计算机控制系统

第7章 计算机控制系统的构建

北京航空航天大学

xia jie

2020年4月

7.1 计算机控制系统硬件配置

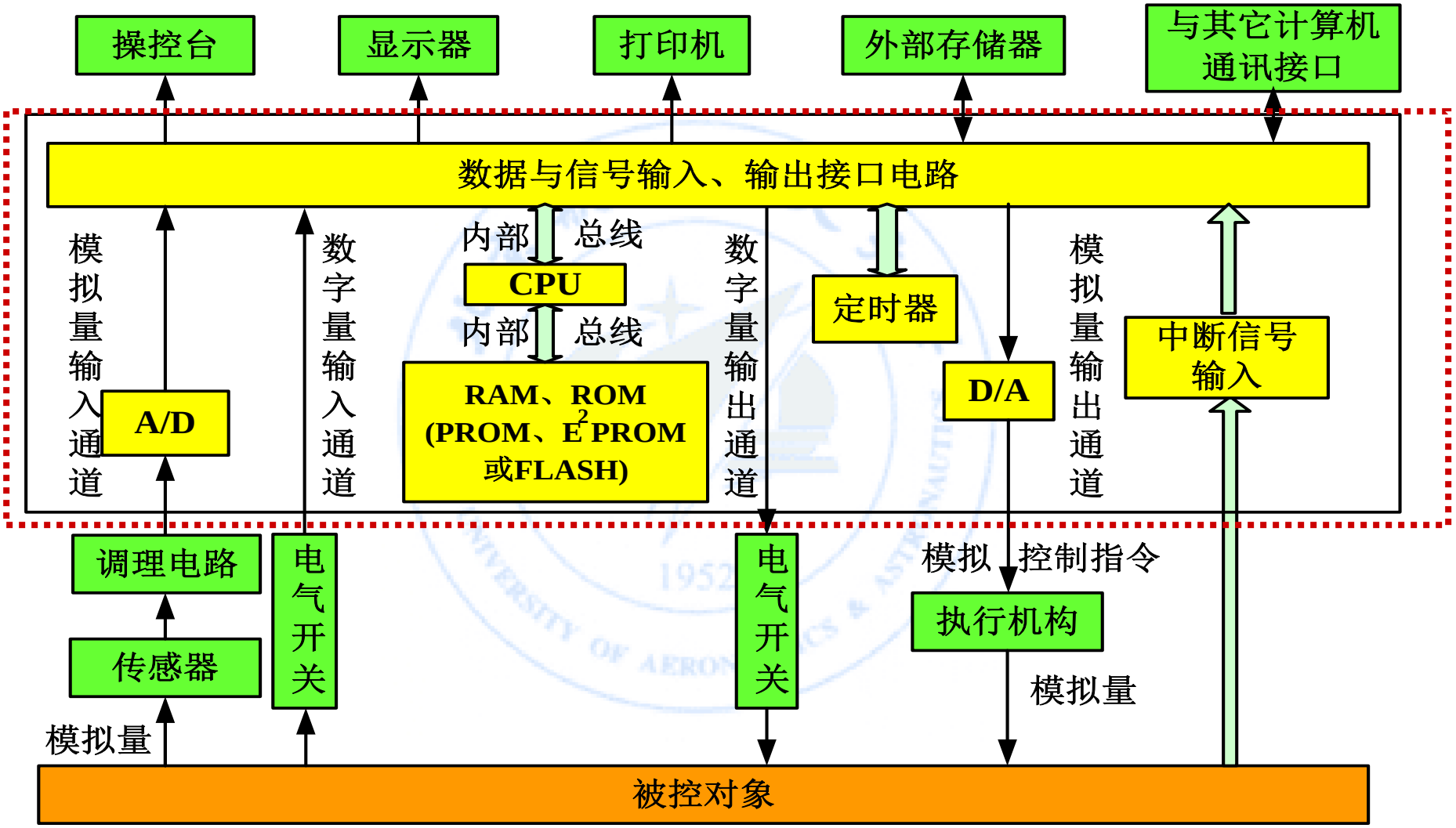


图7-1 计算机控制系统的基本组成

7.1.1 控制用计算机系统的硬件要求

（一）对计算机主机的要求

1. 实时处理能力
2. 比较完善的中断系统
3. 对指令系统的要求
4. 对内存的要求

（二）对过程输入输出通道的要求

1. 有足够输入通道数，并具有一定的扩充能力
2. 有足够的精度和分辨率
3. 应有足够的变换速度

7.1.2 对控制用计算机系统的一般要求

（三）对应用软件系统的要求

- 实时性强、可靠性好、具有在线修改能力、输入输出功能强等

（四）方便的人—机联系

- 显示屏 、各种功能键 、输入数据功能键等

（五）系统的可靠性和可维护性

- 可靠性指系统无故障运行能力，指标——平均无故障间隔时间。
  - 硬件可靠性和软件可靠性
- 可维护性之进行维护时方便的程度
  - 插件式硬件、自检测、自诊断程序
  - 及时发现故障，判断故障部位，进行维修。

7.2 CCS计算机的选用

1、 对控制用计算机的选择

（一）运算速度

- 影响因素：系统计算工作量、采样周期、指令系统、硬件支持

（二）计算机字长

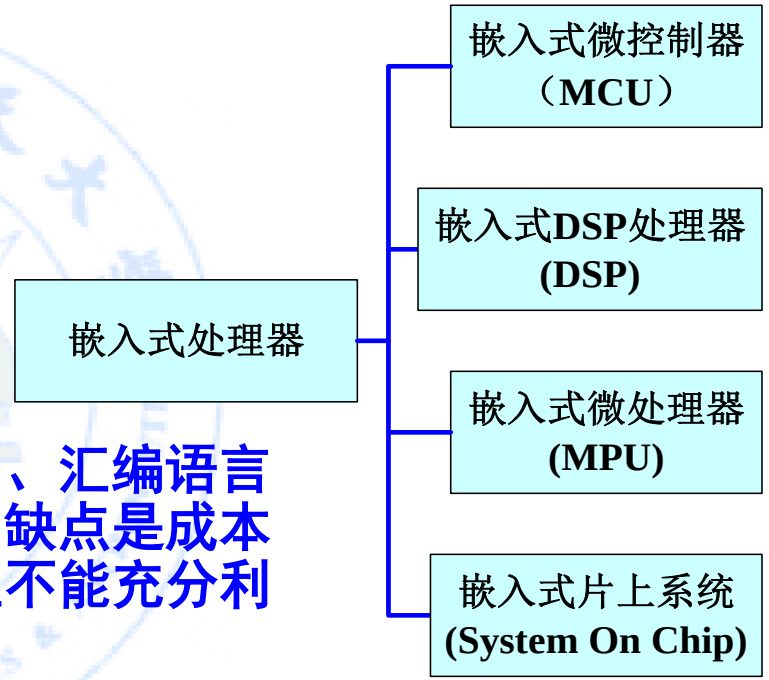
- 影响因素：

1. 量化误差
2. 应与A/D的字长相协调 $n_{cpu} = n_{A/D} + 4$
3. 信号的动态范围 $N = \dfrac{X_{\max}}{X_{\min}} \Longrightarrow n \geq \dfrac{\lg(N+1)}{\lg 2}$
4. 与采样周期T的关系
   - 若T减小，但又希望量化误差保持不变，则所需的计算机的字长就要相应增加。

2、常用控制计算机的类型

还应当考虑成本高低、程序编制难易以及扩充输入输出接口是否方便等因素。

- 嵌入式处理器
  - MicroController Unit，MCU
  - Digital Signal Processor，DSP
  - MicroProcessor Unit，MPU
  - System On Chip，SOC
- 微型计算机（工控机）
  - 有丰富的系统软件，可用高级语言、汇编语言编程，程序编制和调试都很方便。缺点是成本较高，当用于控制小系统时，往往不能充分利用系统的全部功能。
- PLC（可编程控制器）
  - 广泛运用于工业过程控制领域
  - 为现代工业自动化三大支柱之一（PLC、机器人、CAD/CAM）



（1）嵌入式微控制器（MCU）

- 嵌入式微控制器的典型代表是单片机这种８位的电子器件目前在嵌入式设备中仍然有着极其广泛的应用。
- 单片机芯片内部集成ROM/EPROM、RAM、总线、总线逻辑、定时/计数器、看门狗、I/O、串行口、脉宽调制输出、A/D、D/A、Flash RAM、EEPROM等各种必要功能和外设。
- 微控制器的最大特点是单片化，体积大大减小，从而使功耗和成本下降、可靠性提高。
- 微控制器是目前嵌入式系统工业的主流。微控制器的片上外设资源一般比较丰富，适合于控制，因此称为微控制器。
- 由于MCU低廉的价格，优良的功能，所以拥有的品种和数量最多，比较有代表性的包括8051、MCS-251、MCS-96/196/296、P51XA、C166/167、68K系列以及 MCU 8XC930/931、C540、C541。

（2）嵌入式数字信号处理器（DSP）

- DSP处理器是专门用于信号处理方面的处理器，其在系统结构和指令算法方面进行了特殊设计，在数字滤波、FFT、谱分析等各种仪器上DSP获得了大规模的应用。
- DSP的理论算法在70年代就已经出现，但是由于专门的DSP处理器还未出现，所以这种理论算法只能通过MPU等由分立元件实现。 1982年世界上诞生了首枚DSP芯片，在语音合成和编码解码器中得到了广泛应用。DSP的运算速度进一步提高，应用领域也从上述范围扩大到了通信和计算机方面。
- 目前最为广泛应用的嵌入式DSP处理器是TI的TMS320C2000/C5000系列，另外如Intel的MCS-296和Siemens的TriCore也有各自的应用范围。

（3）嵌入式微处理器（MPU)

- MPU嵌入式微处理器是由通用计算机中的CPU演变而来。
- 与计算机处理器不同的是，在实际嵌入式应用中，只保留和嵌入式应用紧密相关的功能硬件，去除其他的冗余功能部分，这样就以最低的功耗和资源实现嵌入式应用的特殊要求。
- 和工业控制计算机相比，嵌入式微处理器具有体积小、重量轻、成本低、可靠性高的优点。目前主要的嵌入式处理器类型有Am186/88、Power PC、68000、MIPS、ARM/ StrongARM系列等。

（4）嵌入式片上系统(SOC)

- SoC嵌入式系统微处理器就是一种电路系统。
- 它在一个硅片上实现一个复杂的系统，其最大的特点是实现了软硬件的无缝结合，直接在处理器内嵌入操作系统的代码模块。
- SoC嵌入式系统微处理器所具有的好处:
  - 利用内部工作电压，降低芯片功耗。
  - 减少芯片对外管脚数，简化制造过程。
  - 减少外围驱动接口单元及电路板之间的信号传递，加快微处理器数据处理的速度。
  - 内嵌的线路可以避免外部电路板在信号传递时所造成系统杂讯。

补充：定时器的选择使用

- 微机
  - Window平台
    - Timer(18.2Hz55ms)　多媒体定时器ms
  - DOS平台、winMe平台
  - 实时操作系统RTOS
- 单片机系统(80C196)
  - 定时器T0、T1 （16位） 、高速输出口定时设置HSO
- 单片机系统(ATmega128)
  - 定时器/计数器T0、T2 （8位）
  - 定时器/计数器T1、T3 （16位）
- DSP
  - 定时器timer0， timer1， timer2

7.3.1 模拟量输出通道

1. D/A转换器工作原理

数字量转换为模拟电压量或电流量的装置。

$$V_0 = V_{REF} \cdot D \cdot K$$

$$D = D_{n-1}2^{n-1} + D_{n-2}2^{n-2} + \cdots + D_1 2^1 + D_0 2^0$$

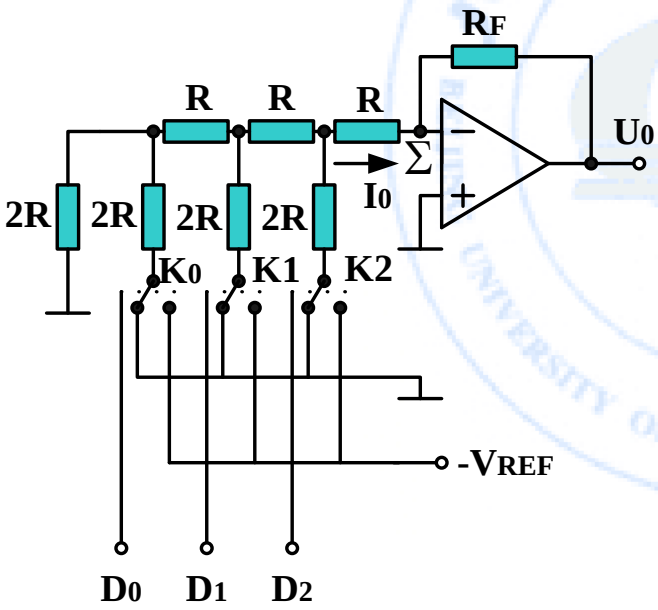


图8-5 采用T型电阻解码网络的D/A转换器

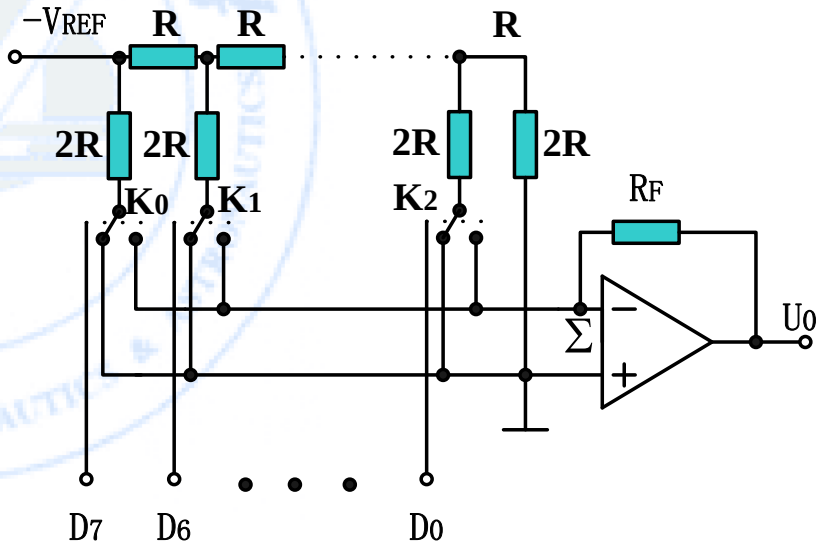


图8-6 采用反向R-2RT型解码网络的D/A转换器

2. D/A转换器的主要性能及常用的主要指标：

① 精度——精度是反映实际输出与理想数学模型输出信号接近的程度。

② 分辨率——分辨率可定义为当输入数字量发生单位数码变化时输出模拟量的变化量。分辨率也常用数字量的位数来表示。注意与精度的区别。

③ 转换时间——最小有效位常以LSB表示，故转换时间定义为D/A转换器中的输入代码有满刻度值的变化时，其输出模拟信号达到满刻度值±½ LSB时所需要的时间。

④ 输出电平——电压型：5~10V，24~30V；电流输出型：20mA，3A等。

⑤ 输入代码形式——D/A转换器单极性输出时，有二进制码、BCD码。当双极性输出时，有符号+数值码，偏移二进制码等。

3. D/A转换器选择的原则

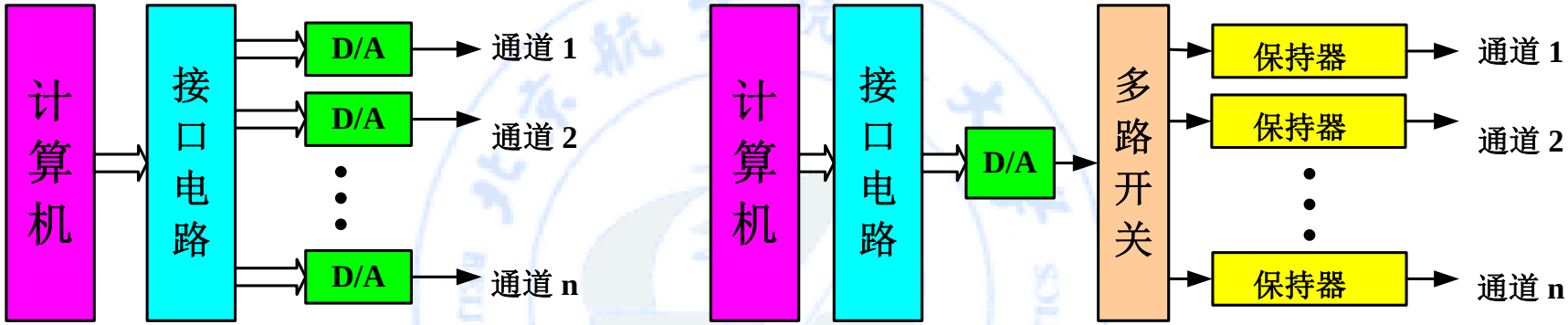
- 集成D/A转换器的输入方式：
  - ❖不带缓冲寄存器（如8位的DAC0808）
  - ❖带缓冲寄存器（如8位的DAC0832、12位的DAC1208等）。
- 选择D/A转换芯片
  - ❖主要考虑芯片的性能、结构及应用特性。在性能上必须满足D/A转换的技术要求，在结构和应用上满足接口方便，外围电路简单，价格低廉等要求。
- 对于D/A转换器字长n的选择，可以由其后的执行机构的动态范围来选定：

执行机构最大输入

$$\frac{u_{\max}}{2^n-1} \le u_R \quad \Rightarrow \quad n \ge \lg\left[\frac{u_{\max}}{u_R}+1\right] / \lg 2$$

执行机构的死区电压

4. 多路D/A输出时的实现方式



(a) 多D/A结构　　(b) 共享D/A结构

图7-2 模拟量输出通道的两种实现结构图

5. D/A的二进制码制与极性

1）单极性二进制编码 $N=\sum_{i=1}^{n}a_i 2^{-i}$ $(i=1,2,\cdots,n)$

2）双极性二进制编码

- 有符号的二进制可以用原码、补码、反码和偏移二进制码来表示。为了把双极性的信号表示成数字代码，就需要增加一位“符号位”。增加一个符号位可以使量程增加一倍，但分辨率却要降低一倍。这几种编码与十进制数的关系如表7-1所示。

注意：

- 计算机内信号的编码可能与D/A输入信号的编码不完全一致。若一致，则可将计算机的运算输出直接作为D/A转换器的输入。但若不一致（多数情况），则需要将计算得到的码制进行相应的转换后，方可作为D/A的输入信号。

7.3.2 模拟量输入通道

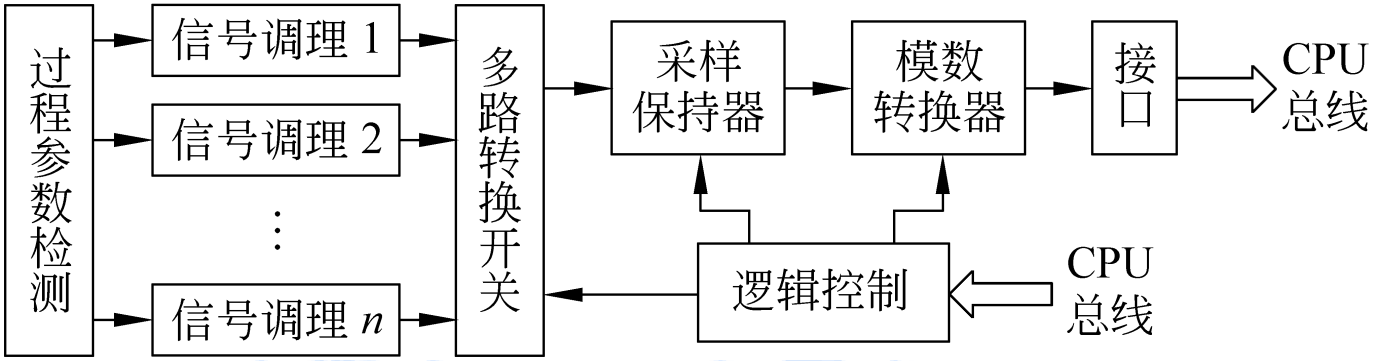
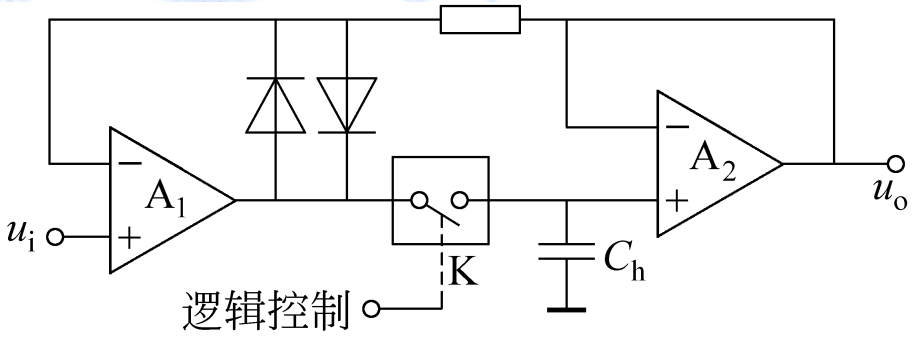


图7-5 模拟量输入通道一般结构图

1. 采样保持器

- 孔径时间——实际的采样过程需要的时间。
- 缩短孔径时间的措施：将对模拟信号的采样和对采样的模拟电压的转换分开，分别由不同的电路完成。
- 采样保持器作用就是以较短的孔径时间对信号进行采样，然后将采得的模拟电压保持，供A/D转换电路进行转换。

图7-8 采样保持器原理图



输入模拟信号量的调理

1、直流电压信号的调整

- 设计相应的调理电路（如分压、放大等），将直流信号转换成计算机所能接受电压形式，再直接使用A/D转换器。

2、直流电流信号的调理

- 设计电流到电压的转换电路。

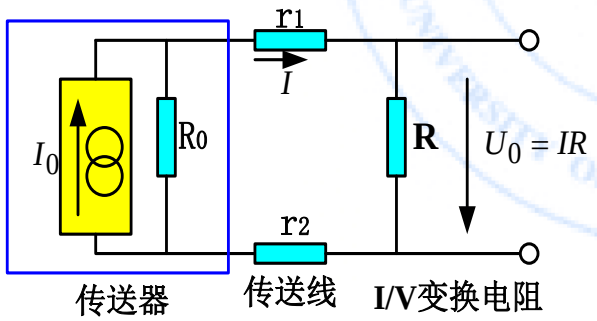


图7-6 电流信号传输的典型电路

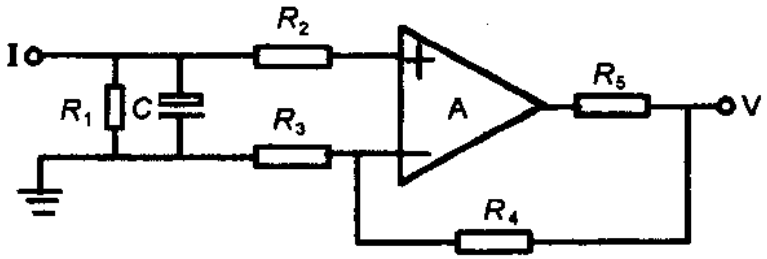


图7-7 有源I/V变换电路

7.3.2 模拟量输入通道

2. A/D 转换器工作原理

将输入的模拟电压按比例地转化为二进制数字信号的装置。

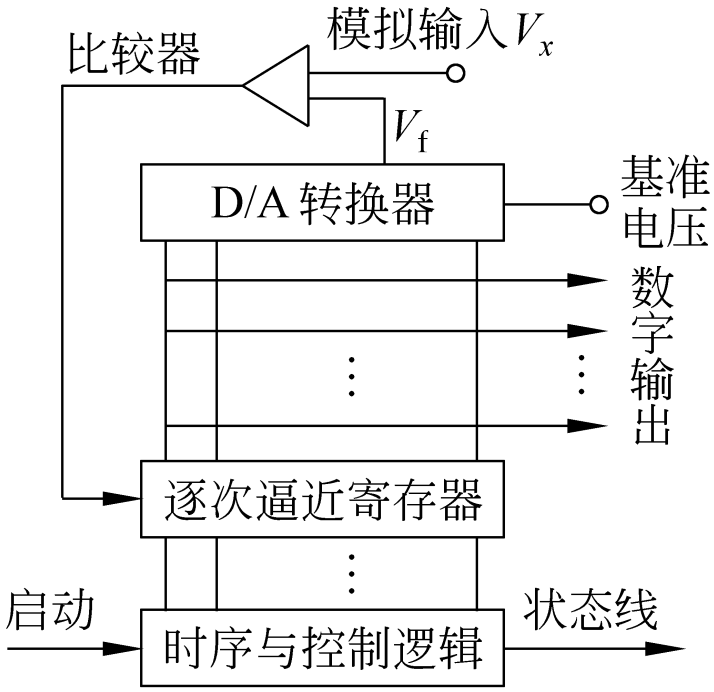


图7-9 逐次逼近式A/D转换器

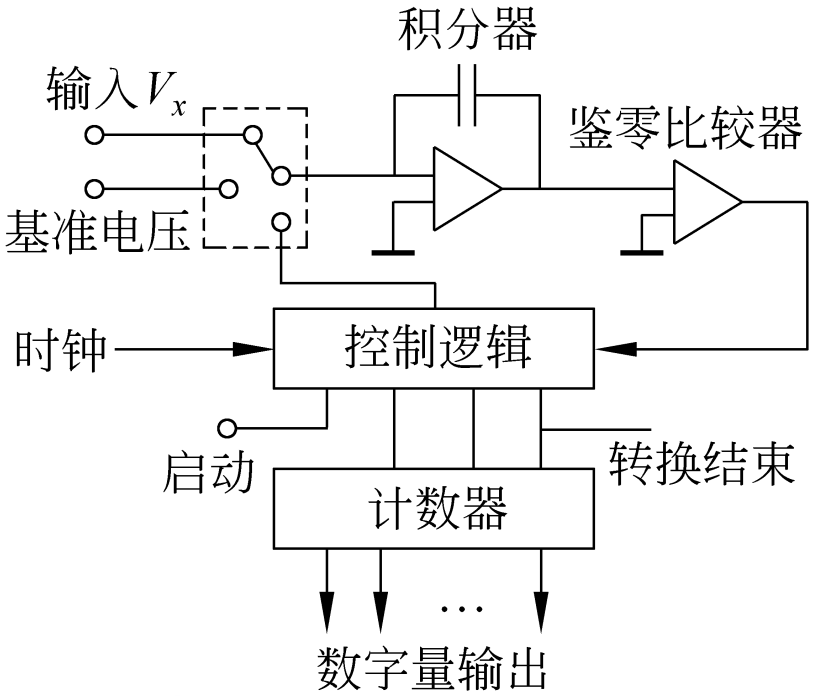


图7-10 双斜积分式A/D转换器

3. A/D转换器的主要性能指标

1）**精度**——指对应一个给定的数字量的实际模拟量输入与理论模拟量输入接近的程度。

2）**分辨率**——指输出数字量对输入模拟量变化的分辨能力。即设A/D转换器的位数为，则A/D转换器的分辨率为

$$D = 1/2^n \quad \text{或} \quad D = 1/(2^n - 1)$$

3）**转换时间**——从A/D转换的启动信号加入时起，到获得数字输出信号为止，所需的时间。

4）**量程** ——指测量的模拟量的变化范围。

- 单极性（如0~10V、0~20V）
- 双极性（例如-5V~+5V、-10V~+10V）。

4. A/D转换器的选择

- 除了要满足用户的各种技术要求外，还必须注意：
  - A/D输出的方式
  - A/D芯片对启动信号的要求
  - A/D的转换精度和转换时间
  - 它的稳定性及抗干扰能力等
- A/D转换器的精度与传感器的精度有关，一般比传感器的精度高一个数量级；A/D转换器的转换速率还与系统的频带有关。
- 根据输入模拟信号的动态范围可选择A/D转换器位数n

模入信号的最大值　　模入信号的最小值

$$\frac{u_{\max}}{2^n-1} \le u_{\min} \longrightarrow n \ge \lg\left[\frac{u_{\max}}{u_{\min}}+1\right] / \lg 2$$

5. 检测通道的数据采集

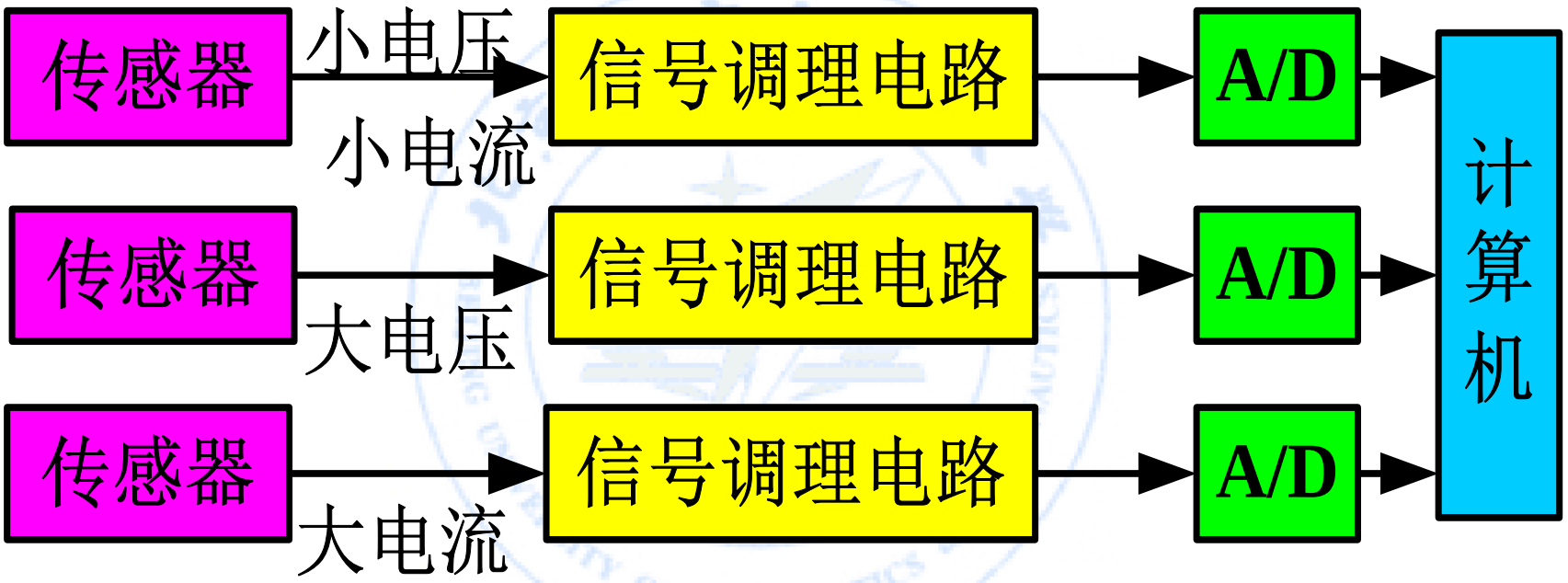


图7-5 单路检测通道结构类型

6. A/D的二进制码制与极性

- A/D的二进制码制与极性
  - 类似于D/A的二进制码制与极性，可同时参见表7-1（此时表中的V_{REF}为A/D的量程）和表7-2。在实际应用中，A/D输出的代码形式可能采用前面介绍的几种二进制编码中的一种。
- 注意：
  - 计算机内信号的编码可能与A/D输出信号的编码不完全一致。若一致，则可将A/D输出信号的编码直接作为计算机的运算输入信号。但若不一致（多数情况），则需要将A/D输出信号的编码进行相应的转换后，方参与到算法的运算中。

6. CPU和A/D转换电路之间的I/O控制方式

1)查询方式

- 由CPU执行I/O指令启动并完成。每次传送数据之前，要先输入A/D转换器状态，经过查询符合条件后才可以进行数据的I/O。
- 灵活，但在读写数据端口指令之前需要重复执行多次查询状态的指令，当外设速度比较慢时，会造成CPU效率的大大降低。

2)中断方式

- 可以省掉重复繁琐的查询，并可及时响应外设的要求。在这种方式下，CPU和外设基本上实现了并行工作，当然由于增加了中断管理功能，所以对应的接口电路和程序要比查询方式复杂。

3)DMA方式

- 在高速数据采集系统中，不仅要选用高速A/D转换电路，而且传送转换结果也要求非常及时迅速，可以考虑选用DMA方式。

7.3.3 数字量输入输出通道

- 输入缓冲器的作用
  - 对外部输入信号进行缓冲、加强和选通。
- 输出锁存器作用
  - 将CPU输出的数据或控制信号进行锁存，以便放大驱动执行机构作用于被控对象。
- I/O电气转换部分的功能：
  - 滤波、电平转换、隔离、功率驱动等。

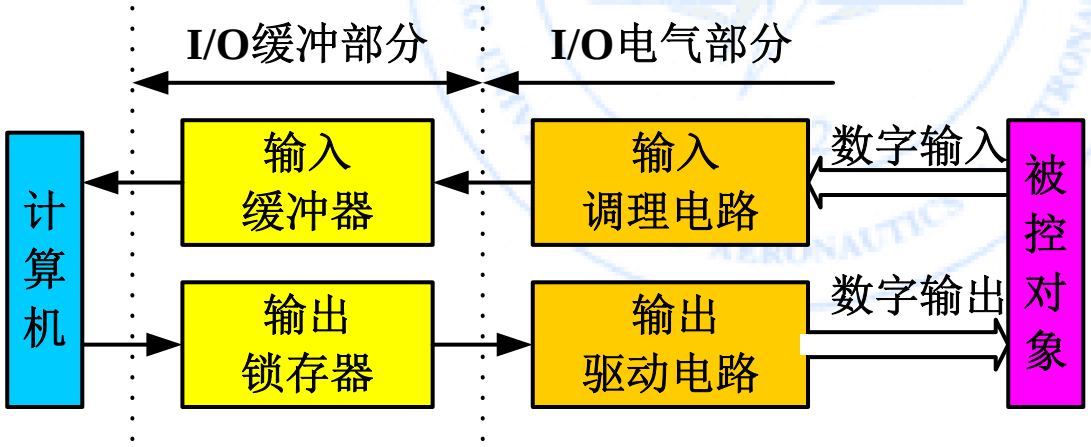


图7-11 开关量输入输出通道结构

- 齐纳二极管或压敏电阻
- 串联二极管
- 限流电阻齐纳二极管
- 光电隔离器
- RC滤波器

7.4 CCS总线技术

7.4.1 总线概述

1. 总线定义

- 总线是一组信号线的集合。这些线是系统的各插件间（或插件内部各芯片间）、各系统之间传送规定信息的公共通道，有时也称数据公路，通过它们可以把各种数据和命令传送到各自要去的地方。

2. 总线类型

1）根据总线不同的结构和用途的分类

- 专用总线
  - 只实现一对物理部件间连接的总线。
- 非专用总线
  - 可以被多种功能或多个部件所共享。准确应称为分时共享总线。

2）根据总线的用途和应用环境的分类

- 局部总线（芯片或元件级总线）
  - 构成中央处理机或子系统内所用的总线。
- 系统总线（内总线和板级总线）
  - 用于各单微处理机之间、模块之间的通信，可用于构成分布式多机系统，如 STD总线、VME总线、PC总线等。
- 外总线（通信总线）
  - 用于微处理机与其它智能仪器仪表间的通信，如RS-232C等。

3）根据总线传送信号的方式的分类

- 并行总线
  - 用若干根信号线同时传递信号，就构成了并行总线。
- 串行总线
  - 按照信息逐位的顺序传送信号。

3. 目前几种通用总线介绍

（1）STD 总线

- 目前工业控制及工业检测系统中使用最广泛的总线，它兼容性好，能够支持任何8位或16位微处理器，成为一种通用标准总线。
- 具有以下特点：
  - 小板结构，高度模块化
  - 严格的标准化，广泛的兼容性
  - 面向I/O的开放式设计，适合工业控制应用
  - 高可靠性
- STD是工业应用中十分有前途的通用标准总线。按此标准设计系统，可使系统具有良好的适应性及组装灵活性。目前国内外许多厂家均按STD标准来生产系统和插件，因此，对应用者来说，按STD标准来组成自己的应用系统将会大大缩短系统的硬件研制周期。

（2）IBM PC/AT 总线

- 由于IBM PC机有丰富的软、硬件支持，而且其价格低廉，目前已成为国际上广泛使用的微型机之一。
- IBM PC机的主板上设计了供输入输出用的总线，这些总线引至系统板上的5个或8个62脚的插座上，这些插座称为扩展插槽。
- 制造商提供的用作扩充PC机的选件板有百余种之多，如同步通讯控制卡、异步通讯控制卡、A/D及D/A转换板、数据采集板、各类存储器扩展板、打印机接口板、网络接口板等。用户可根据需要进行选购，也可根据需要自行设计和开发新的功能板。
- PC/AT总线对环境要求较高，无法保证在工业现场可靠运行。
- PC／AT总线都是主要采取将微处理器芯片总线经缓冲直接映射到系统总线上，没有支持总线仲裁的硬件逻辑，因而不支持多主系统。

（3）RS—232C 串行接口标准总线

- 由电子工业学会正式公布的串行总线标准，也是在微机系统中最常用的串行接口标准，用于实现计算机与计算机之间、计算机与外设之间的同步或异步通讯。
- 采用RS—232C作串行通讯时，传输数据的速率可任意调整，最大可达20Kb。
- 两种连接系统的方式：
  - 近程（传输距离小于15m）通讯，这时可以用RS—232C电缆直接连接。
  - 远程（15m以上的长距离）通讯，需要采用调制解调器（MODEM）经电话线进行。

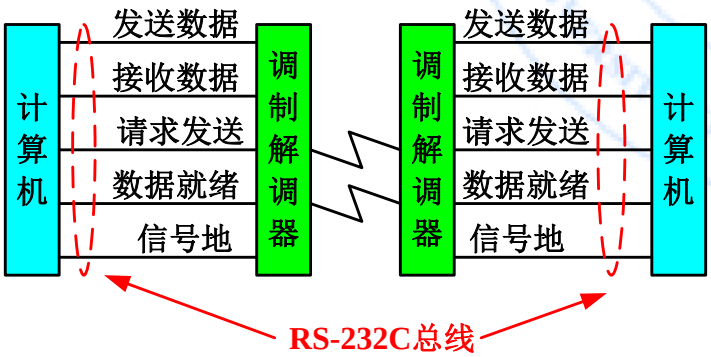


图7-14　计算机与终端的远程连接

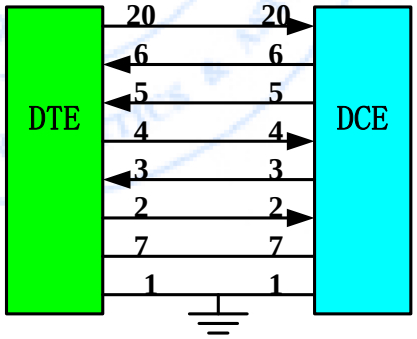


图7-15 RS--232C接口的主要连线

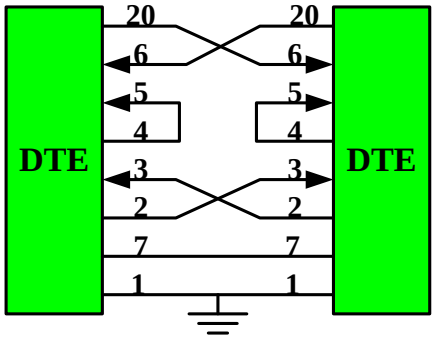
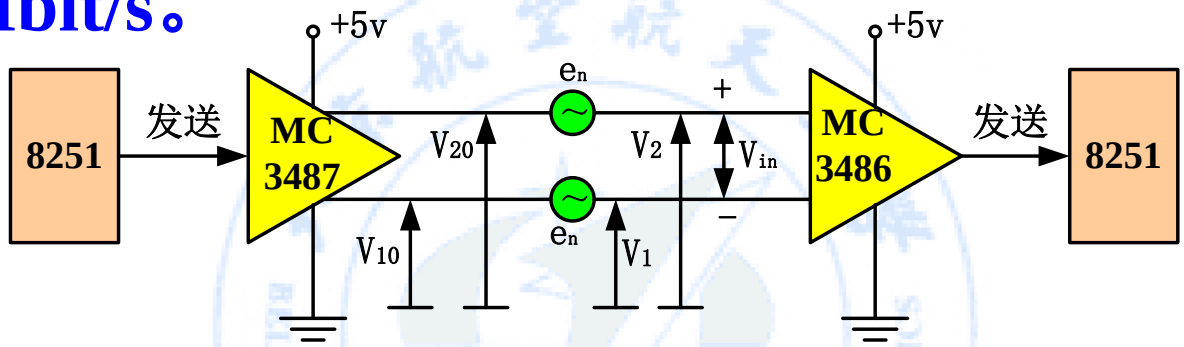


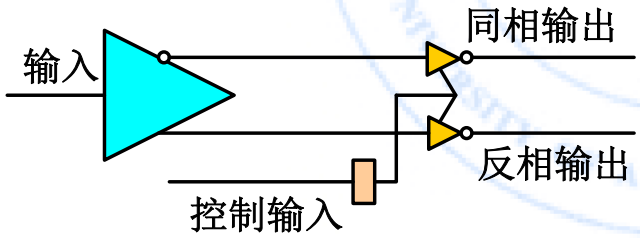
图7-16 计算机与终端间RS--232 C对接

（4）RS—422 串行接口标准总线

- 采用了平衡驱动和差分接收器组合的双端接口方式
- 传输距离可以达到1000米，传输波特率可以达到10Mbit/s。



（a）RS-422接口电路



（b）平衡驱动器电原理图

输入	控制输入	同相输出	反相输出
高	高	高	低
低	高	低	高
X	低	高阻态	高阻态

（c）平衡驱动器真值表

图7-17 RS-422发送驱动器

7.5 CCS实时软件设计

7.5.1 实时软件概述

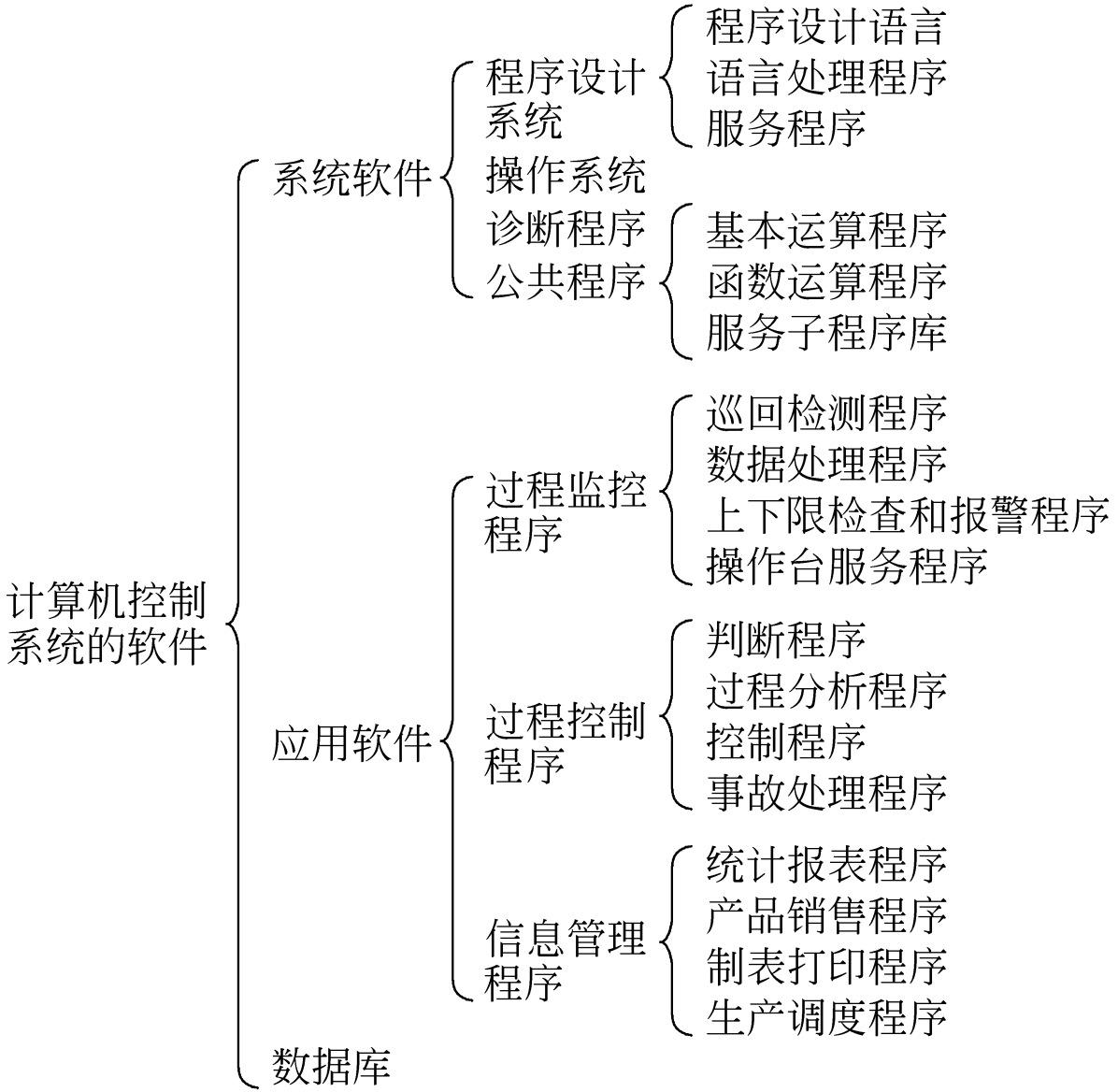


图7-18 计算机控制系统的软件组成

软件的分类

- 系统软件
  - 实时操作系统
    - 管理、调度
  - 应用系统软件
    - 编辑、编译、连接
    - 调试、子程序库
- 应用软件
  - 控制程序
  - 数据采集及处理程序
  - 巡回检测程序
  - 数据管理程序

7.5.2 实时控制程序设计语言

1. 机器语言
2. 汇编语言
3. 高级语言
4. 高级语言和汇编语言的混合使用

7.5.3 实时控制软件的设计

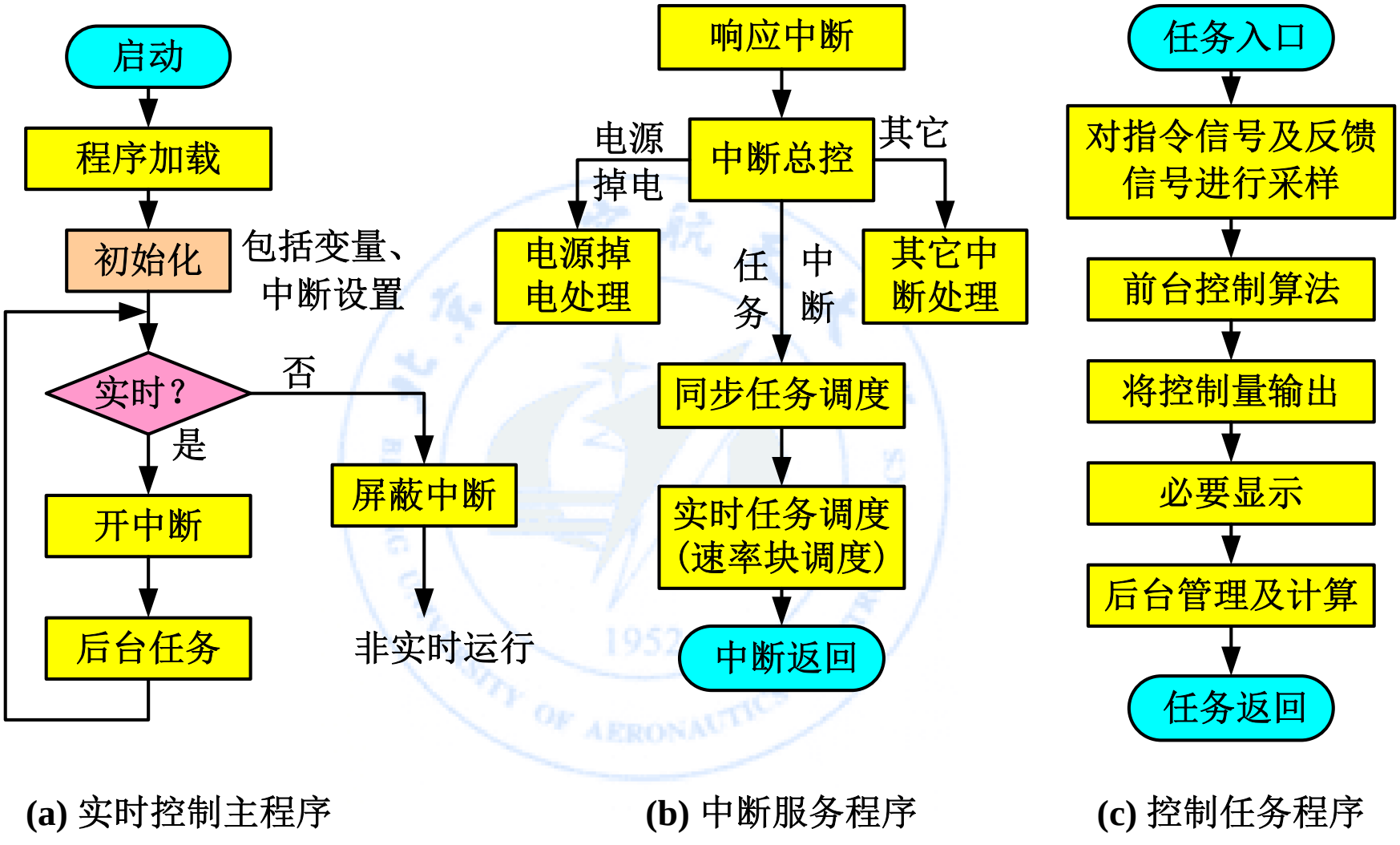
1. 实时控制软件

(1) 实时管理软件

- 实时时钟管理
- 输入 / 输出信息管理
- 中断管理功能
- 任务调度
- 人—机联系
- 设置系统的初始状态

(2) 过程监视及控制算法计算软件

- 数据变换处理程序
- 控制指令生成程序
- 事故处理程序
- 信息管理程序
- 基本运算程序
- 码制及格式转换程序



(a) 实时控制主程序　　**(b)** 中断服务程序　　**(c)** 控制任务程序

图7-19 典型的计算机实时控制系统的程序流程框图

2. 控制算法设计中减少计算时延的方法

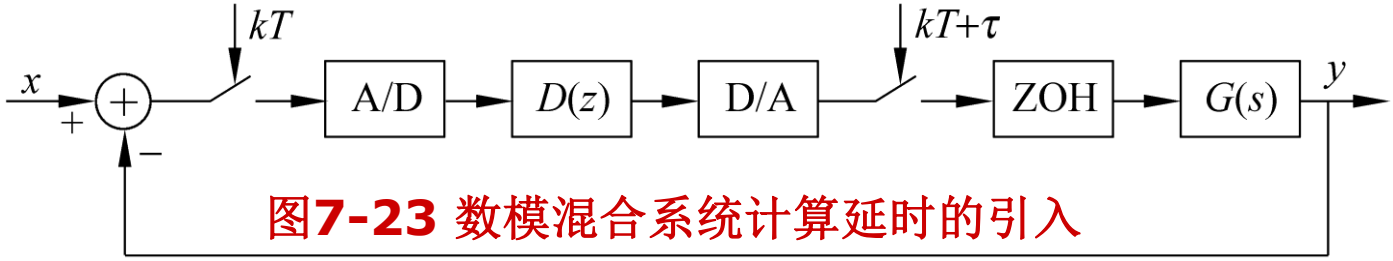
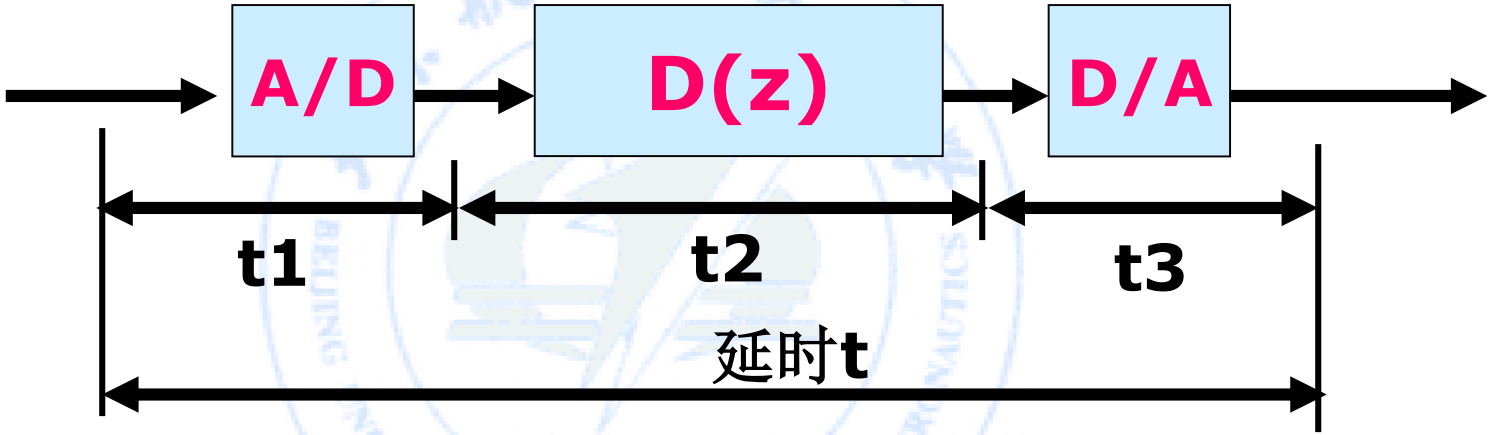


图7-23 数模混合系统计算延时的引入

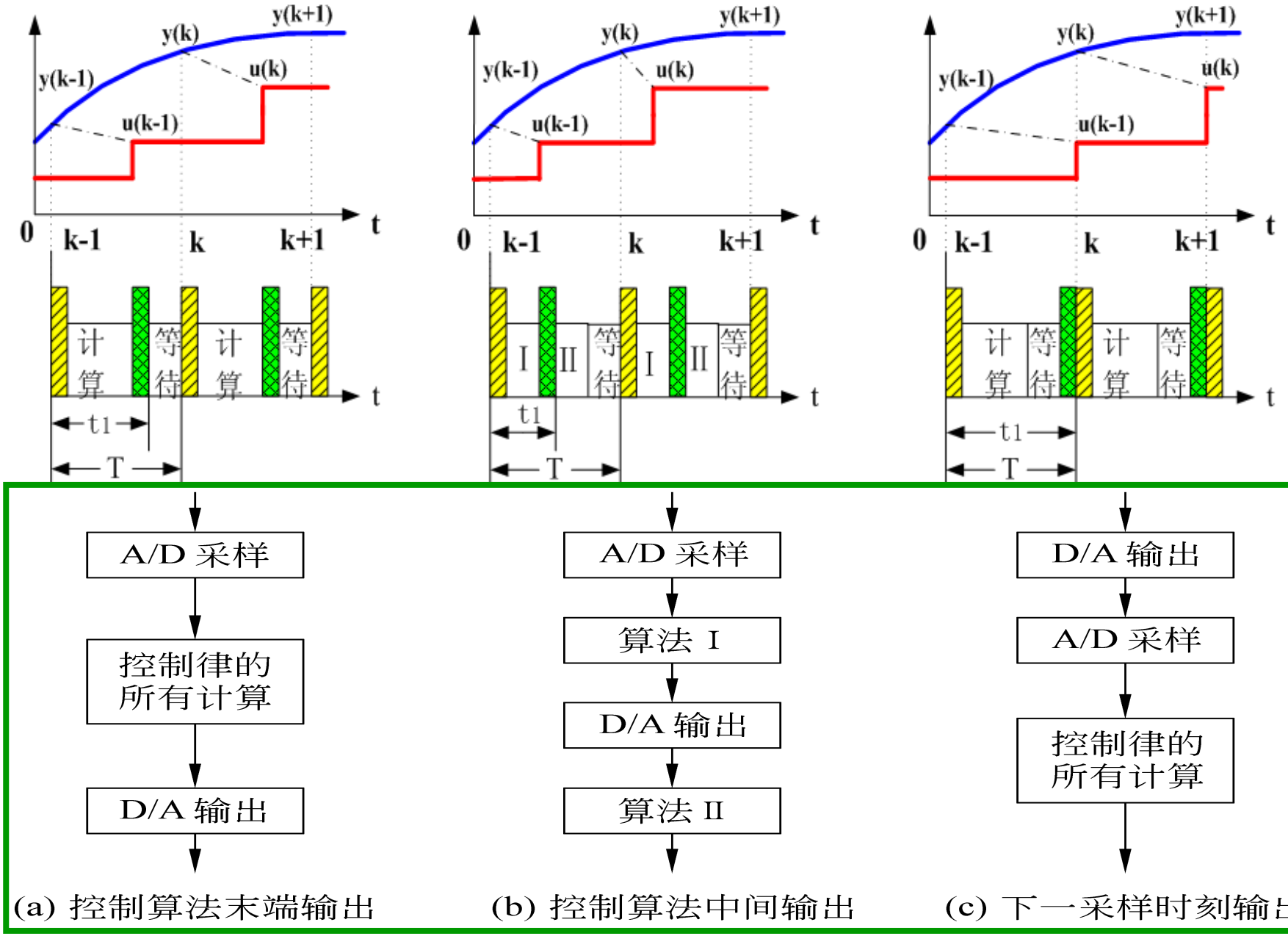


- 延时对控制系统有不好的影响！

算法I：包括那些为了得到当前输出值而必须进行的计算。

算法II：包括那些为了得到下一时刻输出值而必须进行的计算，以及与当前输出无关的其它计算和管理算法。

图7-24 三种控制算法的输出时刻



3、 控制算法的编排

1）直接型结构

$$D(z)=\frac{U(z)}{E(z)}=\frac{\sum_{i=0}^{m}b_i z^{-i}}{1+\sum_{i=1}^{n}a_i z^{-i}}$$

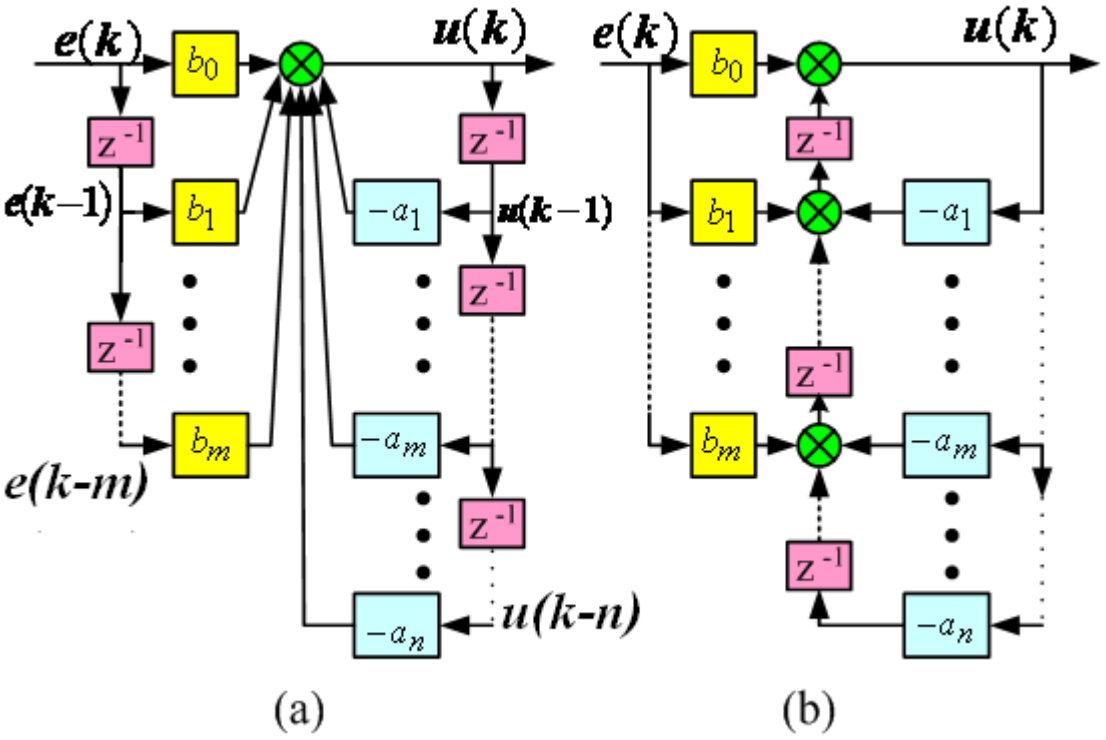
（1） 零极型

$$U(z)=\sum_{i=0}^{m}b_i z^{-i}E(z)-\sum_{i=1}^{n}a_i z^{-i}U(z)$$

$$u(k)=\sum_{i=0}^{m}b_i e(k-i)-\sum_{i=1}^{n}a_i u(k-i)$$

直接型结构

$$D(z)=\frac{U(z)}{E(z)}=\frac{\sum_{i=0}^{m}b_i z^{-i}}{1+\sum_{i=1}^{n}a_i z^{-i}}$$



实现比较简单，不需要做任何变换。

缺陷：如果控制器中任一系数有一定的误差，将使控制器所有的零极点产生相应的变化。

零极型

$$u(k)=\sum_{i=0}^{m}b_i e(k-i)-\sum_{i=1}^{n}a_i u(k-i)$$

(2) 极零型

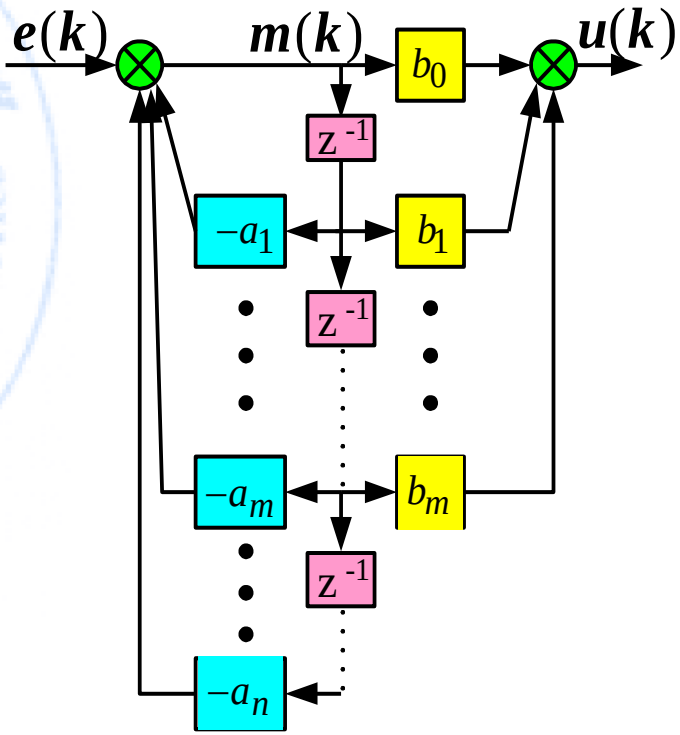
$$D(z)=\frac{U(z)}{M(z)}\cdot\frac{M(z)}{E(z)}$$

$$=\sum_{i=0}^{n}b_i z^{-i}\cdot\left[\frac{1}{1+\sum_{i=1}^{n}a_i z^{-i}}\right]$$

$$D(z)=\frac{U(z)}{E(z)}=\frac{\sum_{i=0}^{m}b_i z^{-i}}{1+\sum_{i=1}^{n}a_i z^{-i}}$$

$$m(k)=e(k)-\sum_{i=1}^{n}a_i m(k-i)$$

$$u(k)=\sum_{i=0}^{m}b_i m(k-i)$$



极零型

2） 串联型结构

$$D(z)=\frac{U(z)}{E(z)}=b_0D_1D_2\cdots D_l$$ 其中D_i为 $\frac{1+\beta_i z^{-1}}{1+\alpha_i z^{-1}}$ 或 $\frac{1+\beta_{i1}z^{-1}+\beta_{i2}z^{-2}}{1+\alpha_{i1}z^{-1}+a_{i2}z^{-2}}$

- 因式分解得一阶或二阶的环节乘积,可用这些低阶环节的编排结构（采用直接型编排实现）进行串联而得。

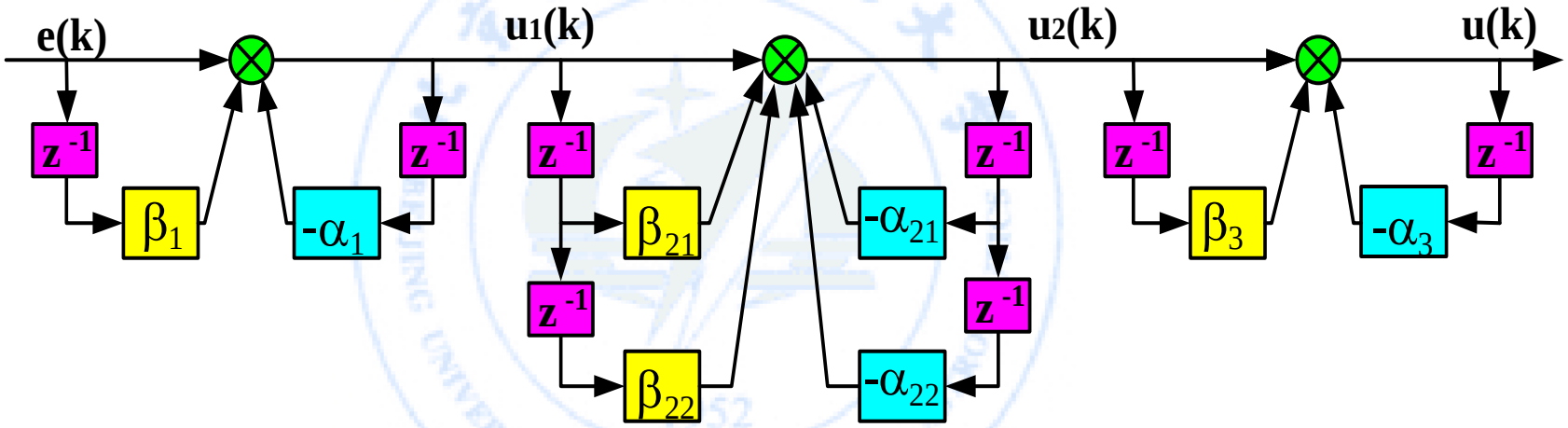


图7-21 串联型编排实现结构图

优点：

- 控制器中某一系数产生误差，只影响相应环节零点或极点；
- 环节前后顺序不同影响系统总的误差；
- 存储器中的系数与相应环节的零点或极点相对应，实验调试非常方便。

3） 并联型结构

$$D(z)=\frac{U(z)}{E(z)}=\gamma_0+D_1+\cdots+D_l$$ 其中D_i为 $\dfrac{\beta_i}{1+\alpha_i z^{-1}}$ 或 $\dfrac{\beta_{i0}+\beta_{i1}z^{-1}}{1+\alpha_{i1}z^{-1}+\alpha_{i2}z^{-2}}$

- 部分分式展开得一阶或二阶环节之和。可用这些低阶环节的编排结构（采用直接型编排实现）进行并联而得。
- 优点
  - 各个通道彼此独立，一个环节的运算误差只影响本环节的输出，对其它环节的输出没有影响。
  - 某一系数产生的误差，只影响相应环节的零点或极点，对其他环节没有影响。

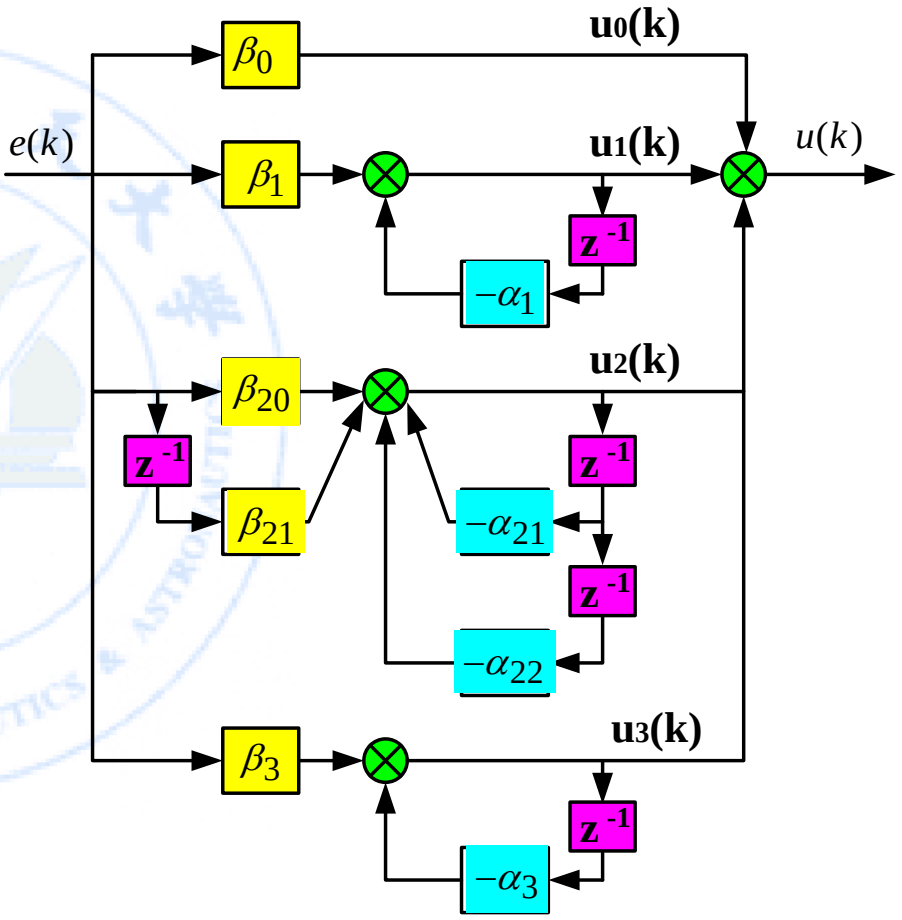


图7-22 并联编排结构

需要说明的问题

- 在没有误差情况下，不管采用哪种编排结构，其对应控制器的静态及动态特性是相同的。
- 存在误差情况下，不同编排结构，其对应控制器的静态及动态特性是不同的，对误差的敏感程度也不相同。
- 不同编排结构，控制器对计算机运算速度及内存容量的要求也不同。

例 控制算法传递函数

$$D(z)=\frac{U(z)}{E(z)}=\frac{0.3+0.36z^{-1}+0.06z^{-2}}{1+0.1z^{-1}-0.2z^{-2}}$$

给出不同编排实现的结构及迭代方程

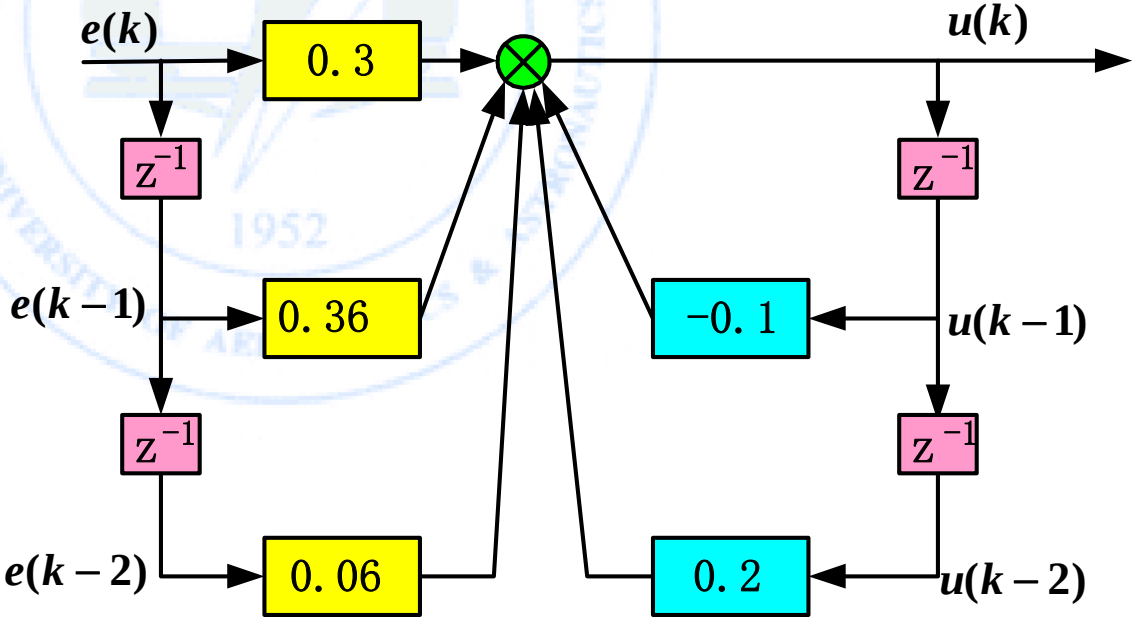
1、直接编排实现——零极型

$$U(z)+0.1z^{-1}U(z)-0.2z^{-2}U(z)=0.3E(z)+0.36z^{-1}E(z)+0.06z^{-2}E(z)$$

$$u(k)+0.1u(k-1)-0.2u(k-2)=0.3e(k)+0.36e(k-1)+0.06e(k-2)$$

u(k) =0.3e(k)+0.36e(k–1)+0.06e(k–2)

–0.1u(k–1)+0.2u(k–2)



(a)

例 控制算法传递函数

1、直接编排实现——极零型

$$D(z)=\frac{U(z)}{E(z)}=\frac{0.3+0.36z^{-1}+0.06z^{-2}}{1+0.1z^{-1}-0.2z^{-2}}$$

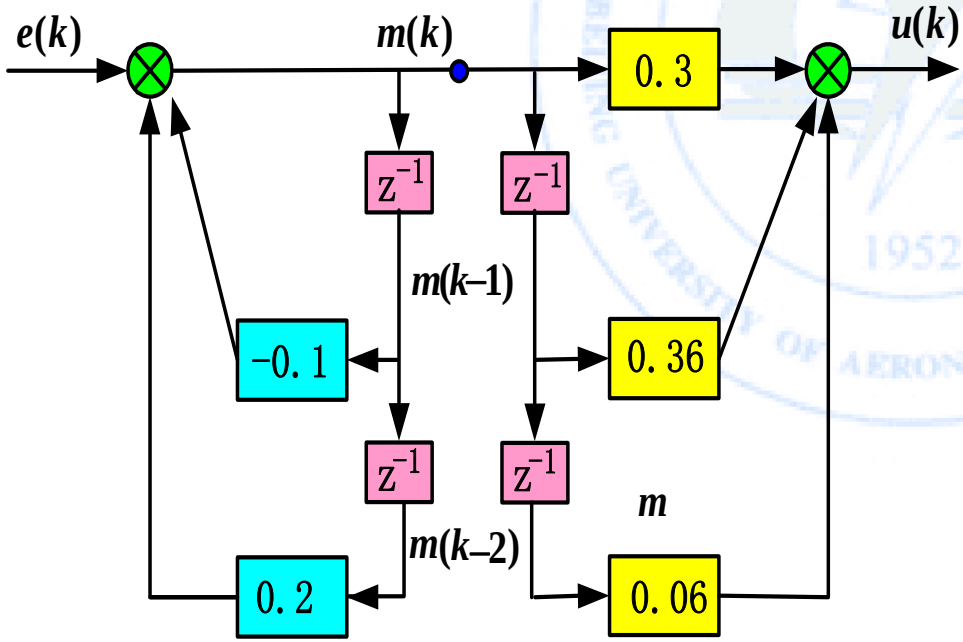
$$D(z)=\frac{U(z)}{M(z)}\cdot\frac{M(z)}{E(z)}=\left(0.3+0.36z^{-1}+0.06z^{-2}\right)\cdot\frac{1}{1+0.1z^{-1}-0.2z^{-2}}$$

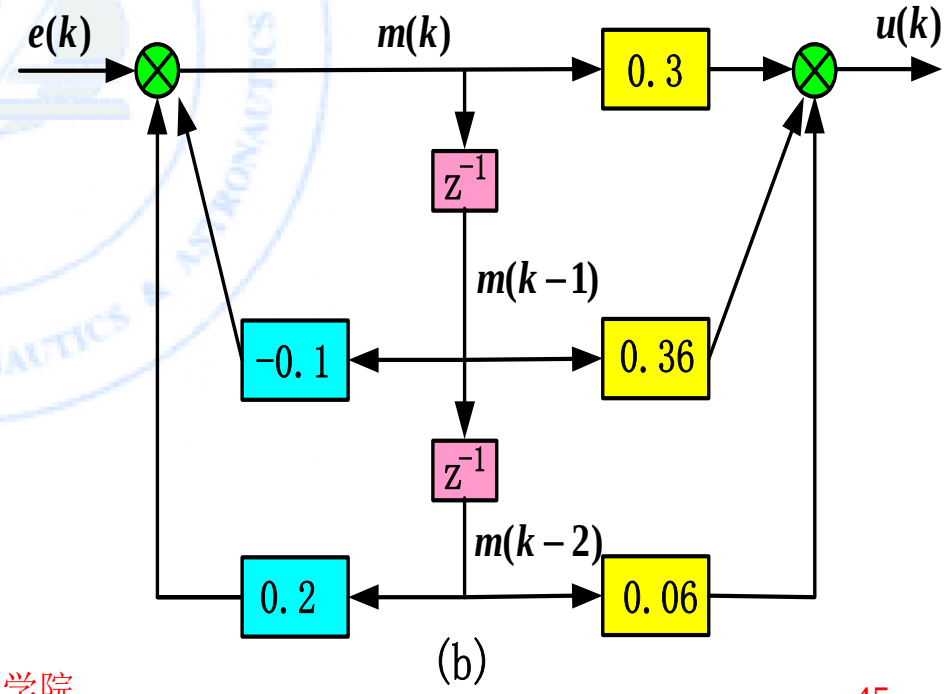
$$M(z)+0.1z^{-1}M(z)-0.2z^{-2}M(z)=E(z)$$

$$U(z)=0.3M(z)+0.36z^{-1}M(z)+0.06z^{-2}M(z)$$

$$m(k)=e(k)-0.1m(k-1)+0.2m(k-2)$$

$$u(k)=0.3m(k)+0.36m(k-1)+0.06m(k-2)$$





例7-2 控制算法传递函数

$$D(z)=\frac{U(z)}{E(z)}=\frac{0.3+0.36z^{-1}+0.06z^{-2}}{1+0.1z^{-1}-0.2z^{-2}}$$

2、串联型编排实现

$$D(z)=\frac{U(z)}{E(z)}=\frac{0.3(1+z^{-1})}{(1+0.5z^{-1})}\frac{(1+0.2z^{-1})}{(1-0.4z^{-1})}=\frac{U_1(z)}{E(z)}\frac{U(z)}{U_1(z)}=D_1D_2$$

D1：

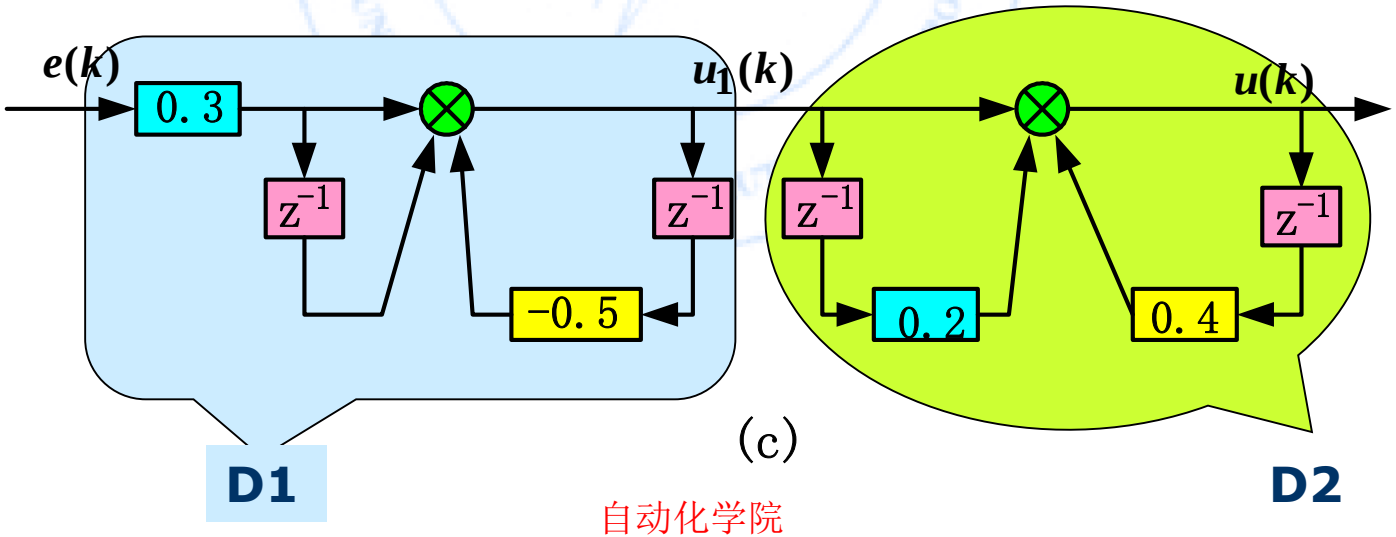
$$U_1(z)+0.5z^{-1}U_1(z)=0.3E(z)+0.3z^{-1}E(z)$$

$$u_1(k)=0.3e(k)+0.3e(k-1)-0.5u_1(k-1)$$

D2：

$$U(z)-0.4z^{-1}U(z)=U_1(z)+0.2z^{-1}U_1(z)$$

$$u(k)=u_1(k)+0.2u_1(k-1)+0.4u(k-1)$$



(c)

例7-2 控制算法传递函数

$$D(z)=\frac{U(z)}{E(z)}=\frac{0.3+0.36z^{-1}+0.06z^{-2}}{1+0.1z^{-1}-0.2z^{-2}}$$

3、并联型编排实现

$$D(z)=\frac{U(z)}{E(z)}=\beta_0+D_1(z)+D_2(z)$$

$$=-0.3-\frac{0.1}{1+0.5z^{-1}}+\frac{0.7}{1-0.4z^{-1}}=\frac{U_1(z)}{E(z)}+\frac{U_2(z)}{E(z)}+\frac{U_3(z)}{E(z)}$$

$$u_1(k)=-0.3e(k)$$

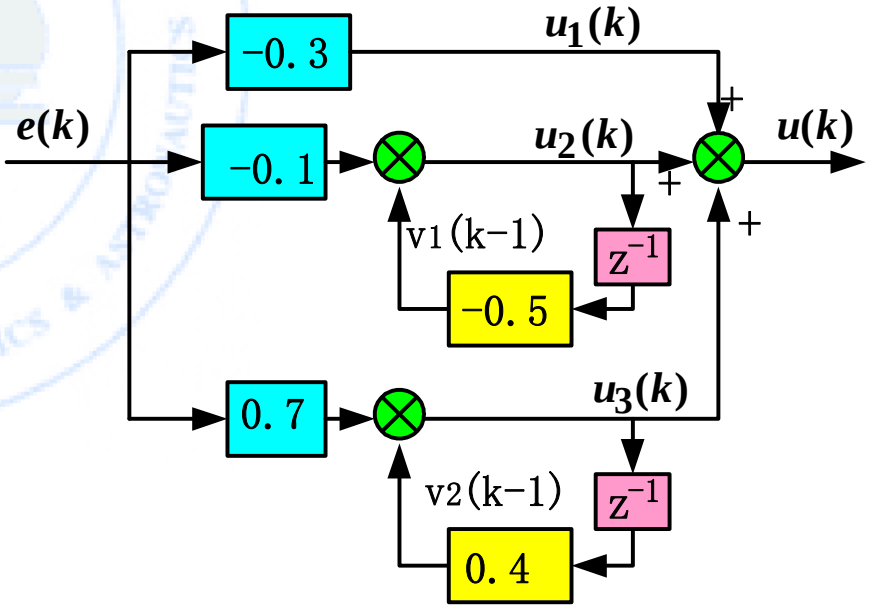
$$U_2(z)+0.5z^{-1}U_2(z)=-0.1E(z)$$

$$u_2(k)=-0.1e(k)-0.5u_2(k-1)$$

$$U_3(z)-0.4z^{-1}U_3(z)=0.7E(z)$$

$$u_3(k)=0.7e(k)+0.4u_3(k-1)$$

$$u(k)=u_1(k)+u_2(k)+u_3(k)$$



(d)

4、比例因子配置

- 原因：定点数要求、D/A前要求
- 原则:

① 绝大多数情况下，使各支路信号不上溢。

② 尽量减少动态信号的下溢值，减小不灵敏区，提高分辨率。

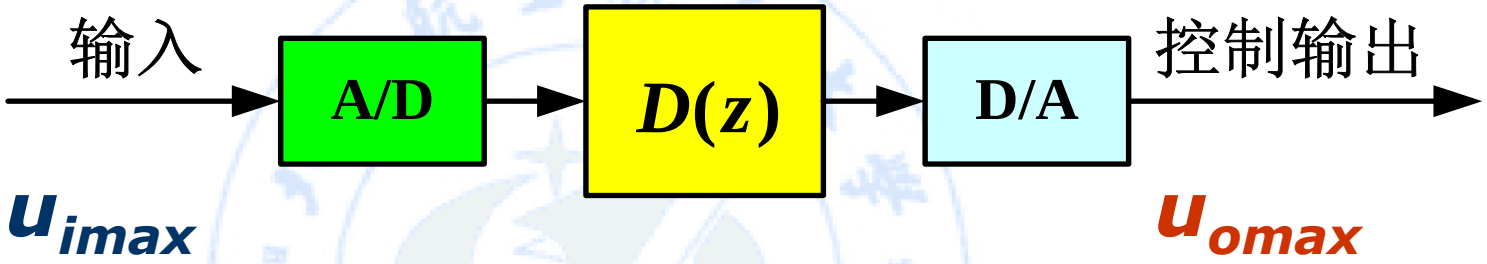
③ 控制算法各支路的比例因子可以采用实际物理量的最大值与计算机代码的最大值之比来确定。采用2的整次幂来缩放。

④ 要保证配置比例因子前后，支路的增益与总的传递特性保持不变，回路的增益保持不变。

⑤ A/D和D/A比例因子的选择，只需使物理量的最大值对应小于1的机器值。

针对A/D和D/A比例因子的考虑和处理

- D(z)=1在理论上表示：控制输出=输入



传递系数$K_{AD}=1/u_{imax}$　　传递系数$K_{DA}=u_{omax}$

故需要在计算机内应配置相应的比例因子$1/K_{AD}$和$1/K_{DA}$。

当控制器增益大于1的情况

$|D(z)|=K*|D1(z)|>1$，（ $K>1$且$|D1(z)|<1$）

处理方法：

①计算机实现增益小于1的控制器D1（z），其余增益移到系统模拟部分完成并设置限幅。

② 将大于1的增益放到最后，并在该增益之前设置数字限幅保护，防止输入信号较大时发生上溢。

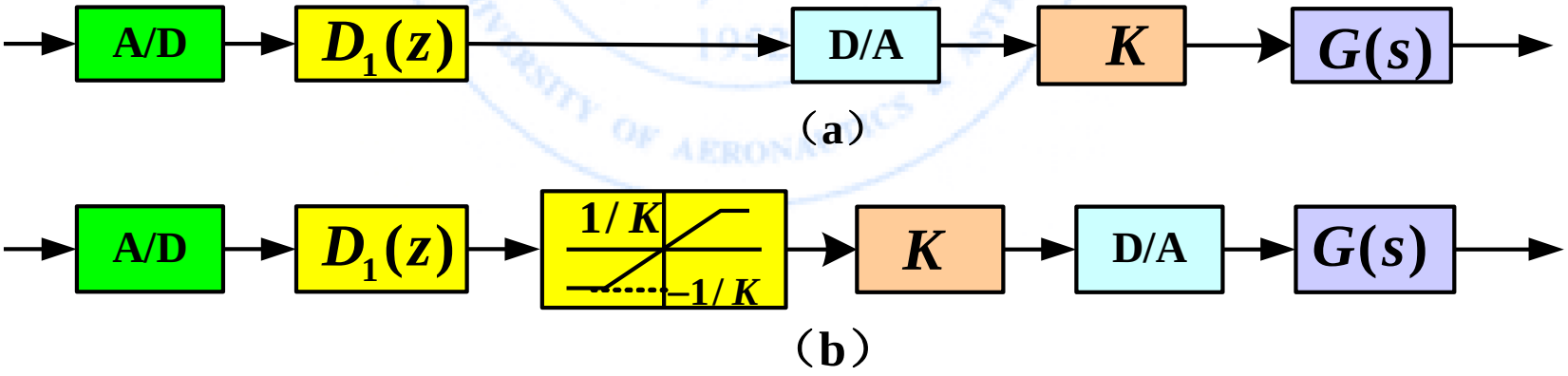
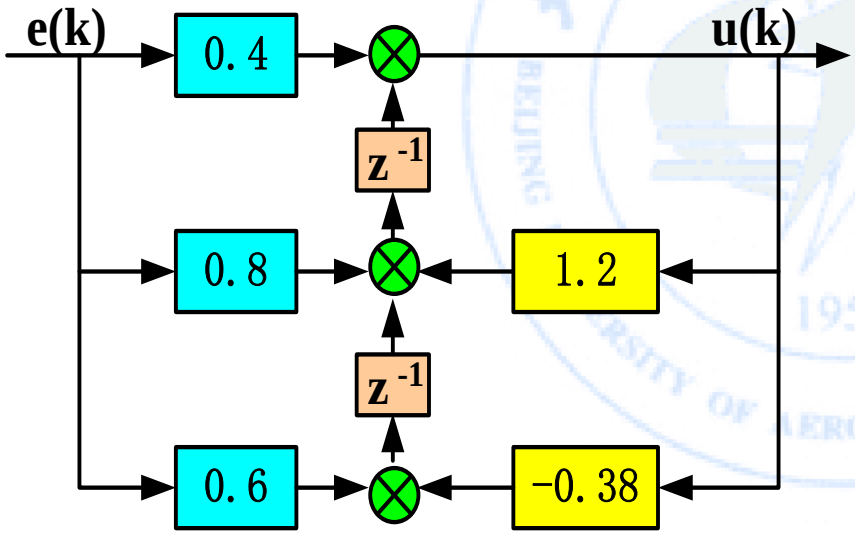


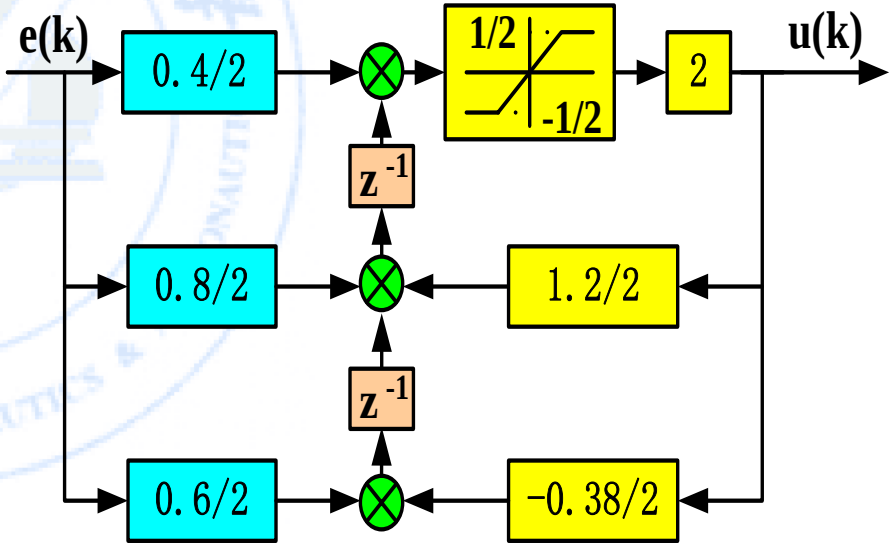
图7-26 数字控制系统控制器增益的分配

例 原则4：要保证配置比例因子前后，支路的增益与总的传递特性保持不变，回路的增益保持不变。

$$D(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}}=\frac{0.4+0.8z^{-1}+0.6z^{-2}}{1-1.2z^{-1}+0.38z^{-2}}$$

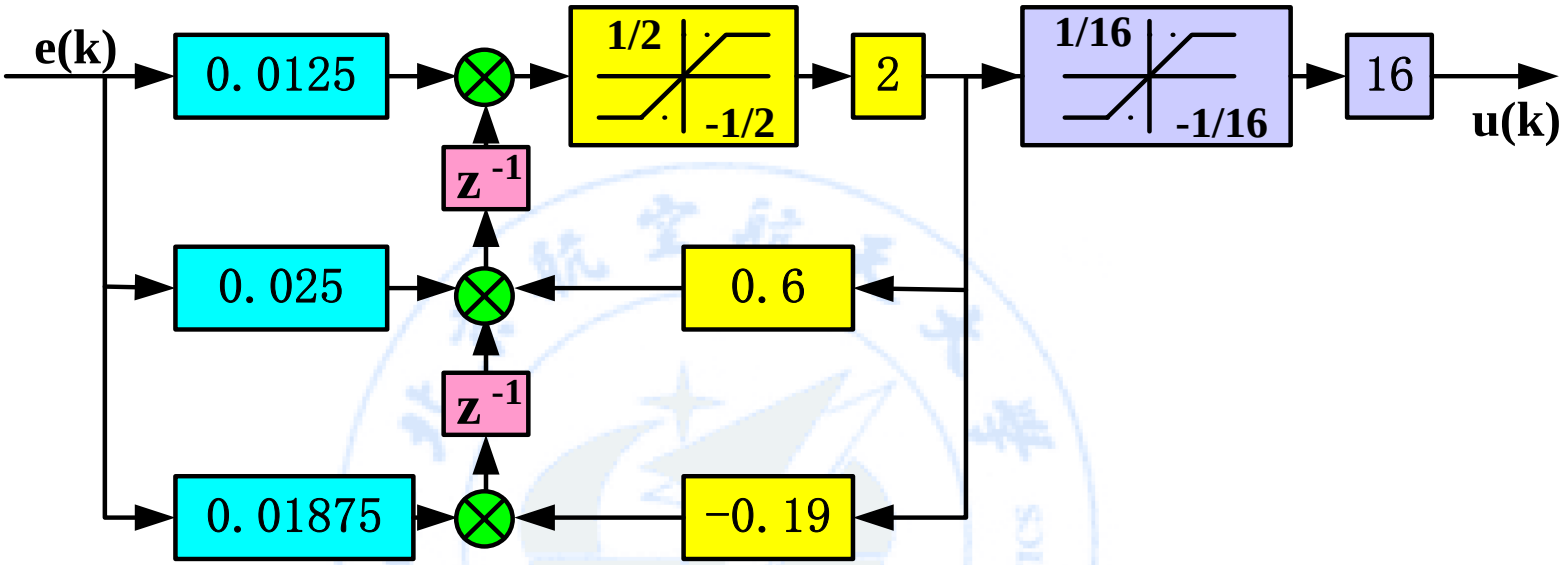


控制算法编排结构图



系数配置后的算法结构图

对整个环节进行配置



$$D(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}}=\frac{0.4+0.8z^{-1}+0.6z^{-2}}{1-1.2z^{-1}+0.38z^{-2}}$$

控制器稳态增益：$D(z)\big|_{\substack{z\to 1\\(t\to\infty)}}=10>1$

控制器高频增益：$D(z)\big|_{\substack{z\to -1\\(t\to\infty)}}=0.039<1$

综合点后防止溢出进行溢出保护

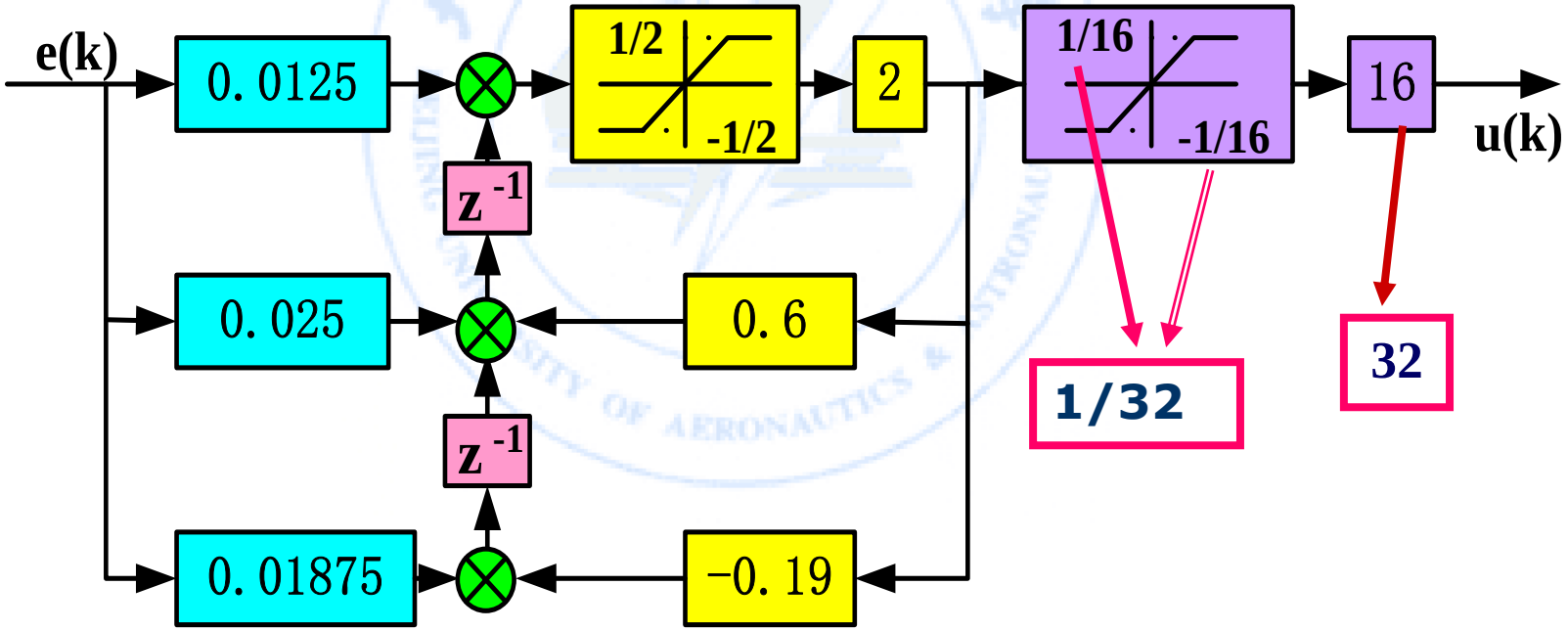
选择比例因子为：$2^4=16>10$

若考虑A/D和D/A的比例因素

A/D: u_{Imax}=10，A/D传递系数$K_{A/D}=1/u_{Imax}=1/10$

D/A:u_{Omax}=5，D/A传递系数$K_{D/A}=u_{Omax}=5$

为了不改变信号的传递关系，应配置比例因子$1/(K_{A/D}*K_{D/A})=2$



考虑A/D和D/A后配置比例因子的算法结构图

控制器算法差分方程实现

极零型 $$D(z)=\frac{U(z)}{E(z)}=\frac{0.3+0.36z^{-1}+0.06z^{-2}}{1+0.1z^{-1}-0.2z^{-2}}$$

算法I： u(k) =0.3e(k) +x1(k–1)

算法 II： x1(k) =0.36e(k) –0.1u(k)+x2(k–1)

x2(k) =0.06e(k) +0.2u(k)

初始化： x1＝0，x2＝0

差分方程实现流程：

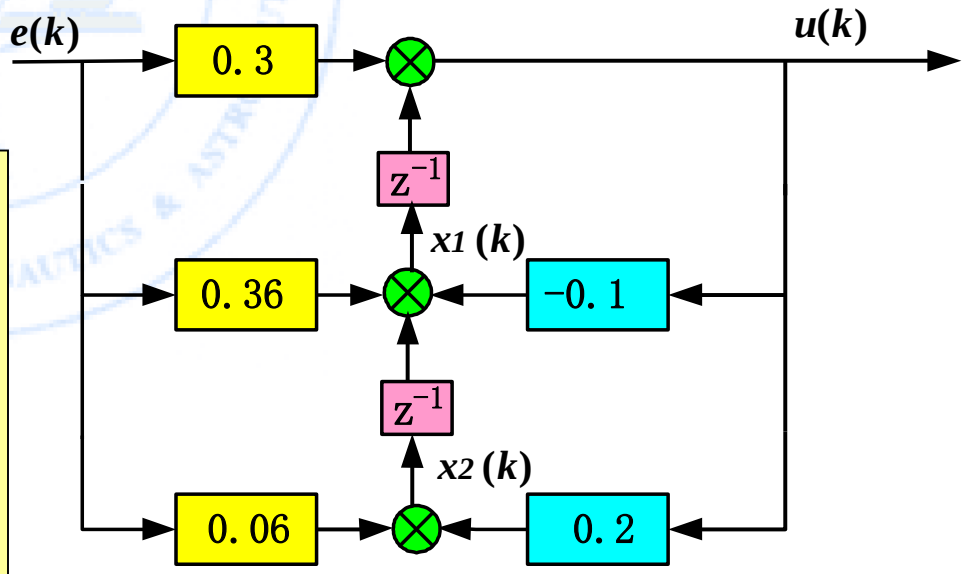
A/D采样e

u =0.3e +x1

D/A输出u

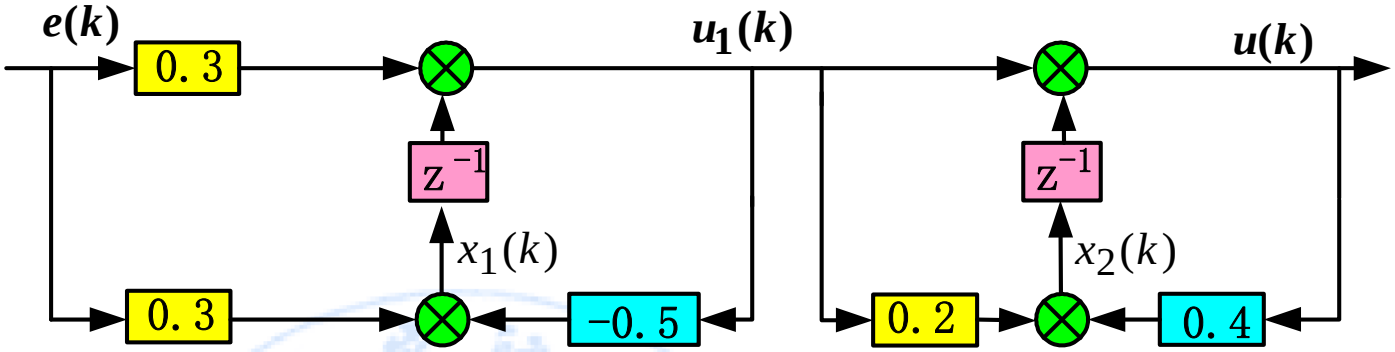
x1 =0.36e –0.1u+x2

X2 =0.06e +0.2u



(a)

2、串联型



初始化： x1＝0，x2＝0

算法I：

u1(k) =0.3e(k)+x1(k–1)

u(k) =u1(k)+x2(k–1)

差分方程实现流程：

A/D采样e

u1 =0.3e+x1

u =u1+x2

D/A输出u

x1=0.3e-0.5u1

x2=0.2u1+0.4u

算法II ：

x1(k)=0.3e(k)-0.5u1(k)

x2(k) =0.2u1(k)+0.4u(k)

例7-4 $$D(z)=\frac{U(z)}{E(z)}=\frac{2(z-0.7)(z-0.8)}{(z-0.9)(z-0.2)}=\frac{2-3z^{-1}+1.12z^{-2}}{1-1.1z^{-1}+0.18z^{-2}}$$

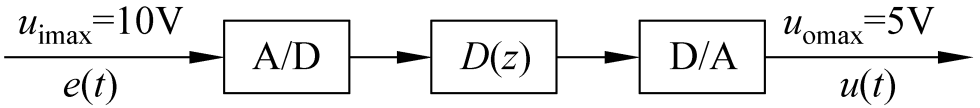


图7-27 某控制器接口图

设主机采用定点小数的补码。

要求：1）画出结构编排图；2）进行适当的比例因子配置；3)写出对应算法差分方程；4）画出相应算法实现流程图。

解：（1）直接编排实现

$$u(k)=2e(k)-3e(k-1)+1.12e(k-2)$$
$$+1.1u(k-1)-0.18u(k-2)$$

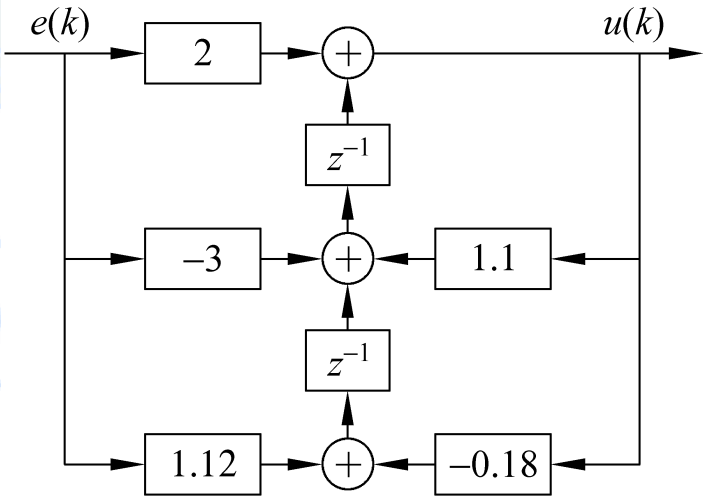


图7-28 控制算法编排结构图

进行比例因子配置

- 考虑系数的情况
  - 注意到：由于主机用定点小数的补码来表示数据，大于1的数据无法在计算机内表示出来。又必须保证每个回路和支路的增益保持不变。
- 确定控制器中间变量的最大值，对整个环节进行配置。

$$\left.\begin{aligned} D(z)\big|_{\substack{z\to 1\\(t\to\infty)}} &= 1.5 > 1 \\ D(z)\big|_{\substack{z\to -1\\(t\to\infty)}} &= 2.6842 > 1 \end{aligned}\right\} \Rightarrow$$

选择比例因子为$2^2=4>3$。

- 考虑A/D和D/A的量程

A/D的量程为10V， A/D的传递系数$K_{AD}=1/10$

D/A的量程为5 V， D/A的传递系数$K_{DA}=5$

为了不改变信号的传递关系，应配置比例因子$1/(K_{AD}*K_{DA})=2$

稍加整理，得配置好比例因子的结构编排图。

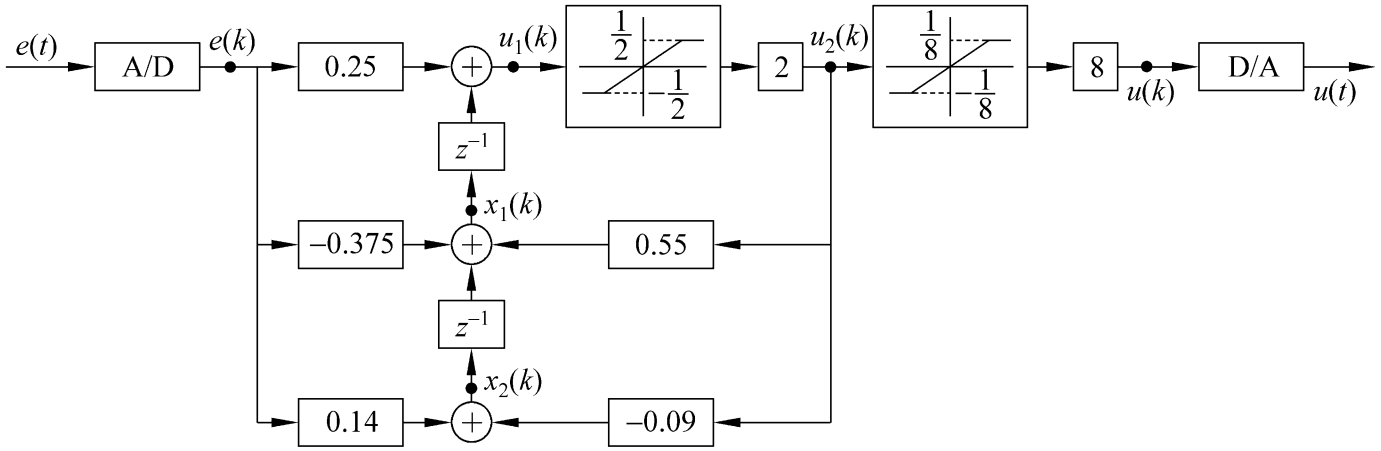


图7-29 整个环节配置比例因子后的直接编排实现结构图

算法I：$u_1(k)=0.25e(k)+x_1(k-1)$

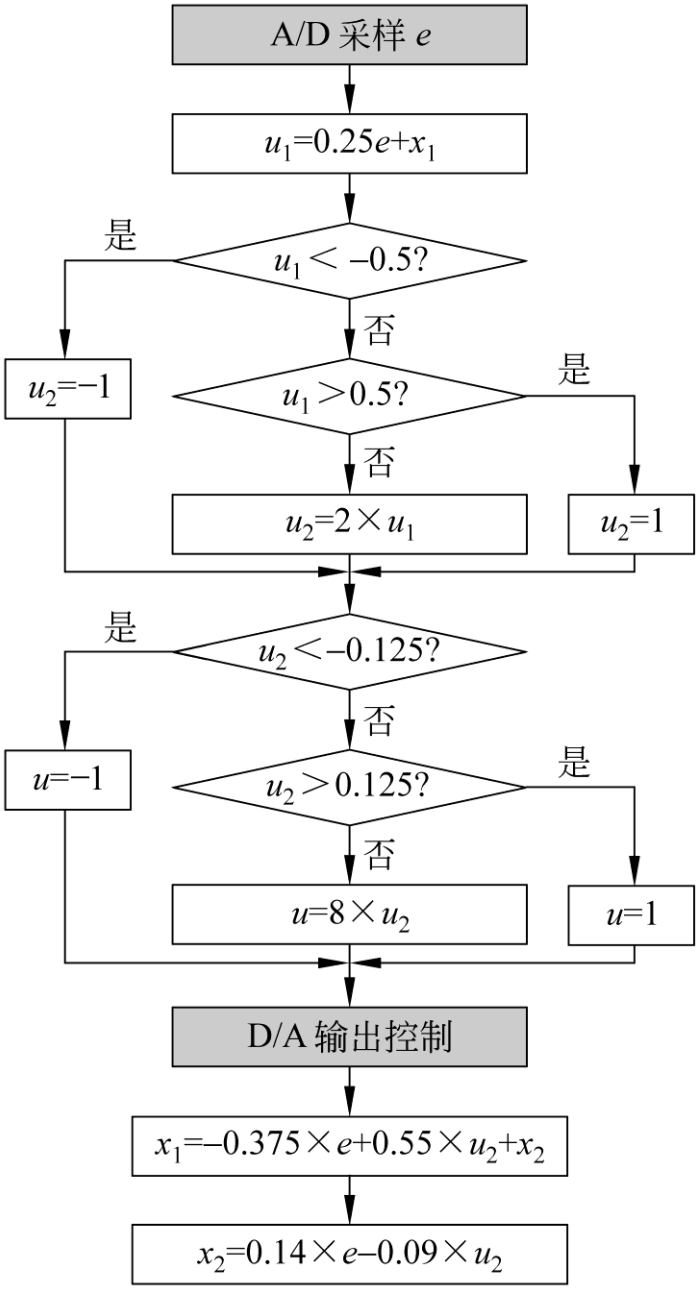
$$u_2(k)=\begin{cases}-1 & ,u_1(k)<-0.5\\ 2*u_1(k) & ,|u_1(k)|<0.5\\ 1 & ,u_1(k)>0.5\end{cases}$$

$$u(k)=\begin{cases}-1 & ,u_2(k)<-0.125\\ 8*u_2(k) & ,|u_2(k)|<0.125\\ 1 & ,u_2(k)>0.125\end{cases}$$

算法II：

$$x_1(k)=-0.375e(k)+0.55u_2(k)+x_2(k-1)$$

$$x_2(k)=0.14e(k)-0.09u_2(k)$$



初始化： $x_1 = 0$, $x_2 = 0$

特点：

无需进行数据传送，而是依靠计算的先后顺序，间接得到和的历次值。

图7-30 算法流程图

为何要讨论各种编排实现方法？

- 理论上一样
- 工程实现不同：
  - 计算延时影响
  - 内存占有不同
  - 量化影响
  - 参数变化——〉特征根会随之变化
- 要求：

  画编排结构图，写差分方程，

  画流程图，进行比例因子配置

第7章 END