的运算，因此可以用一系列的 CORDIC 迭代来实现。把$[r \quad x]$看作是二维向量，则向量$[r(1) \quad x(1)]$经 2-17 那样的 Givens 旋转后则转至横轴(此时，$x'(1)=0$)，$r'(1)$即其模值。这种旋转用 CORDIC 算法实现即相当于式(2-14)那种情况，类似的，取：

$$\xi_i = -Sign(x_i) \tag{2-18}$$

让$[r(1) \quad x(1)]$进行如下迭代：

$$\begin{cases} r' = r - \xi_i x 2^{-i} \\ x' = x + \xi_i r 2^{-i} \end{cases} \tag{2-19}$$

则经过 N 迭代，$x'_N \to 0$。

当然在这种迭代的同时，可以求出其旋转角 θ，然后再让其他向量以式(2-13)那样的方式旋转，可以预料，这样迭代的 ξ_i 控制序列将会和式(2-18)中的完全一致，因为他们代表完全一样的角度信息。事实上，由于旋转偏摆的角度序列 θ_i 都是固定的，所以旋转的角度唯一的由 ξ_i 序列决定。既然旋转角度相同，则它们的 ξ_i 序列就完全一样。因此诸向量 $[r(k) \quad x(k)]$ 的旋转可以以式(2-19)进行迭代，而且可以和向量 $[r(1) \quad x(1)]$ 同时进行，因为这时迭代只需 ξ_i 信息。于是这种旋转就可以撇开角度的迭代，只进行 r，x 的迭代，这就大大节省了硬件量。

这样各向量在式(2-18)控制方式下，进行式(2-19)的迭代，N 次迭代后的结果为：

$$\begin{bmatrix} r(1) & \cdots & r(k) & \cdots \\ x(1) & \cdots & x(k) & \cdots \end{bmatrix} \to \frac{1}{K_N} \begin{bmatrix} r'(1) & \cdots & r'(k) & \cdots \\ 0 & \cdots & x'(k) & \cdots \end{bmatrix} \tag{2-20}$$

再进行模校，去掉因子 $\frac{1}{K_N}$，就可以实现式(2-17)。

下面，讨论一下模校运算：

由式(2-12)可以看出，要使旋转归一化，去掉因子 $\frac{1}{K_N}$，采用如下的方式来进行：

$$\begin{cases} r' = r - \gamma_i r 2^{-i} \\ x' = x - \gamma_i x 2^{-i} \end{cases} \tag{2-21}$$

要求，

$$\prod_{i=1}^{m} (1 - \gamma_i 2^{-k_i}) = K_m \to K_N \tag{2-22}$$

其中 $\gamma_i \in \{1,-1\}$；$k_i \in \{0,1,\cdots N-1\}$，$m$ 为模校的迭代次数。

γ_i 和 k_i 是在以上的约束条件下使目标函数：

$$f(\gamma_i, k_i) = \left| K_m - K_N \right|$$

最小的那些数。(这些值可以在模校运算前得到)

这样，$[r \quad x]$在进行式(2-19)的旋转迭代后再进行式(2-21)的模校迭代，便可以实现式(2-17)。综合一下式(2-19)和(2-21)，可以得到如下的形式：

$$\begin{cases} r' = r \pm x(r)2^{-i} \\ x' = x \pm r(x)2^{-i} \end{cases}$$

对于$\beta \neq 1$的情况，对以上的运算过程进行改进，使得加滑动指数窗在模值校正过程中同时进行。

来看一下式(2-15)的运算过程，当第一组输入数据到来时的情况是这样的：

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & \cdots & 0 & \cdots \\ x_{11} & \cdots & x_{1K} & \cdots \end{bmatrix}$$

$$= \begin{bmatrix} \cos 90^0 & -\sin 90^0 \\ \sin 90^0 & \cos 90^0 \end{bmatrix}\begin{bmatrix} 0 & \cdots & 0 & \cdots \\ x_{11} & \cdots & x_{1K} & \cdots \end{bmatrix} \rightarrow \begin{bmatrix} x_{11} & \cdots & x_{1K} & \cdots \\ 0 & \cdots & 0 & \cdots \end{bmatrix} \quad (2\text{-}23)$$

这样，当第二组数据进来后，进行如下运算：

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_{11} & \cdots & x_{1K} & \cdots \\ x_{21} & \cdots & x_{2K} & \cdots \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} \beta x_{11} & \cdots & \beta x_{1K} & \cdots \\ x_{21} & \cdots & x_{2K} & \cdots \end{bmatrix} \quad (2\text{-}24)$$

于是

$$\theta = -Arc\tan(\frac{\beta x_{11}}{x_{21}}) \quad (2\text{-}25)$$

如果改变式(2-9)的运算次序，让运算次序变为：

$$\begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} r \\ x \end{bmatrix} \quad (2\text{-}26)$$

这样不但不会改变算法的性质，而且便于实现。按照式(2-26)的运算次序，当第一批数进入阵列时，运算即为：

$$\begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \cos 90^0 & -\sin 90^0 \\ \sin 90^0 & \cos 90^0 \end{bmatrix}\begin{bmatrix} 0 & \cdots & 0 & \cdots \\ x_{11} & \cdots & x_{1K} & \cdots \end{bmatrix} \rightarrow \begin{bmatrix} \beta x_{11} & \cdots & \beta x_{1K} & \cdots \\ 0 & \cdots & 0 & \cdots \end{bmatrix} \quad (2\text{-}27)$$

较式(2-23)的运算相比，在这次运算中前一时刻的数据就已经乘上了β。

当第二组数据进入后，则进行下列运算：

$$\begin{bmatrix}\beta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta' & -\sin\theta'\\ \sin\theta' & \cos\theta'\end{bmatrix}\begin{bmatrix}\beta x_{11} & \cdots & \beta x_{1K} & \cdots\\ x_{21} & \cdots & x_{2K} & \cdots\end{bmatrix} \tag{2-28}$$

这样，

$$\theta' = -Arc\tan(\frac{\beta x_{11}}{x_{21}}) = \theta \tag{2-29}$$

上式在没有乘β之前就完成了式(2-24)的运算，其旋转角也完全一样。而式(2-28)运算完后又将为下一次运算预先乘上了β。

这样虽然运算次序改变了，但每个旋转角并没有改变，只不过每一次运算都为下一次运算加好了指数窗，因此，整体的运算过程对算法的实质性内容并没有改变。

于是整个阵列的运算可以用式(2-26)的运算次序，而且在硬件实现上可以沿用$\beta=1$时的硬件结构。因为：

$$\begin{bmatrix}\beta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}c & -s\\ s & c\end{bmatrix}\begin{bmatrix}r\\ x\end{bmatrix}=\begin{bmatrix}\beta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}r'\\ x'\end{bmatrix}\begin{bmatrix}\beta r'\\ x'\end{bmatrix} \tag{2-30}$$

这样只不过在式(2-17)运算后，再对r乘上β，这个运算完全可以在对r的模校过程中同时进行，而且只需让对r的模校从式(2-21)中分离出来，对r进行另外的控制：

$$r' = r - \gamma_i' r 2^{-i}$$

使

$$\prod_{i=1}^{m}(1-\gamma_i 2^{-k_i}) = K_m' \to \beta K_N$$

这样在对r进行模校过程中同时乘上了β，于是式(2-30)也可实现了。

下面总结一下基于 QR 分解的递归最小二乘算法的 CORDIC 实现的过程。

边界单元和内部单元进行下列运算：

1) 矢量旋转：

$$\begin{cases} r_{i+1} = r_i - \xi_i x_i 2^{-i} \\ x_{i+1} = x_i + \xi_i r_i 2^{-i} \\ \xi_{i+1} = -sign(x_{i+1}) \end{cases} \tag{2-31}$$

$$(r_0 = r, x_0 = x)$$

2) 模值校正

$$\begin{cases} r' = r_N - \gamma_i r_N 2^{-i} \\ x' = x_N - \gamma_i x_N 2^{-i} \\ \gamma_i \in \{1,-1\} \end{cases} \tag{2-32}$$

$$\prod_{i=1}^{m}(1-\gamma_i 2^{-k_i}) = \beta K_N$$

根据式(2-31)和(2-32)可以得出 CORDIC 的硬件结构图(图 2.5)

末端单元进行乘法运算，得到残差。末端单元的设计过程将在第四章详细介绍。

2.6 本章小节

本章由最小二乘到最小二乘的 QR 分解解法再到递归最小二乘算法的脉动阵列，最后到宏单元的 CORDIC 结构，一步一步的阐明了论文所采用的算法的理论推导和初步的硬件构架。

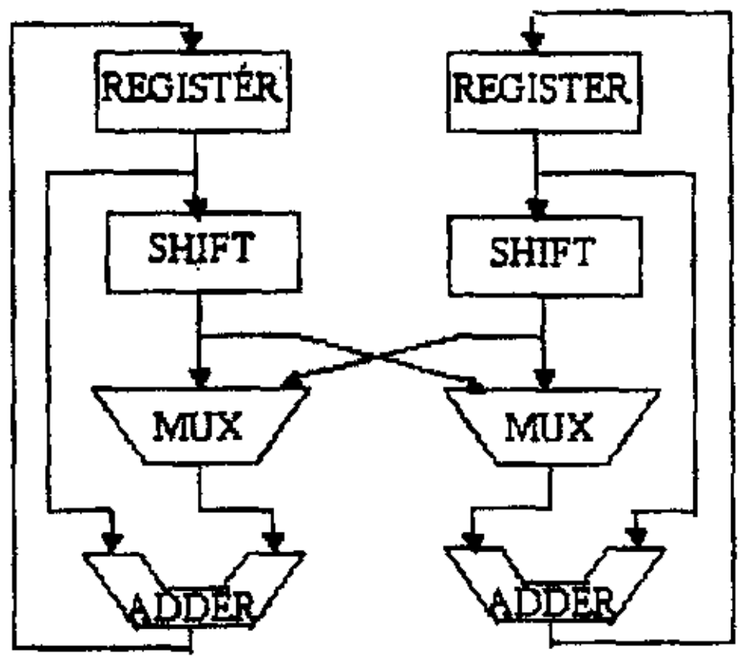


图 2.5 完成旋转与模校功能的 CORDIC 算法结构

第3章 系统构架的改进

3.1 超前处理技术

在这一节，介绍一种能够改进坐标旋转计算的技术—超前处理。超前处理技术通过在算法中引入并行机制，能够使串行的递归算法并发执行。超前处理技术是基于坐标旋转计算的，适合于应用 Givens 旋转的自适应阵列算法的流水执行。为了方便介绍此技术，在此节中，假定遗忘因子为 1。

3.1.1 超前处理技术的引入

在这一小节中，从块处理和递归的角度来介绍超前处理技术，这一技术可以分两步来完成：

1. 构造一个多快拍数据(数据块)更新的结构，数据块包含的快拍个数就是流水级数 M，也就是采样率加速的倍数。
2. 由一系列的 Givens 旋转来完成数据块的更新，先处理数据块中的元素，到数据块中只剩一个元素的时候，再根据上一时刻的三角阵对角线上的一个元素来处理数据块中的最后的一个元素。这样做的目的是为了使数据块更新同单组数据更新时的对角元素更新环具有相同的计算复杂度，同时可以消除不相邻数据处理单元之间的数据依赖性，方便进行流水。

来看一下图 3.1 所示的想要达到三级流水的块数据更新的形式，进行一系列的 Givens 旋转把 X(n-2)、X(n-1)、X(n)消掉，把 R(n-3)更新到 R(n)。这种更新形式是串行的，快拍被一行一行的处理，每一次处理都要对对角元

素 r 更新。信号流图如图 3.3(a)所示，可以看出，反馈环中的旋转的次数随着环中延迟元素的增大而线性增大，这样就不能够带来采样率的提高。

$$\begin{bmatrix} \neq & r & r & (n-3) \\ & r & r & (n-3) \\ & & r & (n-3) \\ \ast & x & x & (n-2) \\ x & x & x & (n-1) \\ x & x & x & (n) \end{bmatrix} \rightarrow \begin{bmatrix} r & r & r & (n-3) \\ & \neq & r & (n-3) \\ & & r & (n-3) \\ 0 & \ast & x & (n-2) \\ x & x & x & (n-1) \\ x & x & x & (n) \end{bmatrix} \rightarrow \begin{bmatrix} \neq & r & r & (n-3) \\ & r & r & (n-3) \\ & & \neq & (n-3) \\ 0 & 0 & \ast & (n-2) \\ x & x & x & (n-1) \\ x & x & x & (n) \end{bmatrix}$$

$$\begin{bmatrix} r & r & r & (n-3) \\ & r & r & (n-3) \\ & & r & (n-3) \\ 0 & 0 & 0 & (n-2) \\ 0 & 0 & 0 & (n-1) \\ 0 & 0 & 0 & (n) \end{bmatrix} \leftarrow \begin{bmatrix} r & r & r & (n-3) \\ & r & r & (n-3) \\ & & \neq & (n-3) \\ 0 & 0 & 0 & (n-2) \\ 0 & 0 & 0 & (n-1) \\ \ast & \ast & \ast & (n) \end{bmatrix} \leftarrow \begin{bmatrix} \neq & r & r & (n-3) \\ & \neq & r & (n-3) \\ & & \neq & (n-3) \\ 0 & 0 & 0 & (n-2) \\ \ast & \ast & \ast & (n-1) \\ x & \ast & x & (n) \end{bmatrix}$$

图 3.1 块数据更新的数据消除形式

超前处理技术如图 3.2 所示，快拍元素是一列一列的消除，对角元素 r 只有在最后一次才被更新。信号流图如图 3.3(b)所示，可以看出，对角元素更新环中的计算复杂度没有增加，但环中的时间延迟却提高到原来的 M 倍，这些时间延迟可以采用割集重定时技术[3]分布到更新环中，获得好的流水粒度。

下面，再从递归的角度来介绍超前处理技术。从前一章的论述中可以得到如下的 QR 递归的基本形式：

$$\begin{bmatrix} r(n) \\ 0 \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} r(n-1) \\ x(n) \end{bmatrix} \tag{3-1}$$

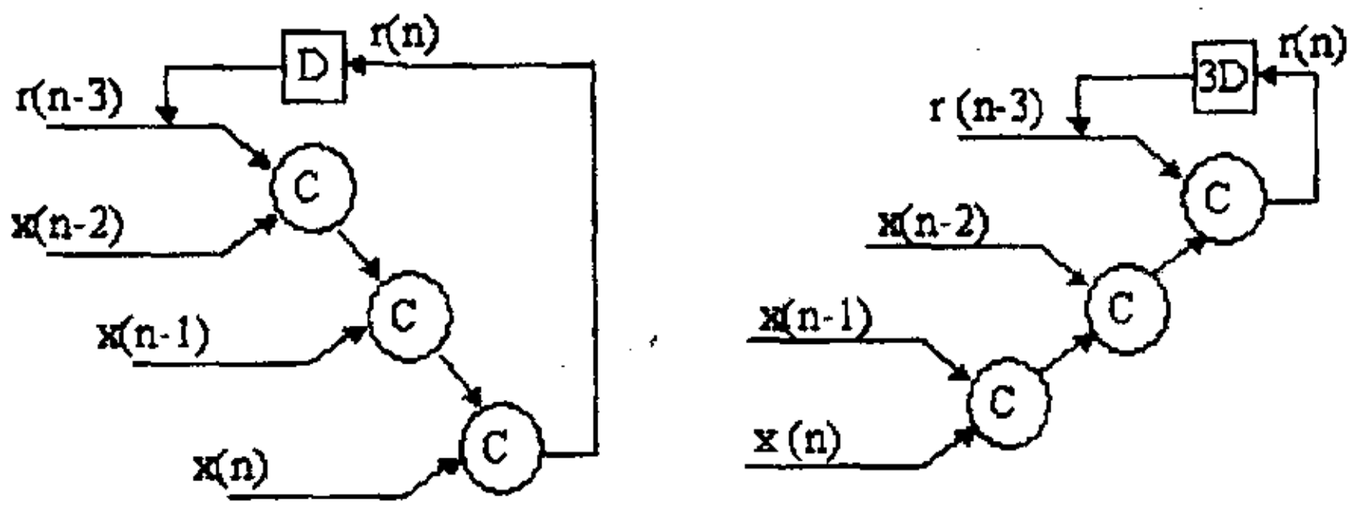
式中，$r(n)$ 和 $x(n)$ 分别对应图 2.4 中的边界元素和输入数据。类似的，我们还可以得到式(3-2)。

$$\begin{bmatrix} r(n) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & s_1 \\ 0 & 1 & 0 \\ -s_1 & 0 & c_1 \end{bmatrix} \begin{bmatrix} r(n-1) \\ 0 \\ x(n) \end{bmatrix} \tag{3-2}$$

$$\begin{bmatrix} r & r & r & (n-3) \\ & r & r & (n-3) \\ & & r & (n-3) \\ x & x & x & (n-2) \\ * & x & x & (n-1) \\ * & x & x & (n) \end{bmatrix} \rightarrow \begin{bmatrix} r & r & r & (n-3) \\ & r & r & (n-3) \\ & & r & (n-3) \\ * & x & x & (n-2) \\ * & x & x & (n-1) \\ 0 & x & x & (n) \end{bmatrix} \rightarrow \begin{bmatrix} * & r & r & (n-3) \\ & r & r & (n-3) \\ & & r & (n-3) \\ * & x & x & (n-2) \\ 0 & x & x & (n-1) \\ 0 & x & x & (n) \end{bmatrix}$$

$$\begin{bmatrix} r & r & r & (n-3) \\ & r & r & (n-3) \\ & & r & (n-3) \\ 0 & 0 & 0 & (n-2) \\ 0 & 0 & 0 & (n-1) \\ 0 & 0 & 0 & (n) \end{bmatrix} \leftarrow \begin{bmatrix} r & r & r & (n-3) \\ & r & r & (n-3) \\ & & * & (n-3) \\ 0 & 0 & * & (n-2) \\ 0 & 0 & * & (n-1) \\ 0 & 0 & * & (n) \end{bmatrix} \leftarrow \begin{bmatrix} r & r & r & (n-3) \\ & * & r & (n-3) \\ & & r & (n-3) \\ 0 & * & x & (n-2) \\ 0 & * & x & (n-1) \\ 0 & * & x & (n) \end{bmatrix}$$

图 3.2 超前处理技术的数据消除形式



(a) (b)

图 3.3 数据消除的信号流图 (a) 数据块更新 (b) 超前处理技术

由式(3-1)，可以得到：

$$\begin{bmatrix} r(n-1) \\ 0 \\ x(n) \end{bmatrix} = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r(n-2) \\ x(n-1) \\ x(n) \end{bmatrix} \tag{3-3}$$

将式(3-3)代入式(3-2)，可得：

$$\begin{bmatrix} r(n-1) \\ 0 \\ x(n) \end{bmatrix} = \begin{bmatrix} c_1 & 0 & s_1 \\ 0 & 1 & 0 \\ -s_1 & 0 & c_1 \end{bmatrix} \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r(n-2) \\ x(n-1) \\ x(n) \end{bmatrix} \tag{3-4}$$

依次类推，可以得到更多级的迭代形式：

$$\begin{bmatrix} r(n) \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & 0 & s_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_1 & 0 & 0 & c_1 \end{bmatrix} \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ 0 & 1 & 0 & 0 \\ -s_2 & 0 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & s_3 & 0 & 0 \\ -s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r(n-3) \\ x(n-2) \\ x(n-1) \\ x(n) \end{bmatrix} \tag{3-5}$$

图 3.3(a)是式(3-5)的信号流图，虽然反馈环中有三个延迟，但是环中计算的时间也是原来的三倍，因此并没有能够提高采样率。为了提高采样率，可以这样来考虑，改变原来消除元素的顺序，使输入数据消零的顺序变更为(3，4)，(2，3)，(1，2)，(i, j)表示式(3-5)中的行序号。这样，对角元素 r 只有在最后一次才被更新，得到如下所示的形式：

$$\begin{bmatrix} r(n) \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1' & s_1' & 0 & 0 \\ -s_1' & c_1' & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_2' & s_2' & 0 & 0 \\ -s_2' & c_2' & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_3' & s_3' \\ 0 & 0 & -s_3' & c_3' \end{bmatrix} \begin{bmatrix} r(n-3) \\ x(n-2) \\ x(n-1) \\ x(n) \end{bmatrix} \tag{3-6}$$

信号流图如图 3.3(b)所示，与由块处理的观点得出的结论一样。在没有增加环中计算时间的前提下，使环中的延迟由一个增加到了三个，把这三个延迟在环中重新分配，可以得到三级流水的形式。

3.1.2 超前处理技术中的并行机制

这一小节说明一下超前处理技术中的并行机制。先来看一下图 3.3(a)的数据更新的情况，它的更新进程的相关图如图 3.4 所示，图中的小圆圈表示 CORDIC 宏单元，k 方向的箭头表示时间前进的方向，箭头表示元素之间的相关性，例如，$r(n+1)$依赖于$r(n)$，$r(n)$依赖于$r(n-1)$，等等。就是这种相关性限制了数据更新的速度，也就降低了数据采样率。超前处理技术技术打破了这种相关性，把以前的相关图分成不相关的 M 个子图，既然这 M 个子相关图

彼此之间不相关，就可以并行执行了。这 M 个不相关的子图就是超前处理技

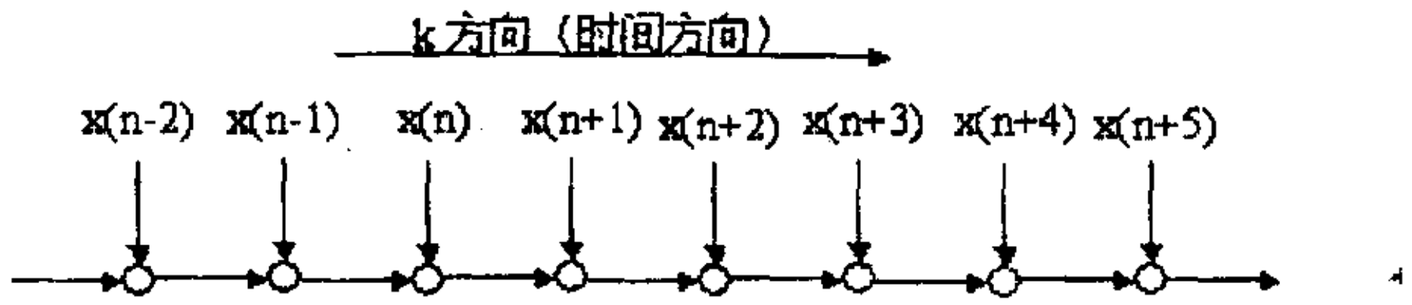


图 3.4 块数据更新进程的相关图

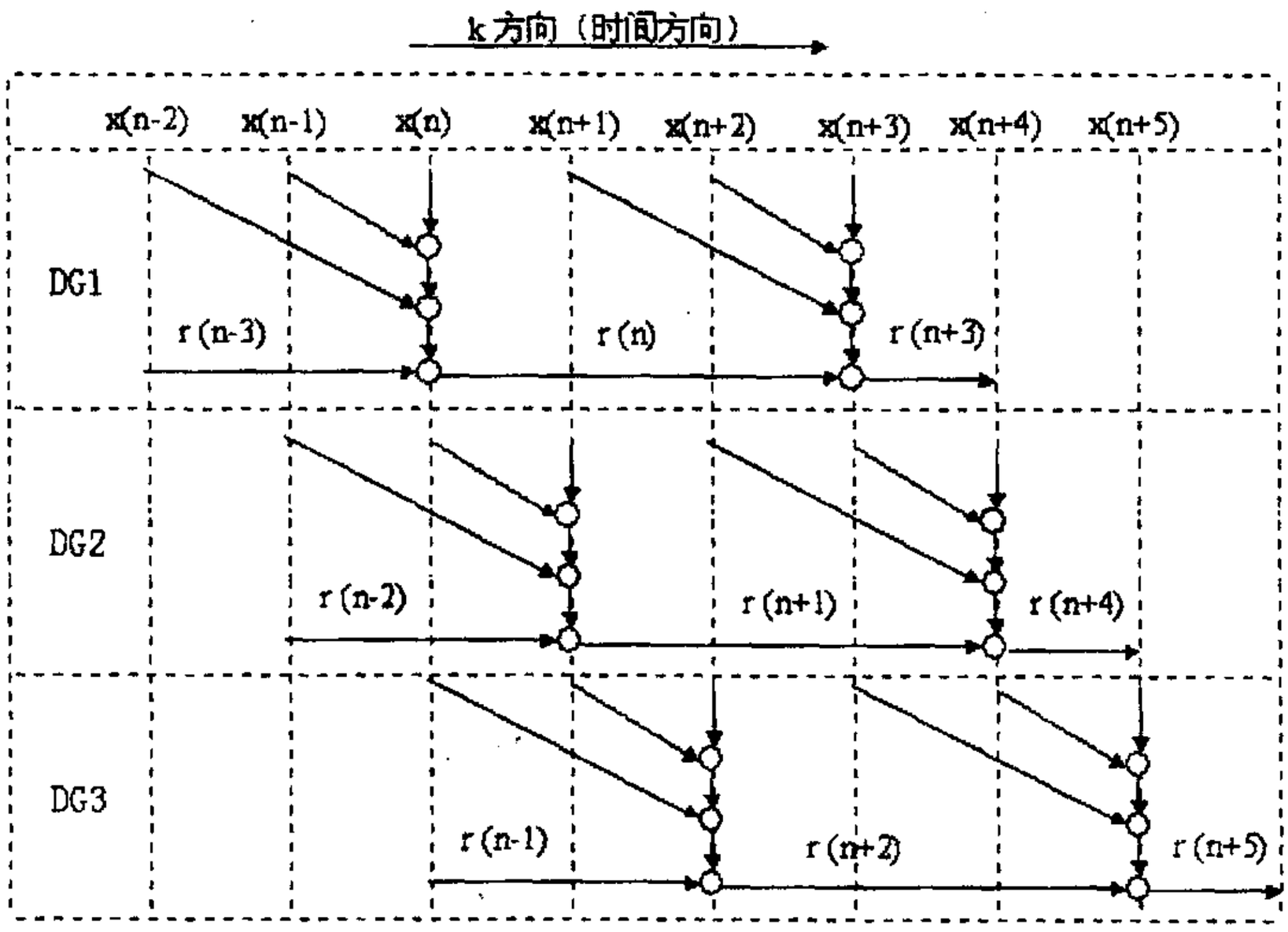


图 3.5 超前处理技术数据更新进程的相关图

术增加的并行机制。图 3.5 就是这 M 个子相关图(DG1、DG2、DG3)，$r(n)$ 的计算不再依赖于 $r(n-1)$，而是与 $r(n-3)$ 相关。同样的，$r(n+1)$ 依赖于 $r(n-2)$，$r(n+2)$ 依赖于 $r(n-1)$。一般的，为了实现 M 级流水结构，一次迭代中需要创建 M^2 个不相关的 CORDIC 运算，随着 M 的增大，越来越多的并行机制被建立，也就使采样率越来越高。

3.1.3 超前处理技术的实现

这一小节将阐述超前处理技术的流水实现，考虑图 3.5 中的三个子相关图，假如沿着 k 方向投影，就可以得到信号流图。既然三个子相关图彼此不相关，可以把他们投影到同一个 CORDIC 宏单元上，以流水交错[22]的模式来执行。这样，就得到了超前处理技术的流水实现形式(图 3.7)。串行的输入数据经延迟线转化成了并行数据，进入宏单元进行运算。系统的硬件量与流水级成线性关系，图 3.7 中的流水级为 3，即 M=3。

3.2 递归最小二乘算法的流水CORDIC实现

这一节，把超前处理技术应用到基于 CORDIC 的递归最小二乘算法中，得到了一个具有优良流水粒度的拓扑结构。这里，我们只考虑权向量隐式算法。

3.2.1 算法的流水实现

在前一章的式(2-8)，我们已经得到了 QRD-RLS 算法的 CORDIC 的实现形式，利用类似的方法，同时引入超前处理技术，来推导这种算法的具有优良流水粒度的硬件结构表达式。

对数据矩阵和参考向量组成的矩阵进行 QR 分解：

$$\begin{aligned} &Q(n-M)B(n-M)\begin{bmatrix} X(n-M) & y(n-M) \end{bmatrix} \\ &=\begin{bmatrix} R(n-M) & p(n-M) \\ O & v(n-M) \end{bmatrix} \end{aligned} \tag{3-7}$$

其中，$R(n-M)$ 是 $p\times p$ 维上三角阵，$p(n-M)$ 和 $v(n-M)$ 分别是 $p\times 1$ 和 $(n-M-p)\times 1$ 维向量，$B(n-M)$ 是 $(p+1)\times(p+1)$ 维遗忘因子矩阵。在 n 时刻，新进入的 M 组数据 $X_M(n)$ 和 $y_M(n)$ 有如下关系：

$$\begin{bmatrix} X(n) & y(n) \end{bmatrix}=\begin{bmatrix} X(n-M) & y(n-M) \\ X_M(n) & y_M(n) \end{bmatrix} \tag{3-8}$$

定义，

$$B'(n)=\begin{bmatrix}\beta^M B(n-M) & O\\ O & I_M\end{bmatrix}$$

$$Q'(n-M)=\begin{bmatrix}Q(n-M) & O\\ O & I_M\end{bmatrix}$$

因此，

$$Q'(n-M)B'(n)\begin{bmatrix}X(n) & y(n)\end{bmatrix}$$

$$=\begin{bmatrix}\beta^M R(n-M) & \beta^M p(n-M)\\ O & \beta^M v(n-M)\\ X_M^T(n) & y_M(n)\end{bmatrix} \tag{3-9}$$

用 $B'(n)$ 替代 $B(n)$，可以避免影响新进入的M组数据，由于 $B'(n)$ 中有 β^M，因此不会影响算法的收敛性能。

用正交矩阵来消除式(3-9)新进的 M 数据，

$$\begin{bmatrix}R(n) & p(n)\\ O & v(n)\\ O_{M\times p} & \alpha(n)\end{bmatrix}=Q(n)\begin{bmatrix}\beta^M R(n-M) & \beta^M p(n-M)\\ O & \beta^M v(n-M)\\ X_M^T(n) & y_M(n)\end{bmatrix} \tag{3-10}$$

由式(2-1)有，

$$e(n-M)=y(n-M)-X(n-M)w(n-M) \tag{3-11}$$

结合 3-7，有，

$$Q(n-M)B(n-M)e(n-M)$$

$$=\begin{bmatrix}p(n-M)\\ v(n-M)\end{bmatrix}-\begin{bmatrix}R(n-M)\\ O\end{bmatrix}w(n-M) \tag{3-12}$$

令，

$$\varepsilon(n-M)=Q(n-M)B(n-M)e(n-M)$$

$$e'_M(n) = y_M(n) - X_M^T w(n-M)$$

$$e_M(n) = y_M(n) - X_M^T w(n) \tag{3-13}$$

由式(2-1)可以看出，$e_M(n)$的最后一个元素是所需要的n时刻的残差$e(n)$。由式(3-12)和(3-13)可以得出：

$$\begin{bmatrix} \beta^M \varepsilon(n-M) \\ e'_M(n) \end{bmatrix} = \begin{bmatrix} \beta^M p(n-M) \\ \beta^M v(n-M) \\ y_M(n) \end{bmatrix} - \begin{bmatrix} \beta^M R(n-M) \\ O \\ X_M^T(n) \end{bmatrix} w(n-M) \tag{3-14}$$

正交矩阵作用于式(3-14)两边，结合式(3-10)，有

$$Q(n)\begin{bmatrix} \beta^M \varepsilon(n-M) \\ e_M(n) \end{bmatrix} = \begin{bmatrix} p(n) \\ v(n) \\ \alpha(n) \end{bmatrix} - \begin{bmatrix} R(n) \\ O \\ O \end{bmatrix} w(n) \tag{3-15}$$

消除输入数据矩阵$X_M(n)$以后，$w(n-M)$更新为$w(n)$。相应的，$e'_M(n)$变成了$e_M(n)$。把$Q(n)$移到等式的右边，并考虑到$Q(n)$的正交性，可得：

$$\begin{bmatrix} \beta^M \varepsilon(n-M) \\ e_M(n) \end{bmatrix} = Q^T(n)\begin{bmatrix} p(n) - R(n)w(n) \\ v(n) \\ \alpha(n) \end{bmatrix} \tag{3-16}$$

既然最优权向量$w(n)$满足关系式$p(n) - R(n)w(n) = 0_p$，因此式(3-16)可以简化为：

$$\begin{bmatrix} \beta^M \varepsilon(n-M) \\ e_M(n) \end{bmatrix} = Q^T(n)\begin{bmatrix} 0_p \\ v(n) \\ \alpha(n) \end{bmatrix} \tag{3-17}$$

考虑到的$e_M(n)$的最后一个元素是所需要的n时刻的残差$e(n)$，可以得到：

$$e(n)=\begin{bmatrix}0_{n-M}^T & \delta_M^T\end{bmatrix}Q^T(n)\begin{bmatrix}0_p\\ v(n)\\ \alpha(n)\end{bmatrix}$$

$$=\begin{bmatrix}0_{n-M}^T & \delta_M^T\end{bmatrix}Q^T(n)\begin{bmatrix}O_{(n-m)\times M}\\ I_M\end{bmatrix}\alpha(n) \qquad (3\text{-}18)$$

由于$Q^T(n)$只利用向量$\begin{bmatrix}0_p^T & v^T(n) & \alpha^T(n)\end{bmatrix}$中的$\alpha(n)$和$0_p$，由此推出了式(3-18)的第二个等式。对式(3-18)进一步变形，可以得到：

$$e(n)=\alpha^T(n)\begin{bmatrix}O_{M\times(n-M)} & I_M\end{bmatrix}\left(Q(n)\begin{bmatrix}0_{n-M}\\ \delta_M\end{bmatrix}\right)$$

$$=\alpha^T(n)\begin{bmatrix}O_{M\times(n-M)} & I_M\end{bmatrix}\begin{bmatrix}s(n)\\ \gamma(n)\end{bmatrix}$$

$$=\alpha^T(n)\gamma(n) \qquad (3\text{-}19)$$

综上，得到了 QRD-RLS 的具有优良流水粒度的硬件表达式(式 3-20 和 3-21)和系统结构(图 3.7)。

$$\begin{bmatrix}R(n) & y_p(n) & s(n)\\ o_{M\times p} & \alpha_n & \gamma_p\end{bmatrix}=Q(n)\begin{bmatrix}\beta^M R(n-M) & \beta^M y_p(n-M) & 0\\ X_M(n) & y_M(n) & \delta_M\end{bmatrix} \qquad (3\text{-}20)$$

其中，$X_M(n)$是$M\times p$维矩阵，表示新近的M个快拍数据；$y_M(n)$是M维向量，表示新近的M个期望信号的采样；δ_M为M维常向量，$\delta_M=\begin{bmatrix}0 & \cdots & 0 & 1\end{bmatrix}$。残差向量表示为：

$$e(n)=\gamma_p^T\alpha_{n+1} \qquad (3\text{-}21)$$

式(3-20)中的$Q(n)$与式(2-8)中的$Q'(n)$都是用来消除新近到达的数据的，不同之处在于，式(2-8)中的$Q'(n)$消除的是一个快拍的采样数据，由p个Givens 旋转组成，而式(3-20)中的$Q(n)$消除的是由 M 个快拍的数据块，由$M\times p$个 Givens 旋转组成，按图 3.4 所示的顺序来消除数据，获得更新了的三角阵以及α_{n+1}和γ_p。

依照式(3-20)和(3-21)进行 MATLAB 仿真(图 3.6),得到了与图 2.2 所示的相同的算法性能。

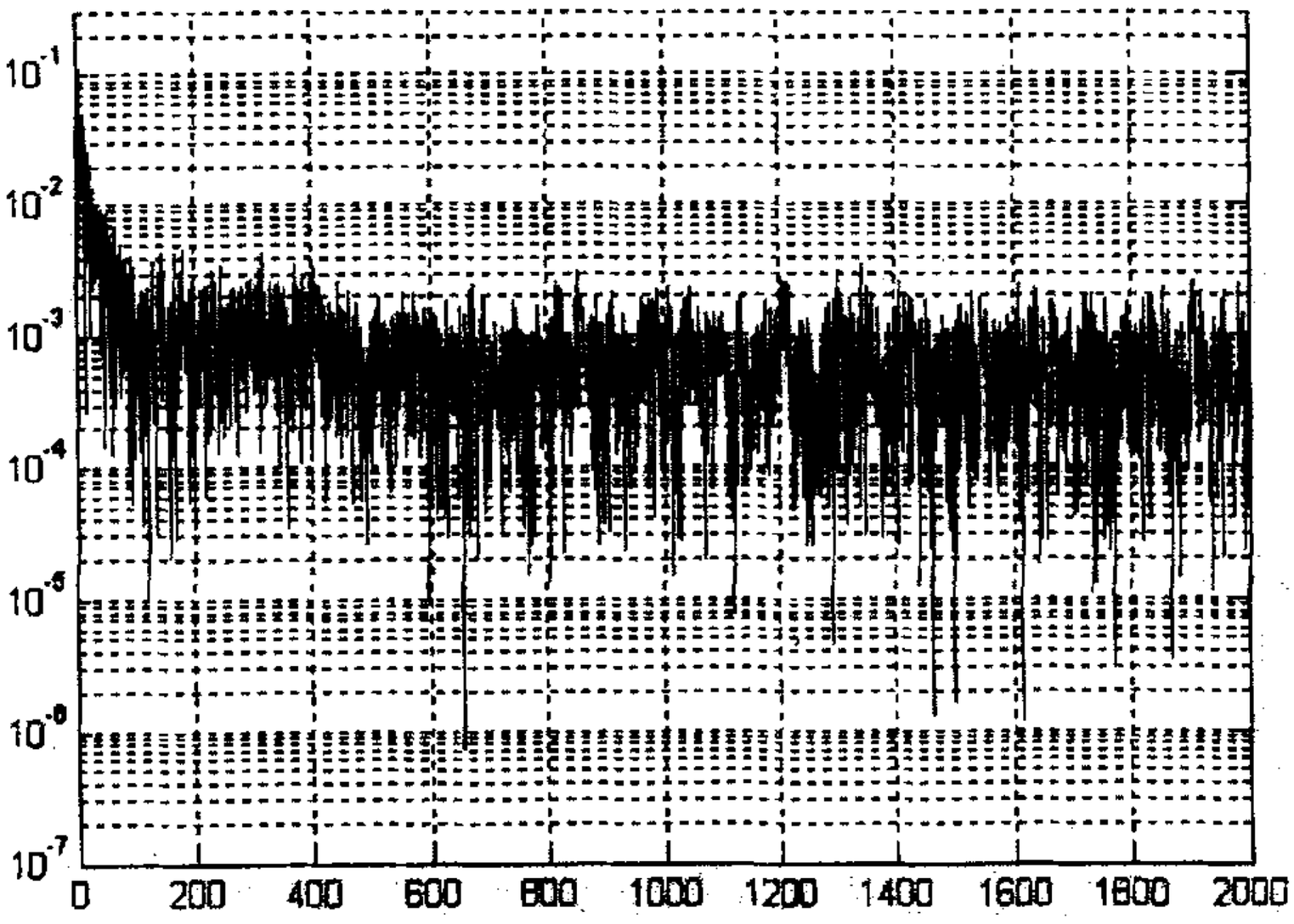


图 3.6 采用超前处理技术的 QRD-RLS 残差曲线

图 3.7 与图 2.4 中宏单元是相同的，只不过图 3.7 中的宏单元是几个的组合。系统整体的 CORDIC 宏单元数量为 $M(p+2)^2/2$，其中，M 是流水级 ，p 是阵元个数。

3.2.2 稳定性分析

因为 QR 分解通常由一系列的正交变换完成，所以基于 QR 分解的算法都具有较好的数值稳定性，就是说，运用此类算法的系统可以用较短的字长来达到不错的性能。由以上的论述可以看到，超前处理技术只涉及到了正交变换，并没有破坏算法的正交性，也就是说，文章所阐述的流水算法依然具有良好的数值特性。

3.3 本章小节

本章介绍了超前处理技术，阐述了此项技术中的并行机制，并将此技术

应用到了基于 QR 分解的递归最小二乘算法中，使得实现此算法的系统具有良好的流水粒度，在不改变算法性能的基础上提高了系统的采样速率。

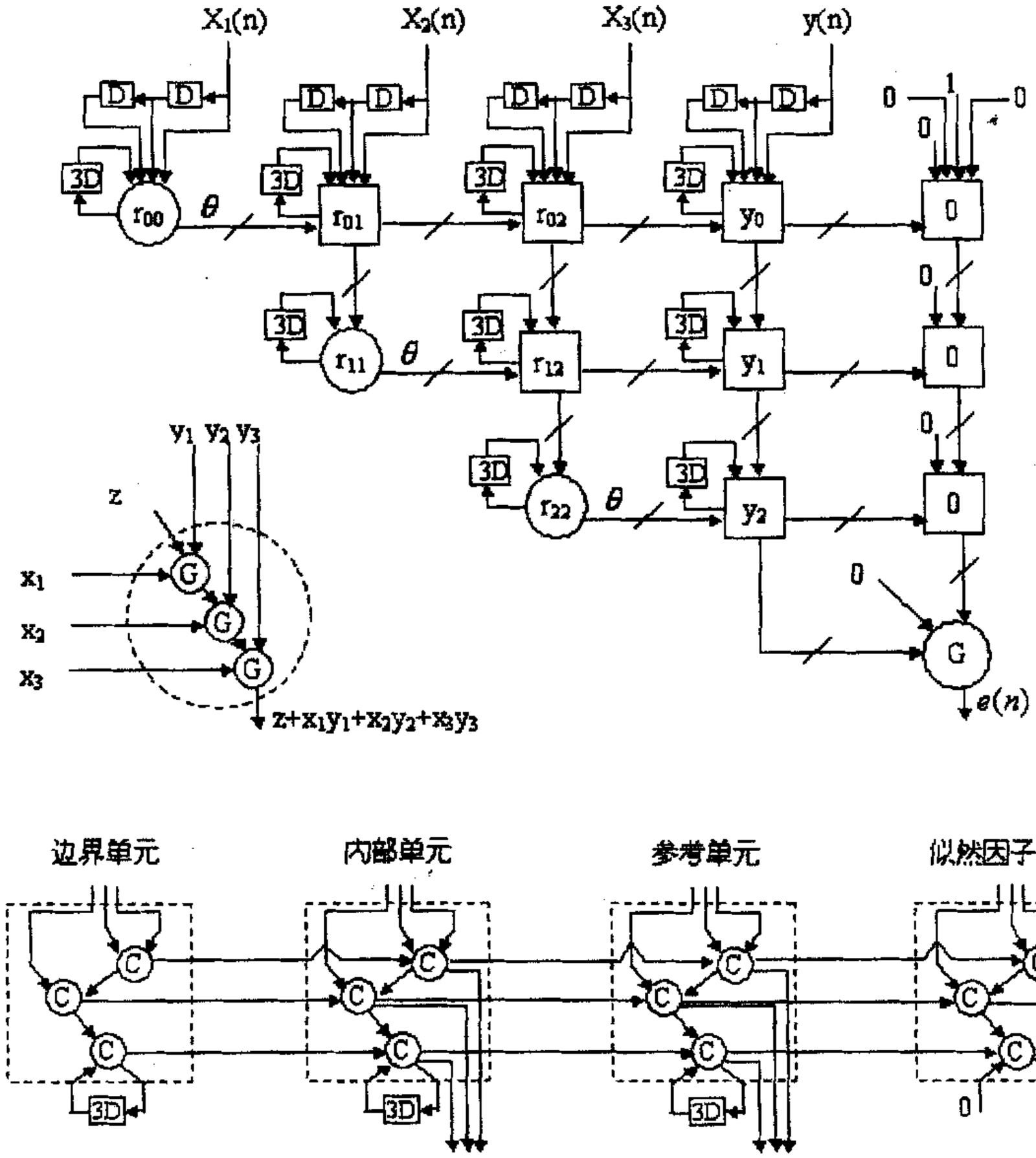


图 3.7 基于 QRD-RLS 的 CORDIC 的流水结构

第4章 高速实时硬件系统的设计与实现

4.1 系统论述

文章的目的是设计一个高速实时的自适应抗干扰阵列系统，算法采用基于 QR 分解的递归最小二乘，拓扑结构采用脉动阵列，宏单元采用 CORDIC，为了提高采样率，采用了能够增加系统流水级的超前处理技术。

系统的全部时间都用来运算，因此要选择适于运算的集成器件来构造系统。DSP 具有专用的算术指令，而且运算速度很快，但是 DSP 是单处理器结构，构成此系统需要多片 DSP。系统具有高度的并行流水性，由许多运算宏单元组成，而且宏单元中只有加法和移位运算，由此看来，不适宜采用 DSP 构造系统。FPGA 是一个很好的选择，他没有固定的处理器结构，允许构造任意的系统结构。

下面的一节，将阐述系统的数据流级设计，此设计可以直接下载到 FPGA 中完成系统。

4.2 系统设计

系统采用定点运算、16 位有符号数，CORDIC 的移位量为 0—15 位(旋转角误差不会超过 $Arc\tan 2^{-15}$)。

4.2.1 各宏单元构架

整个系统的宏单元包括边界单元、内部单元和末端单元。边界单元和内部单元的结构及运算过程基本相似，内部单元除了加减控制来自边界单元，其他与边界单元完全一样。根据式(2-8)和图 2.4 可以得到图 4.1 所示的边界单元和内部单元的硬件结构。

其中，CLK 为时钟信号，ACLR 为寄存器异步置零，T 为外部数据/内部反馈数据选择控制(T=1，选择外部数据；T=0，选择内部反馈数据)，X 为旋转

和模校选择信号(X=1，进行旋转；X=0，进行模校)，S 为右移位量(0—15bit)信号，CON1 和 CON2 为输出的加减控制信号，此信号被传到内部单元，使内部单元随着边界单元一起旋转。

下面，说明一下边界单元的运行过程：首先，在 T 的控制下，外部数据进入寄存器，在接下来的 22 个时钟周期里，单元不接收外部数据，只是内部数据在循环。然后，在 X 的控制下，按照 S 给出的移位量，进行交叉的移位和加法运算，经过 16 个周期的运算，X 寄存器变成了零(误差为$1/2^{15}$)，R 寄

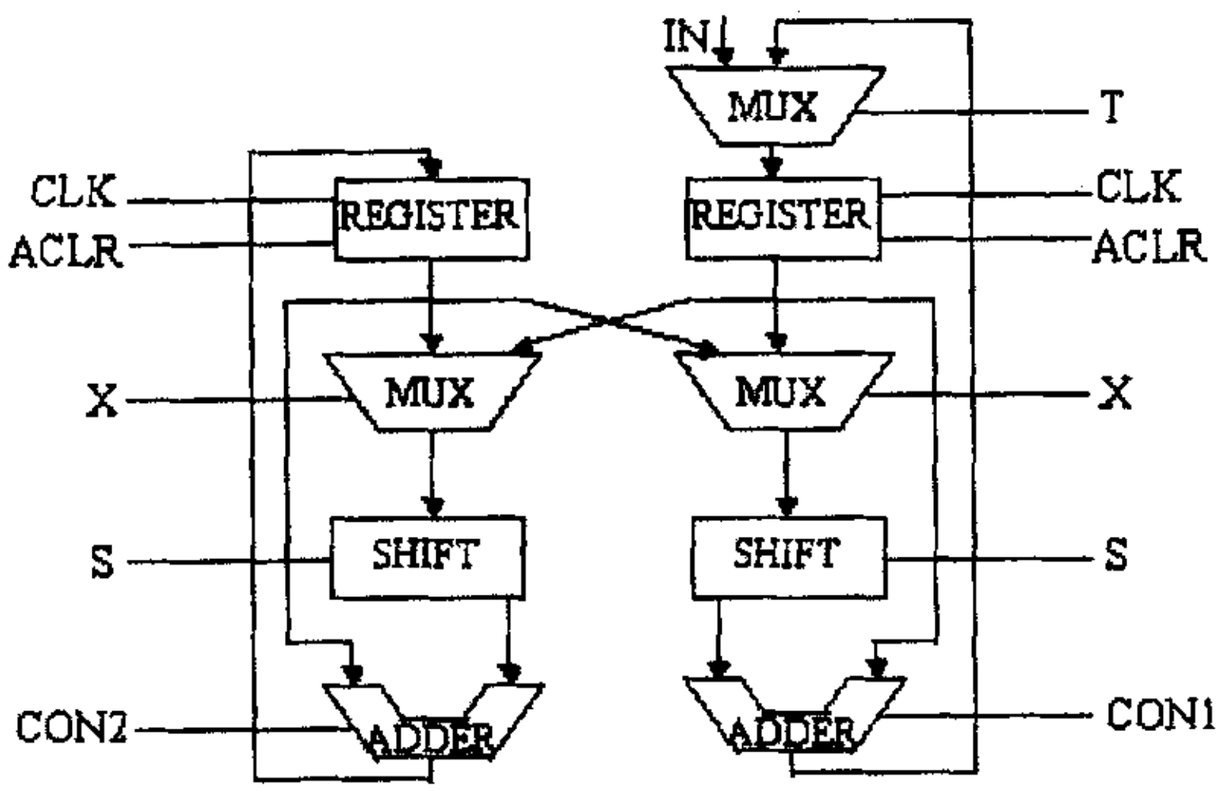


图 4.1 边界单元和内部单元的硬件结构

存器中的数据是带有一个系数的模值，接下来的模校过程就是为了把这个系数去掉。最后，X 变成 0，进行模校过程，按照 S 给出提前计算好的移位量，X 和 R 进行自身的移位加运算，经过 6 个时钟周期的运算，R 寄存器中的数据变成了向量$[X \quad R]$的模。

在图 4.1 中最主要的控制是两个加法器的加减控制，由式(2-32)可以得出，在模校过程中都是减法运算，可以由 X 来充当此过程的加减控制信号，在旋转过程中的加减运算由数据 X 的符号来确定，数据 X 选用由符号数表示，因此，此过程中的加减控制可以由数据 X 的最高位来充当。如是产生了图 4.1 中的加减控制电路。

末端单元要完成的运算是：

$$e_{n+1} = \gamma_p \alpha_{n+1} \quad , \quad \gamma_p = \prod_{i=1}^{p} \cos\theta_i \le 1$$

显然这是一个乘法器，下面看一下如何用 CORDIC 来完成乘法运算：

本文中采用的 CORDIC 的基本表示式：

$$\begin{cases} r_{i+1} = r_i - \xi_i x_i 2^{-i} \\ x_{i+1} = x_i + \xi_i r_i 2^{-i} \\ \xi_{i+1} = -sign(x_{i+1}) \end{cases}$$

前两个等式可以变形为以下形式：

$$x_N = x_0 + \sum_{i=0}^{N-1} r_i \xi_i 2^{-i} \tag{4-1}$$

若 r_i 为一常数，且 x_0 为 0，则可以得到两个数相乘的形式，其中另一个数为 $\sum_{i=0}^{N-1} \xi_i 2^{-i}$，$\sum_{i=0}^{N-1} \xi_i 2^{-i}$ 有一定的取值范围，这正好与 γ_p 相匹配。可以建立两个线性旋转[4]来完成这一乘法，其中一个线性旋转通过把 z 变成零来得到 γ_p 的 $\sum_{i=0}^{N-1} \xi_i 2^{-i}$ 表示形式，同时把 ξ_i 传递到另一个线性旋转，其实就是把 γ_p 传递过去，这样就在另一个线性旋转的过程中完成了 γ_p 与 α_{n+1} 的乘法。有以下表示式：

$$\begin{cases} w_{i+1} = w_i = 1 \\ z_{i+1} = z_i - \xi_i w_i 2^{-i} \\ \xi_{i+1} = sign(z_{i+1}) \end{cases} \tag{4-2a}$$

$$\begin{cases} u_{i+1} = u_i = u_0 \\ v_{i+1} = v_i + \xi_i u_i 2^{-i} \end{cases} \tag{4-2b}$$

若 $z_0 = \gamma_p$，$u_0 = \alpha_{n+1}$，按照式(3-1)推导，可以得出：

$$v_N \approx \gamma_p \alpha_{n+1} = e_{n+1}$$

依照式(3-2a)和(3-2b)可以得到如图 4.2 所示的硬件结构。

图中的各个信号与图 4.1 中的信号是相同的。需要说明的是，V 寄存器的初始值是 0，W 寄存器的值始终是 1，这个 1 可以用任意值来表示，但必须同系统中其他地方的 1 统一起来。

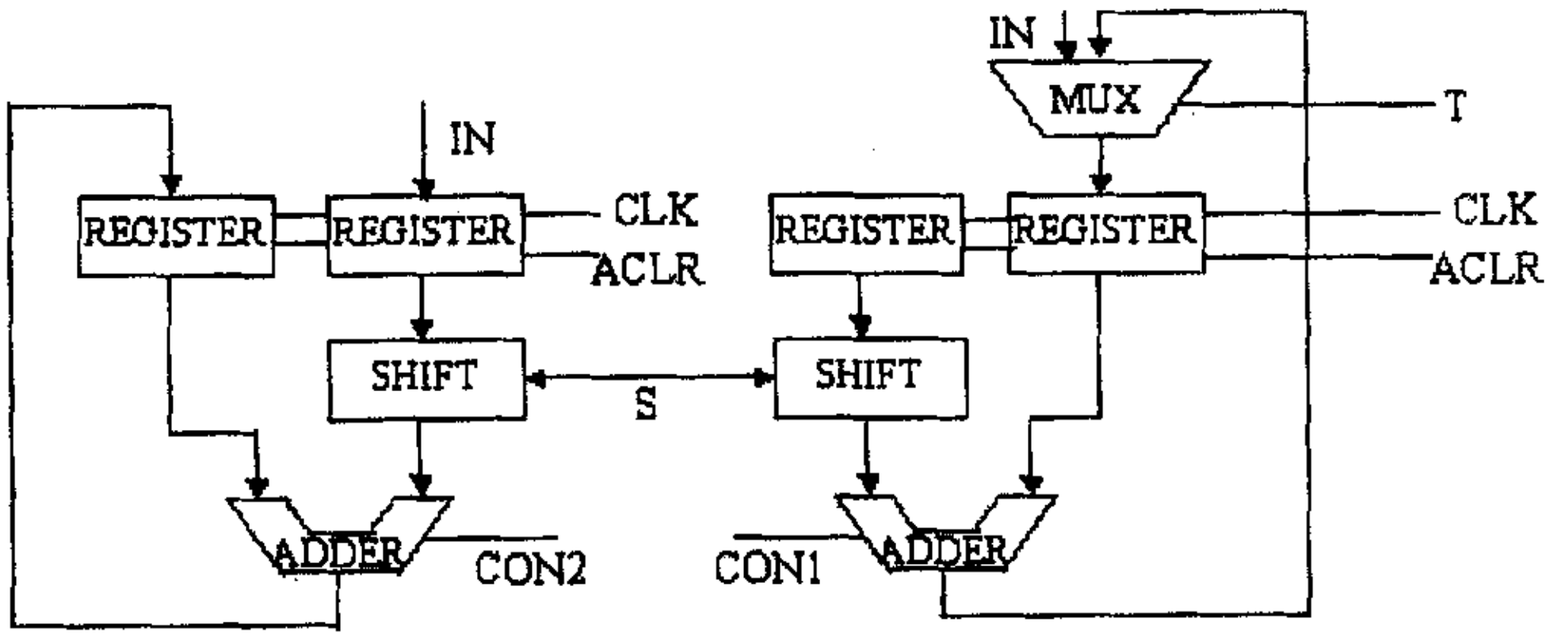


图 4.2 末端单元的硬件结构

由式(4-2)，末端单元只需要 16 个时钟周期就完成了乘法运算，下面的 6 个周期末端单元要保持其数据，这可以通过对末端单元的寄存器加一个使能信号来完成。

4.2.2 逻辑电路的设计

系统采用的是 FPGA，所谓逻辑电路的设计其实就是系统各个模块的数据流级或者模块级描述。宏单元主要由寄存器、移位器、加法器和数据选择器组成，另外，还有提供各种控制信号的控制模块。寄存器、加法器和数据选择器选择的是 EDA 设计软件提供的宏模块，在此就不再累述。

1. 移位器

移位器采用桶形移位器，这种移位器以增加硬件量为代价，提高移位速度，它能够在一个时钟周期内移完所需的位数，而且移位的速度与所移位的量没有关系。这很好的匹配 CORDIC 运算，这里选用的是一个 16 位的可右移 0—15 位的移位器。经过挑选，选择了如下的方案(distence 为移位量)：

```
IF distence(0)='1' THEN
   ssbit:=rzero(0) & ssbit(15 downto 1);
ELSE
   ssbit:=ssbit;
END IF;
IF distence(1)='1' THEN
```

```
  ssbit:=rzero(1 downto 0) & ssbit(15 downto 2);
ELSE
  ssbit:=ssbit;
END IF;
IF distence(2)='1' THEN
  ssbit:=rzero(3 downto 0) & ssbit(15 downto 4);
ELSE
  ssbit:=ssbit;
END IF;
IF distence(3)='1' THEN
  ssbit:=rzero(7 downto 0) & ssbit(15 downto 8);
ELSE
  ssbit:=ssbit;
END IF;
```

系统中选用的是有符号数，要进行有符号数的移位运算，要进行如下考虑：

```
rzero :=(15 downto 0 => data(15));
```

另外，当输入为全 1 的时候，右移位的结果应该是 0，因此，要对输出作如下处理：

```
IF ssbit="1111111111111111" THEN
   result <="0000000000000000";
ELSE
   result<=ssbit;
END IF;
```

2. 控制模块

控制模块的主要作用是产生外部数据/内部数据选择信号 T、旋转/模校控制信号 X、移位量 S 和末端单元寄存器的使能信号(EN)。在系统时钟的同步下，按表 4-1 所示的顺序产生各个信号。

表 4-1 控制信号时序列表

CLK 上升沿	T	X	S	EN
1	1	X	XXXX	X
2	0	1	0000	1
3	0	1	0001	1
4	0	1	0010	1
5	0	1	0011	1
6	0	1	0100	1
7	0	1	0101	1
8	0	1	0110	1
9	0	1	0111	1
10	0	1	1000	1
11	0	1	1001	1
12	0	1	1010	1
13	0	1	1011	1
14	0	1	1100	1
15	0	1	1101	1
16	0	1	1110	1
17	0	1	1111	1
18	0	0	0011	0
19	0	0	0011	0
20	0	0	0011	0
21	0	0	0100	0
22	0	0	0101	0
23	0	0	1001	0

由表 4-1 可以看出，23 个时钟周期构成一个循环，18-23 时钟周期完成模校过程，其中滑动指数窗的改变正是通过在模校过程中对移位控制序列编程来达到。当然先要算出移位序列，即 S。为此编了一个程序，该程序附于

下面。

```
% this is the program to find gi in the equation
% Kn=|| (1-gi*2^(-i))
clear;clc;
p=8; % precision
q=4; % revises
bita=1; % convergence factor
% get Kn=|| (1/sqrt(1+2^(-2*i)))( 0<= i <= p-1 )
m=1;
for i = 0:(p-1)
    t=2^(-2*i);
    m=m*sqrt(1+t);
end
Kn=bita/m;
% get gi
i = 0:(p-1);
x = 1-2-^(-i);
smin=10;
n=ones(1,q)+1;
for i = 1:p^q
    s=1;
    for j=1:q
        s=s*x(n(j));
    end
    sabs=abs(s-Kn);
    smin=min(sabs,smin);
    if sabs==smin
        Knee=smin;
Kne=s;
g=n;
```

```
end
for k=1:q
    if n(k) < p
        n(k)=n(k)+1;
        for w=1:k-1
            n(w)=2;
            end
            break
        end
    end
end
```

4.2.3 各宏单元的实现

有了以上各模块，依照 4-2-1 节中的各宏单元构架，就可以得到各宏单元的基于 FPGA 的实现形式。需要考虑两个问题

1. 边界单元的加减控制

由式(2-31)，旋转过程中的加减控制信号是 x 的符号，系统采用的是有符号数，因此旋转过程中 x 与 r 的加法器的加减控制可以由 x 的最高位(MSBx)来充当，取反可以得到 r 与 x 加法器的加减控制信号。

由式(2-32)，模校过程中一直是减法运算，可以由旋转/模校控制信号 X 来充当。因此，整个过程中的加减控制由(MSBx AND X)和(NOT MSBx AND X)来完成。

2. 末端单元的加减控制

由式(3-2)，图 4.2 的右边部分通过线性旋转把 z 变成 0，把这

个过程中的加减控制信号取反后传递给左边部分，其实就是把右边部分的 z 传递给了左边部分，产生 z 与 u 的积。因此末端单元的加减控制可以由 z 的最高位(MSBz)和(NOT MSBz)来充当。

运用 EDA 软件对其进行仿真和时间分析，其结果表现出了良好的性能。图 4.3-4.6 分别是边界单元和末端单元的 FPGA 的实现形式和仿真波形。经过时间分析，各个宏单元可以运行在 72MHz，因此系统可以达到 3MHz 的采样速

率。

图 4.4(a)显示的是边界单元旋转完成后的状态，X 的初始值为 99(b0000000001100011)，R 的初始值为 0。经过 16 个时钟周期的旋转 Xout 变成了-1(b1111111111111111)，Rout 变成了 161(b0000000010100001)。

图 4.4(b)显示的是边界单元模校完成后的状态，经过 6 个时钟周期的模校，Rout 由 161(b0000000010100001) 变成了 100(b0000000001100100)。

图 4.6 显示的是末端单元经过 16 个时钟周期的移位和加法后的结果。用 16384(b0100000000000000)表示 1，则图中的 Datayin 和 DataXin 的乘运算就是 DataXin 乘2^{-7}，结果应该为 b101010，而 Uvaddout 为 b110010,此结果可以接受。

4.2.4 系统的实现

用以上几小节设计的边界单元、内部单元和末端单元，依照图 3.7 的拓扑结构，就可以构成整个系统。

要考虑整个系统的时间协调:边界单元和内部单元需要 22 个时钟周期完成一次循环，旋转角，也就是加减控制信号要在一个宏单元完成一次循环后才可以传递到下一个宏单元。因此，需要加 22 位的串行移位器来协调时间。

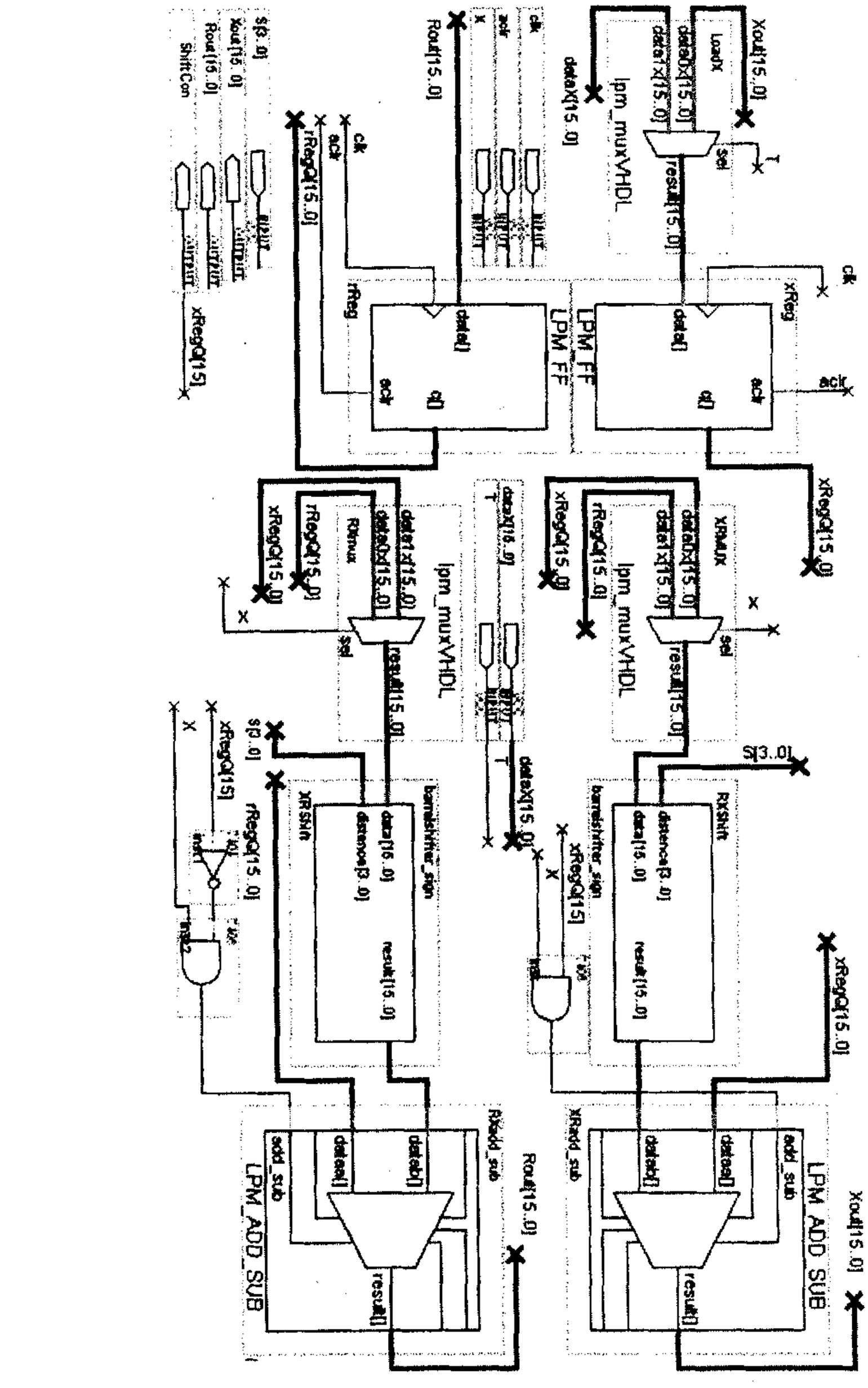
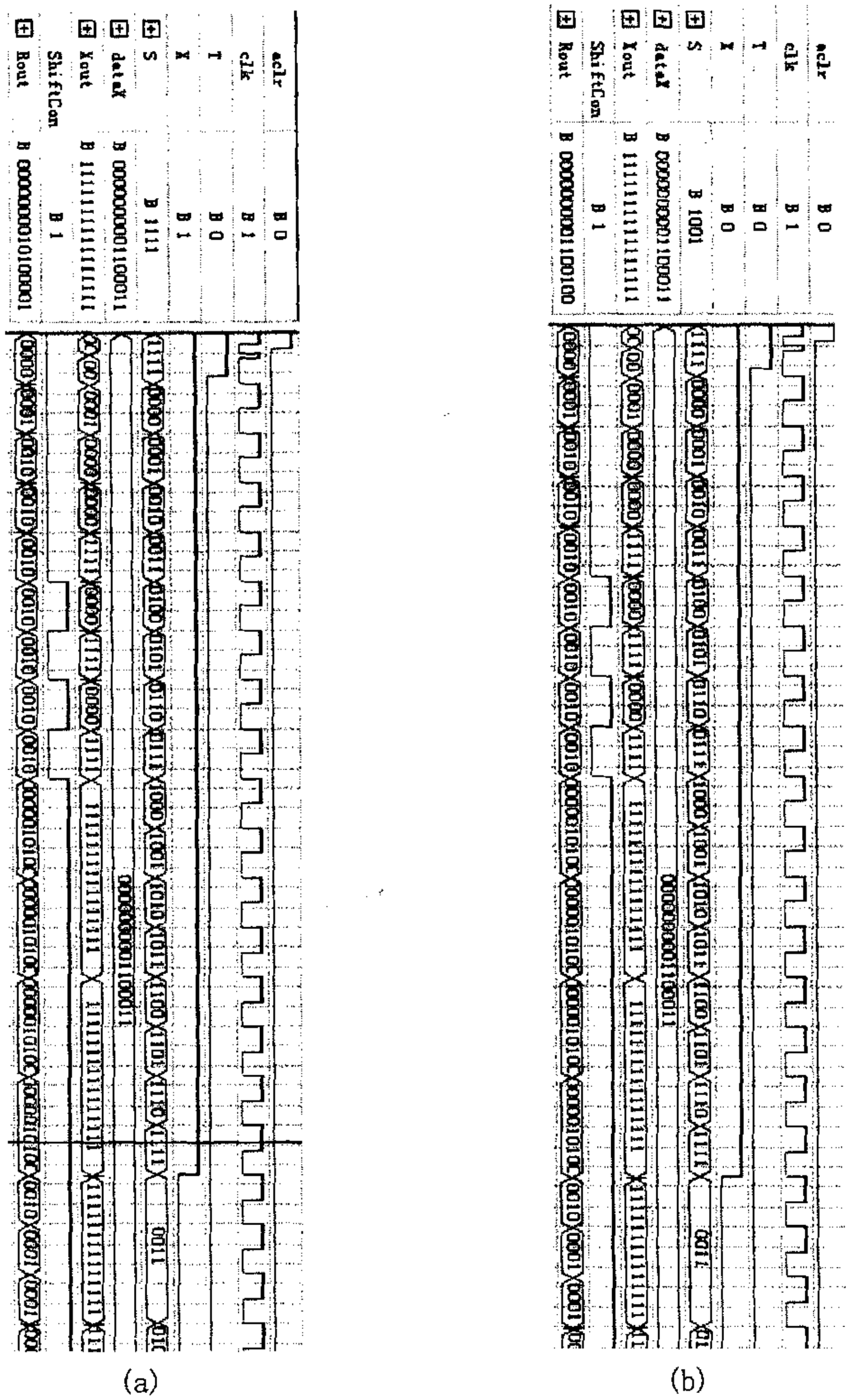


图 4.3 边界单元的 FPGA 实现形式



(a) (b)

图 4.4 边界单元的仿真波形

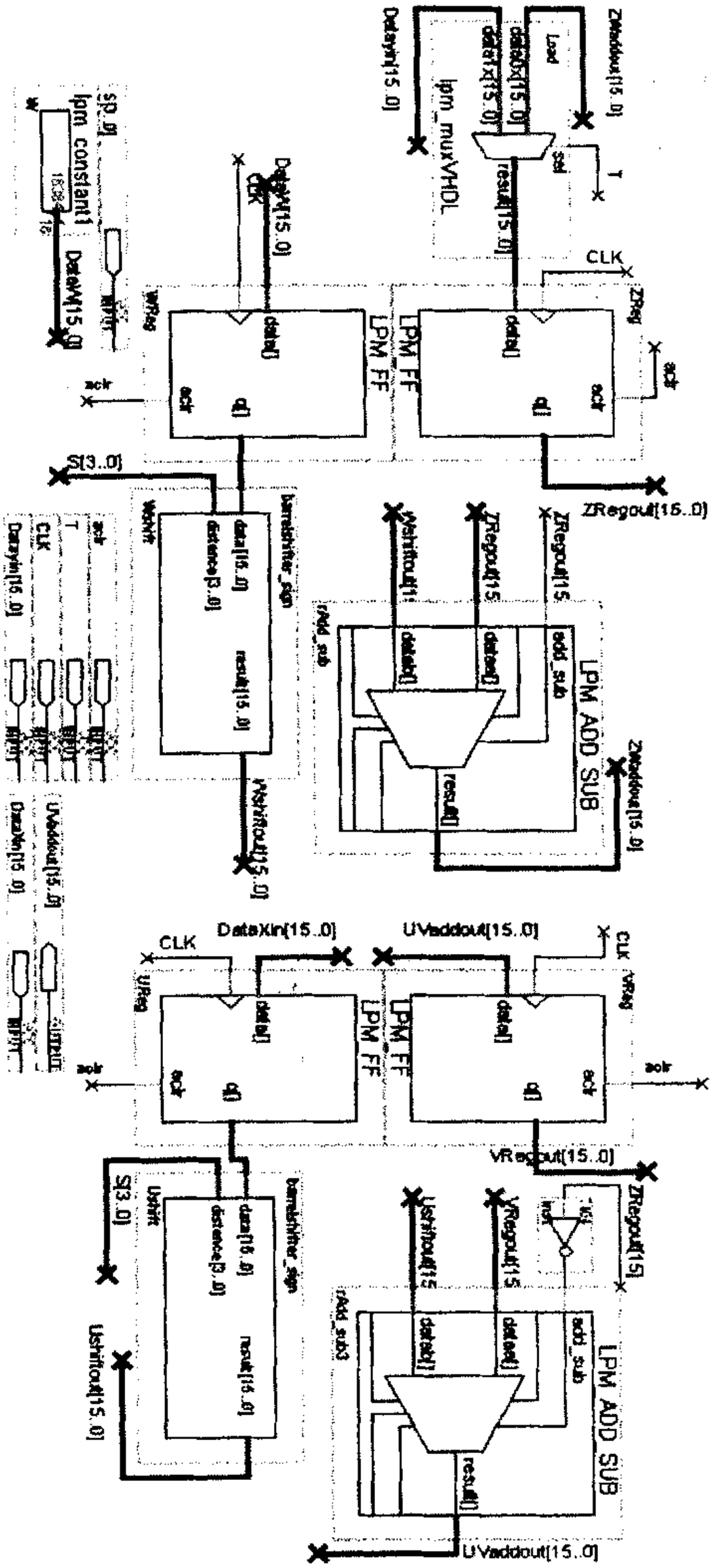


图 4.5 末端单元的 FPGA 实现形式

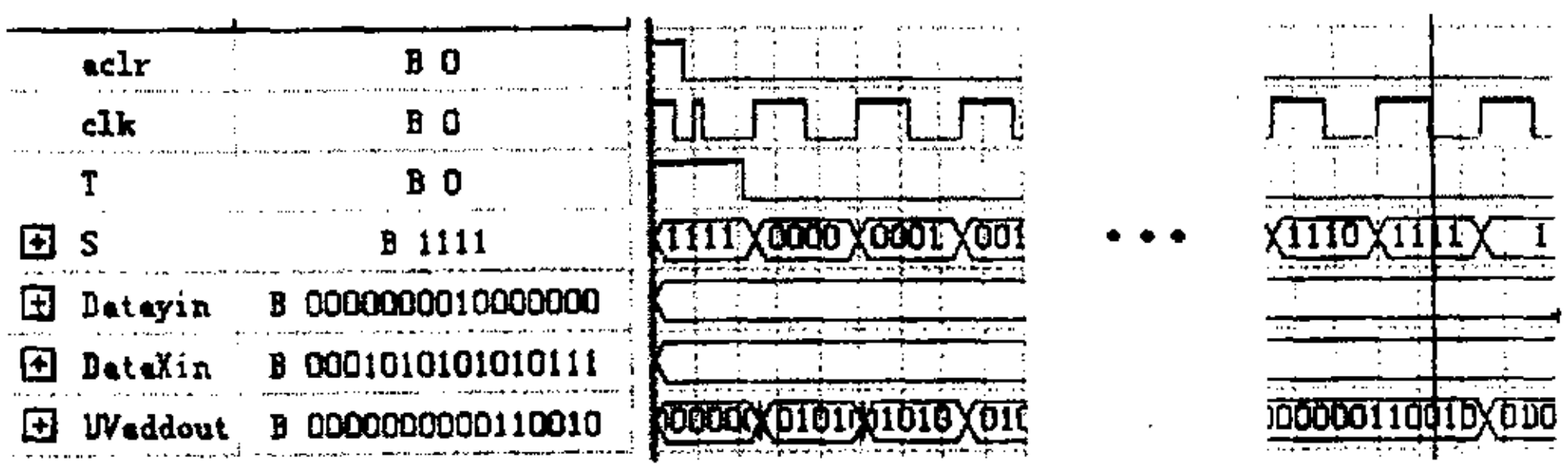


图 4.6 末端单元的仿真波形

4.3 系统硬件平台

4.3.1 芯片选型

系统由许多 CORDIC 宏单元以脉动阵列的形式组合而成，这就要求 FPGA 要有足够的逻辑单元(LE)。系统要求提供很高的时钟，单靠晶振很难达到，这样就需要倍频模块，为了提高系统的稳定性，最好是倍频模块与逻辑单元集成在一起。系统的输入信号是一个高速序列的数字信号，需要有一定的存储空间。集合以上的要求，在考虑价格因素后，选用了 ALTERA 公司的 Cyclone 系列 FPGA。每个边界单元和内部单元都需要 200 个左右的 LE 来完成，如果阵元数为 4，根据硬件量公式 $M(p+2)^2/2$ 可以计算出系统需要的总的 LE 个数为 10800。因此，选用 Cyclone 系列器件—EP1C12，该器件特点如下所示：

1. 12060 个 LE
2. 239616 个 RAM 位
3. 支持价格低廉的串行配置器件
4. 支持 LVTTL, LVCMOS, SSTL-2, and SSTL-3 接口标准
5. 支持 66MHz，32bit PCI 标准
6. 具有 2 个锁相环路，可以用来倍频

4.3.2 硬件电路设计

EP1C12 的外围硬件电路由五部分组成，电源模块、输出显示模块、输入模块、串口模块和配置模块。输出显示模块和输入模块是供调试时使用的，由数码管、发光二极管、拨码开关和外围电路构成，比较简单，不做累述。

串口模块提供一个与 PC 机的通信链路，可以把计算的结果传送到 PC 机供分析，也可以接受 PC 机传过来的数据，此模块供系统功能扩展用，在此不做详细叙述。下面将详细介绍电源模块和配置模块。

4.3.1.1 电源模块设计

电源模块的作用是为晶振和串口驱动芯片提供+5V 电源；为串行配置器件和 EP1C12 的 I/O 口提供+3-3V 电源；为 EP1C12 的内核和锁相环路(PLL)提供+1-5V 的电源。这是一个多电压系统，不同的电压间需要匹配，EP1C12 兼容 1-5V、1-8V、2-5V、3-3V 电压，EP1C12 可以驱动 5-0V 的 LVTTL 器件，但不能驱动 5-0V 的 LVCMOS 器件，当输出高电位为 5-0V 的器件作为 EP1C12 的输入时要在器件之间串入一个 165 欧姆的电阻来降低 EP1C12 的输入电压[50]。

Cyclone 器件的锁相环路是嵌入在数字集成电路中的模拟部分，为了避免数字集成电路中产生的干扰影响锁相环路的稳定性，数字部分与模拟部分的电源和地是严格分开的。即使不使用器件中的锁相环路，其电源引脚 Vcca 也必须接 1-5V 电源，而且 Vcca 必须同 Cyclone 器件中的电源隔离，也必须同电路板上的其他电源隔离。可以有几种方法来隔离 Vcca：

A. 一般的，既有模拟电路又有数字电路的电路板上要有各自的电源敷铜，在这样的电路板上，可以把 Vcca 接到模拟电源的铺铜上。
B. 电路板只有数字电路，没有模拟电源的敷铜，这样的话，可以为锁相环路的 Vcca 建立一个电源岛，这个岛的绝缘边界要大于 25mil。
C. 电路板面积的限制，以上两种方案不能实行的话，可以采用比较宽一些的 Vcca 电源走线，宽度要大于 20mil。

无论采用哪种方案，都要采用如图 4.7 所示去耦电路对 Vcca 进行滤波。在电源进入电路板的地方接入一个磁铢和一个 10uF 的钽电容，选择在 50MHz 频率或是更高频率的地方表现出高电抗性的磁铢。另外，还要在尽可能靠近 Cyclone 器件的地方并行接入 0.1uF 和 0.001uF 的陶瓷电容对 Vcca 去耦。锁相环路的地可以接在电路板的数字地上。

4.3.1.2 配置模块设计

配置电路完成对 Cyclone 器件的配置，电路分两部分，一部分执行主动(AS)配置模式，另一部分执行 JATG 配置模式。

1.主动配置模式

在这种模式中，采用串行配置器件。配置期间，Cyclone 器件从串口读取数据来配置静态随机存储器(SRAM)单元。配置器件有四针接口：串行时钟输入、串行数据输出、AS 数据输入和片选信号。这四个信号与 Cyclone 器件的连接如图 4.8 所示。

串行配置器件在掉电的时候仍然能够存储数据，可以通过 AS 编程接口使用下载电缆对其进行在系统编程。编程期间，下载电缆把 nCE 拉高来阻止 Cyclone 器件对 AS 接口的访问，Cyclone 器件的 nCONFIG 也被置成低电平。编程完成后，下载电缆释放 nCE 和 nCONFIG，允许上拉电阻和下拉电阻驱动 VCC 和 GND。图 4.9 表明了下载电缆与串行配置器件的连接情况。

2. JATG 配置模式

JATG 是为边界扫描测试(BST)而开发的，BST 可以在很密的电路板对各个部分进行有效的测试，BST 可以在器件工作的时候不用借助物理探针和任何的功能数据就能探测到管脚的连接情况。也可以借助 JATG 把配置数据植入 Cyclone 器件。JATG 配置指令具有最高优先级，在其他非 JATG 配置指令执行的时候运行 JATG 配置，JATG 配置将首先被执行。在 JATG 配置期间，配置数据通过下载电缆下载到器件中，类似于在系统编程。下载电缆与 Cyclone 器件管脚的连接如图 4.10 所示。如果只是采用 JATG 配置模式，nCONFIG 要连接到 Vcc，MSEL0 和 MSEL1 要连接到 GND，DATA0 和 DCLK 要上拉到高电平或下拉到低电平。

这样就完成了硬件平台的设计，图 4.11 是硬件平台的原理图。

4.4 本章小结

对第三章所述系统进行设计，首先对各个宏单元进行整体构架，然后给出了宏单元中各模块的数据流级描述，最后由各个宏单元按照第三章的结构

搭建整个系统。本章还进行了基于以上系统的硬件平台的设计。

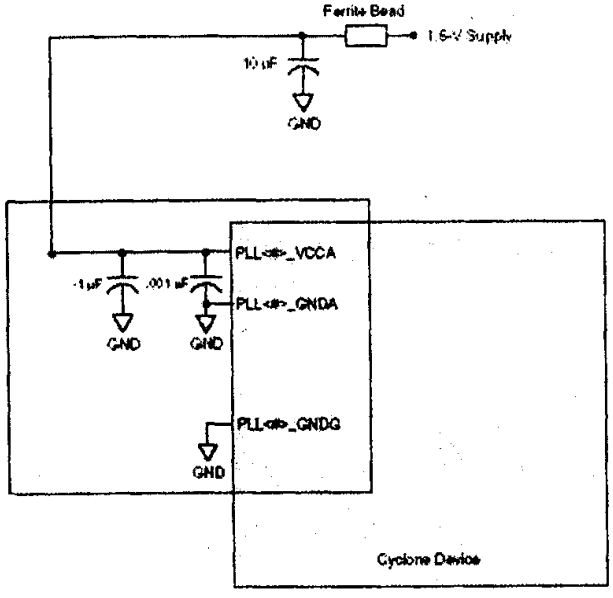


图 4.7 锁相环路的电源去耦电路

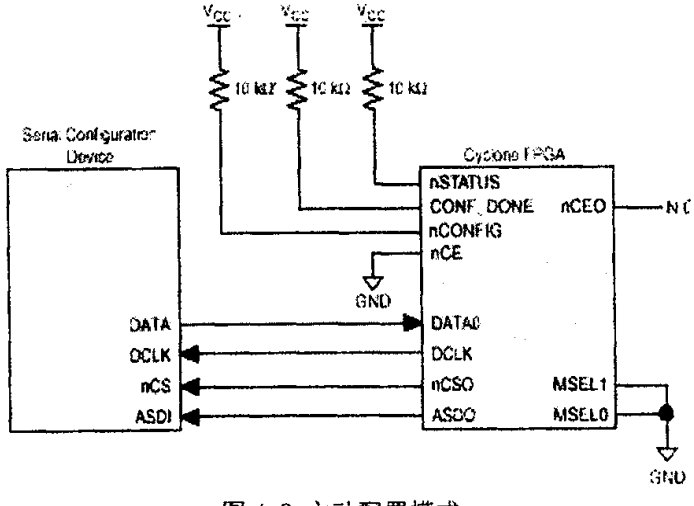


图 4.8 主动配置模式

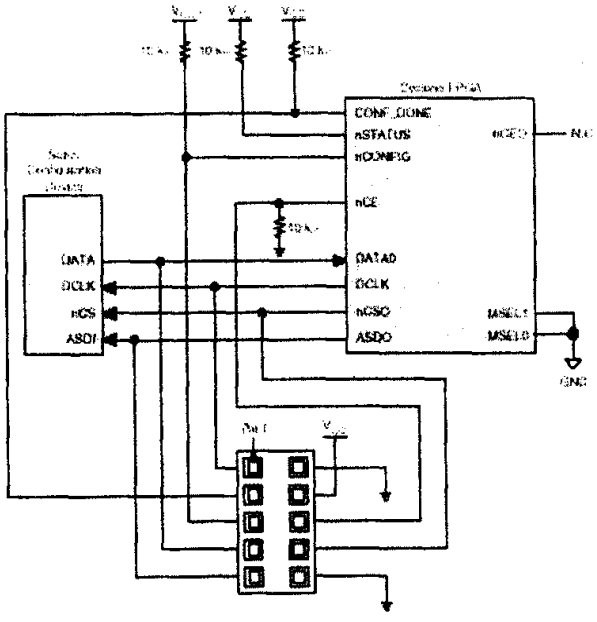


图 4.9 可在系统编程的主动配置模式

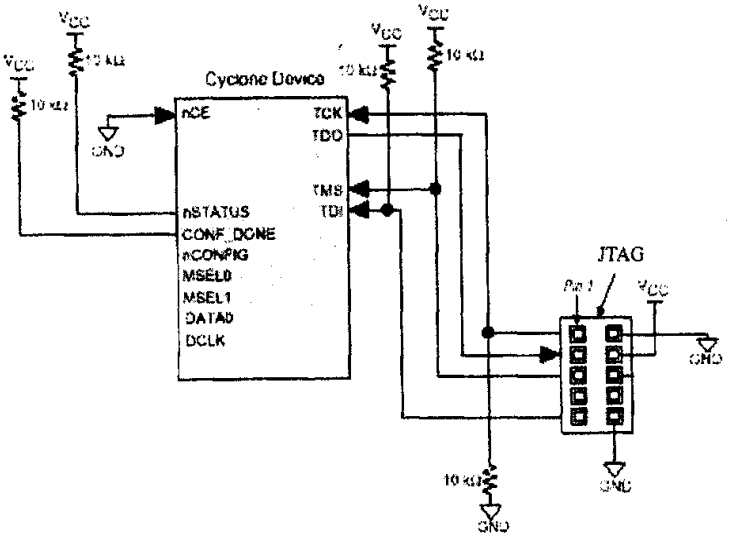


图 4.10 JATG 配置模式

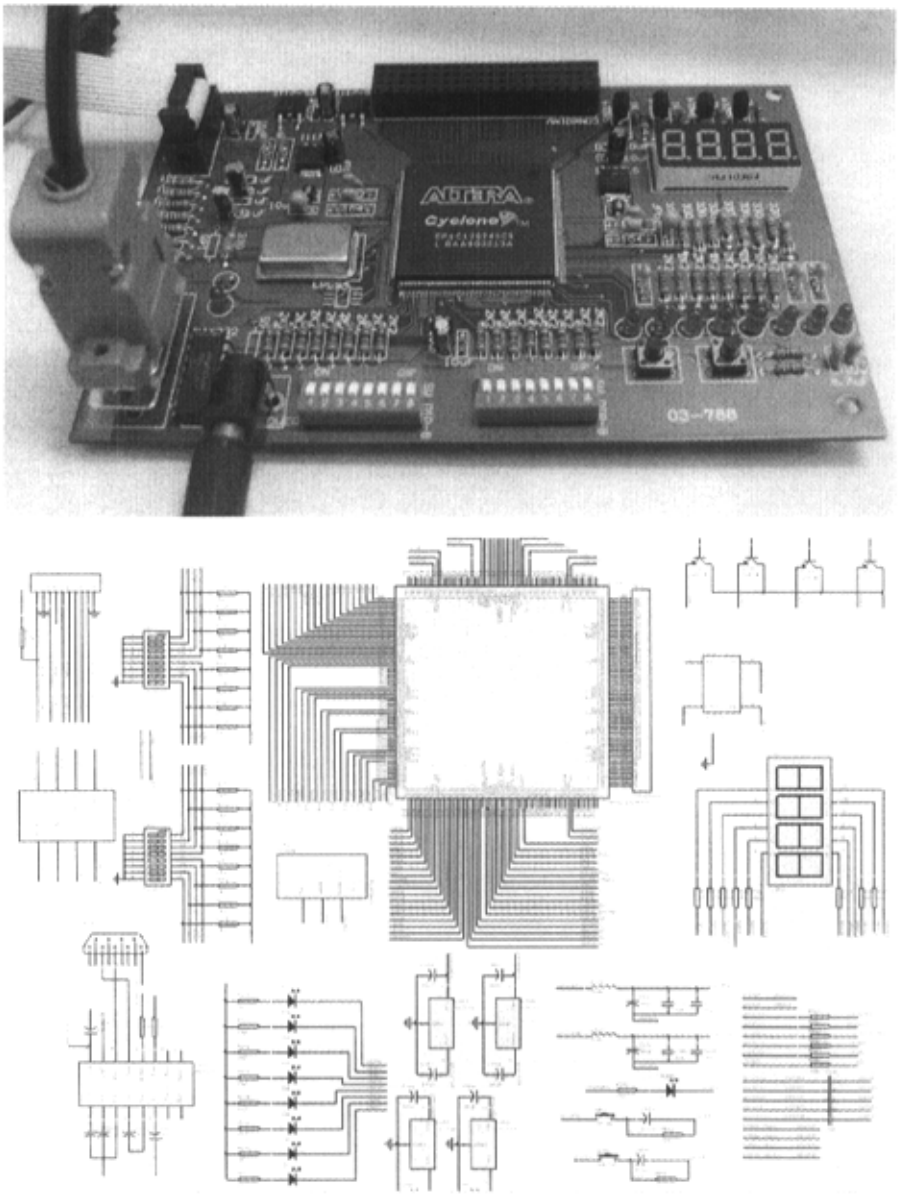


图 4.11 硬件平台的原理与实物图

结　论

论文对基于 QR 分解的递归最小二乘自适应阵列信号处理算法进行研究，讨论了实现此算法的拓扑结构和运算宏单元。在此基础之上采用了能够引入更强并行流水机制的超前处理技术，对应用此技术的自适应抗干扰阵列信号处理系统进行了基于 FPGA 的数据流级设计。最后，搭建了承载此系统的硬件平台。

MATLAB 仿真结果表明：系统具有与经典最小二乘算法同样的收敛速度，但比其具有高的数值稳定性。实验结果表明：系统能够 达到 9MHz 的数据采样速率和 16bit 的数据精度。

参考文献

[1] Haykin S.Adaptive Filter Theory[M]. Third Edition.New Jersey: Prentice Hall, 1996.

[2] 何振亚. 自适应信号处理[M]. 北京:科学出版社,2002:293-296 页

[3] 贡三元. VLSI 阵列处理[M]. 南京:东南大学出版社,1992.12-34 页

[4] 胡国荣, 孙允恭. CORDIC 算法及其应用. 信号处理.1991,7(4):229-242 页

[5] J E Volde.The CORDIC trigonometric computing technique. IEEE Trans. Electronic Computer.1959,Sept:330-334P

[6] J S Walthe.A Unified Algorithm for Elementary Functions.in Proceedings of the Spring Joint Computer Conference. 1971: 379 -385P

[7] S P Applebaum.Adaptive Arrays.IEEE Trans.on AP. 1976,24(5): 585- 598P

[8] B Widrow,et al.Adaptive Antenna Systems.Proc.of The IEEE. 1967,55(12): 2143-2159P

[9] L Frost.An Algorithm for Linearly Constrained Adaptive Array Processing. Proc.of The IEEE.1972,60(8):926-935P

[10] L J Griffiths,C W Jim.An Alternative Approach to Linearly Constrained Adaptive Beamforming.IEEE Trans.on AP.1982,30(1): 27-34P

[11] I S Reed,et al.Rapid Convergence Rate in Adaptive Arrays.IEEE Trans.on AES.1974,10(6)

[12] L E Brennan,et al.Adaptive Arrays in Airborne MTI Radar.IEEE Trans.on AP.1976,24(5):607-615P

[13] R A Monzingo,et al.Introduction to Adaptive Arrays.New York : Wiley, 1980.

[14] W M Gentleman,H T Kung.Martrix triangularization by systolic arrays.in Proc.SPIE: Real.time Signal Process 4.1981:298-303P

[15] J G Mcwhircer.Recuesive least.squares minimization using systolic array.in Proc.SPIE: Real.time Signal Process 4.1983,8: 105-112P

[16] C R Ward,A J Robson.Application of a systolic array to adaptive beamforming.IEE.Proc.1984,10

[17] Y H Hu.Cordic.based VLSI architectures for digital signal processing.IEEE Trans. Signal Processing Mag. 1992,7:16-35P

[18] S F Hsieh,K J R Liu,K Yao.A unified square.root.free Givens rotation approach for QRD.based recursive least squares estimation.IEEE Trans. Signal Processing.1993,41(3):1405-1409P

[19] E Franrzeskakis and K J R Liu.Class of square root and division free algorithms and architectures for QRD.based adaptive signal processing. IEEE Trans. Signal Processing.1994,42(9): 2455-2469P

[20] J M Cioffi.The fast adaptive ROTOR's RLS algorithm.IEEE Trans. Acoust. Speech Signal Processing.1990,38(4):631-653P

[21] 龚耀寰, 李航, 苟仲文. 基于CORDIC的无开方GIVENS旋转处理方法. 电子科技大学学报.1997,26(6):565-569P

[22] K K Parhi,D G Messerschmitt.Pipeline interleaving and parallelism in recursive digital filters—Part1: Pipelining using scattered look.ahead and decomposion.IEEE Trans. Acoust. Speech Signal Processing.1989,37(7): 1118-1134P

[23] K K Parhi,D G Messerschmitt.Pipeline interleaving and parallelism in recursive digital filters—Part2: Pipelined incremental block filtering.IEEE Trans. Acoust. Speech Signal Processing.1989,37(7):1099-1117P

[24] K J Raghunath,K K parhi.Pipelined RLS adaptive filtering using scaled tangent rotations (STAR).IEEE Trans. Signal Processing. 1996,40(10): 2591 -2604P

[25] J Ma,K K Parhi,E F Deprettere.Annihilation.Reordering Look.Ahead pipelined CORDIC.based RLS adaptive filters and their application to adaptive beamforming.IEEE Trans. on Signal Processing.2000,48(8): 2414 -2428P

[26] L J Gao,K K Parhi.Hierarchical Pipelining and Foldeing of QRD.RLS Adaptive Filters and Its Application to Digital Beam. forming.IEEE Trans.

on Circurts and Systems II.2000,47(12): 1503 -1518P

[27] T J Shepherd,J G Mcwhirter,J E Hudson.Parallel weight extration from a systolic adaptive beamformer.In Mathematics Signal Processing 2.1990: 775-790P

[28] J G Mcwhirter,T J Shepherd.Systolic array processor for MVDR beamforming.In Proc. Inst. Elect. Eng.1989,136(4):75-80P

[29] 李承阳, 胡光锐, 龚耀寰. 基于 QR 分解算法的自适应抗干扰阵列最优权值并行提取. 应用科学学报.1998,16(3):326-330 页

[30] 陈晓初，冷　梅。自适应波束形成权递推算法及其脉动阵实现。 电子科学学刊.1997,19(6):751-755 页

[31] 陈晓初. 相控阵雷达自适应数字波束形成. 西安电子科技大学博士论文.1992

[32] M D Miranda,M Gerken.A Hybird Least Squares QR.Lattice Algorithm Using A priori Erroes.IEEE Trans. on Signal Processing. 1997, 45(12): 2900 -2911P

[33] P A Regalia,M G Bellanger.On the Duality Between Fast QR Methods and Lattice Methods in Least Squares Adaptive Filtering.IEEE Trans.on Signal Processing.1991,39(4):879-891P

[34] E N Frantzeskakis,K J R Liu.A Class of Squares Root and Division Free Algorithms and Architectures for QRD.Based Adaptive Signal Processing. IEEE Trans. on Signal Processing. 1994,42(9): 2455- 2469P

[35] H T Kung,C E Leiserson.Systolic Arrays for VLSI.In Sparse Matrix Symposium.256-282P

[36] H T Kung.Why Systolic Architectues.IEEE Trans. on Computer. 1982,1

[37] S Y Kung.VLSI Array Processors.Prentice Hall.1988

[38] K A Gallivan,et al.High Performance Architectures for Adaptive Filtering Based on The Gram Schimidt Algorithm.Proc. of SPIE Real Time Signal Processing VII.1984,495:30-38P

[39] W C Liles,et al.Design Tadeoffs and Implemention of Gram Schimidt Adaptive Arrays.Proc. of 1980 Adaptive Antenna Symp.RADC Griffiss Ari.

Force Base NY :220-234P

[40] W M Gentlemen,H T Kung.Maritx Triangularization by Systolic Array.Proc. of SPIE Real Time Signal Processing IV.1981,29: 253- 270P

[41] H M Ahmed.Alternative Arithmetic Unit Architectures for VLSI DigitalSignal Processors.In VLSI and Modern Signl Processing, Prentice Hall.1985:277-303P

[42] A M Despain.Fourier Transform Computers Using CORDIC Iterations. IEEE Trans.on Computers.1974,10:993-1001P

[43] A M Despain.Very Fast Fourier Transform Algorithms Hardware for Implementation.IEEE Trans.on Computers.1979,5:333-341P

[44] H M Ahmed,M Morff,D T Lee,P H Ang.A VLSI Speech Analysis Chip Set Based on Square.Root Normalized Ladder Forms.ICASSP. 1981: 648-653P

[45] J R Cavallaro,Franklin T Luk.CORDIC Arithmetic for an SVD Processor.In Proc.8th Symp.Comput.Arithmetic.1987,113-120P

[46] M Morf,C H Muravhik,P H Ang,J M Delosme.Fast Cholesky Algorithm and Adaptive Feedback Filters.ICASSP.1982,1727-1731P

[47] P W Baker.Suggestion for a Fast Binary Sine/Cosine Generator.IEEE Trans.on Computers.1976,12:1134-1136P

[48] V Considine.CORDIC Trigonometric Function Generator For DSP. ICASSP. 1989,2381-2384P

[49] G Haviland,A Tuszynsky.CORDIC Arithmetic Processorchip.IEEE Trans.on computers.1980,29(2):647-660P

[50] Cyclone Device Handbook.Altera Corporation,2003

[51] Configuration Handbook.Altera Corporation,2003

[52] 胡国荣, 侯朝焕, 孙允恭. 面向 VLSI 实现的实时自适应滤波算法. 电子学报.1996,24(8):76-82 页

[53] 林敏, 方颖立. VHDL 数字系统设计与高层次综合. 北京:电子工业出版社,2002

[54] 侯伯亨, 顾新. VHDL 硬件描述语言与数字逻辑电路设计. 西安:西安电子

科技大学出版社,1997

[55] 宋万杰,罗丰,吴顺.CPLD 技术及其应用. 西安:西安电子科技大学出版社,2000

[56] 陈弘毅，曾志强.一种快速旋转器的 VLSI 硬件结构. 清华大学学报.1999,39(1):59-62 页

[57] 陈弘毅，曾志强.以快速旋转器为运算核的 VLSI 变换微处理器. 清华大学学报.1999,39(1):67-70 页

[58] 陈弘毅，曾志强.基于旋转算术的正交变换快速分解算法.清华大学学报.1999,39(1):63-66 页

[59] 胡国荣.递归最小二乘脉动阵列及其实现.中国科学院声学研究所硕士论文.1992: 19-30 页

攻读硕士学位期间发表论文和取得的科研成果

[1] 杨苹元，徐法禄，郝敬涛. 一种自适应阵列信号处理算法的高速实现. 哈尔滨工程大学学报. 2004 年第一期发表

[2] 徐法禄，杨苹元，张波. 面向 VLSI 的快速自适应最小二乘算法. 哈尔滨工程大学学报. 被录用

[3] 徐法禄，杨苹元. 基于 GSM 的 GPS 车辆监控系统. 应用科技. 2004 年第三期发表

[4] 徐法禄，杨苹元. CDMA/GSM 直放站远程监控系统. 应用科技. 已投

致　　谢

本文的工作是在恩师杨莘元教授的悉心指导和严格要求下完成的，每每在工作的关键时刻，杨老师以严谨的治学态度、深刻的洞察力和对问题的独到精辟的见解，为我指明进一步研究的方向，并使我建立起前进的信心。两年多来的学习和生活，杨老师教我们怎样做人，怎样学习和工作，这些将使我获益终生。

同时还要感谢 808 教研室的全体老师和各位师兄师弟，他们给了我很大的帮助。