第 3 章　嵌入式处理器

It is possible to invent a single machine which can be used to compute any computable sequence. If this machine U is supplied with a tape on the beginning of which is written the S.D ["standard description" of an action table]of some computing machine M, then U will compute the same sequence as M.(1936)

A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal Turing Machine.(1948)

—Alan Turing

在 IEEE 给出的定义中，计算机是包括一个或多个处理单元以及一组外围设备的功能可编程装置，通过内在程序可自动执行特定的算术和逻辑运算。那么从广义上讲，以数字处理单元为核心设计的通用计算机与形形色色的嵌入式计算装置就都可纳入计算机的范畴。随着嵌入式系统技术在越来越多的行业中的应用以及万物智联时代的开启，嵌入式处理器的发展日益呈现出架构多元、类型丰富、快速演化等特点与趋势。那么，总结并厘清不同类型嵌入式处理器的体系结构与功能特性，就成为学习和设计嵌入式系统的重要环节。

3.1　处理器模型与逻辑体系

3.1.1　处理器基本组成模型

1. 核心组件

中央处理单元(CPU)也称处理器，是所有数字计算装置的“大脑”。任何数字计算装置的计算体系及其软硬件设计都要围绕处理器展开。为了更好地学习和理解不同形式的嵌入式处理器并对比其与通用处理器(General Purpose Processor，GPP)的异同，这里首先回顾处理器的基本工作过程。图 3.1 给出了 CPU 指令执行过程抽象的运行逻辑。由图 3.1 中标识的步骤可知，系统上电后，处理器中的控制单元(Control Unit，CU)会根据程序计数器(Program Counter，PC)的值读取和解码指令，然后交由执行单元(Execute Unit，EU)进行算术和逻辑运算并进一步输出结果。

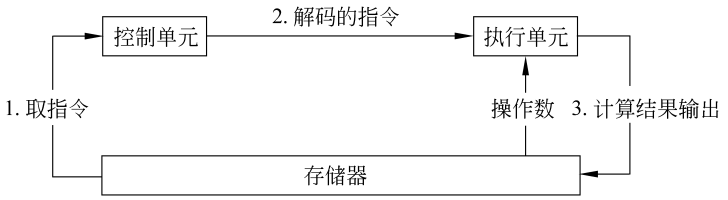


图 3.1　CPU 指令执行过程抽象的运行逻辑

由此，可以给出一个简化的 CPU 核心组件模型，如图 3.2 所示。在该模型中，控制单元

由程序计数器(PC)、指令寄存器(Instruction Register,IR)和指令解码器(Instruction Decoder,ID)等组件构成,其通过高速缓存提高访问速度,同时采用堆栈来解决中断、函数调用返回时的现场恢复问题。执行单元由算术逻辑单元(Arithmetic Logic Unit,ALU)和数据寄存器(Data Register,DR)、功能寄存器(Function Register,FR)构成。一个控制单元和一个执行单元即可构成一个完整、独立的指令执行引擎,称之为**计算核**(core)。处理器中计算核的数量则是指执行单元的数量,且每个执行单元至少需要一个控制单元的支持。

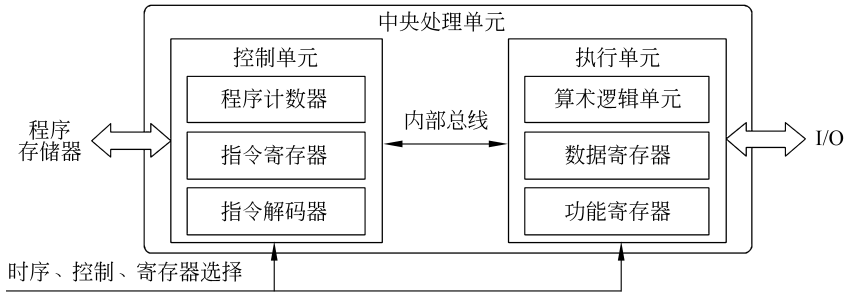


图 3.2 CPU 核心组件模型

一条指令的执行分为多个阶段,在每个阶段并不会占用执行单元的全部资源。那么,将空闲资源提前用于下一条指令的读取与执行,就可以大幅提升处理器的资源利用率、吞吐量和执行效率。因此,在多核处理器出现之前,部分处理器体系中就采用了增加控制单元和优化流水线的方式,以此实现虚拟的多核计算环境,称为**单核多任务计算模式**。例如,Intel 奔腾 4 等处理器中采用的超线程便是类似技术,可以使单核 CPU 的计算效能提升 20%～30%。而在多核处理器中,片内多个独立的计算核硬件同时运行,真正实现指令、任务级的并行,即**多核多任务计算模式**。

2. 指令、汇编语言与指令集

指令是可被处理器识别、译码和执行的编码,是连接程序与处理器的桥梁。在计算系统中,指令由表示运算方式的机器操作码(opcode)以及零个或多个地址码组成,而可执行程序则是一组机器码的顺序存储序列。总体上,指令可被分为使用和改变寄存器值的数据处理指令、给操作对象赋值的数据传送指令、用于分支和连接控制的控制流指令以及程序状态寄存器处理指令、异常产生指令等。为了增强指令的可读性以及便于基于指令的软件设计,进一步为指令设计了对应的编程符号,也就是**汇编语句**,进而形成了**汇编语言**。对于不同系列的处理器,其操作码、指令格式及汇编语句通常不同,但对于同一系列的处理器,其操作码、指令格式及汇编语言通常相同或相似。

一系列机器操作码以及由特定处理器执行的基本指令构成了**指令集体系结构**(Instruction Set Architecture,ISA)。任何处理器都拥有至少一套指令集,指令集在一定程度上也反映了处理器的逻辑体系。指令集是计算机体系结构中与程序设计相关的部分,与基本数据类型、寄存器、寻址模式、存储体系、中断、异常处理以及外部 I/O 操作等密切相关。根据指令集功能强弱、完备性及寻址方式,通常将指令集分为复杂指令集和精简指令集,相应的计算机称为复杂指令集计算机(Complex Instruction Set Computer,CISC)和精简指令集计算机(Reduced Instruction Set Computer,RISC)。与 CISC 相比,RISC 具有以下特点:指令长度与执行周期固定,指令类型较少,硬件更简洁,寄存器更多,寻址模式少,易于实现流水线以

及编译器更复杂，典型的 RISC 有广泛应用于各类计算装置的开源指令集 RISC-V[①]。早期的 x86 处理器采用复杂指令集；而常见的嵌入式处理器体系中大都采用了精简指令集，如 ARM、MIPS、PowerPC、SPARC 等。另外，指令集与处理器中执行指令的微架构并非必须一一对应。即使同一系列的处理器，其微架构也会存在差异，但一般基于同一指令集或其不同子集进行设计。当然，同一微处理器也可以同时支持多套指令集。例如，ARM 处理器可采用 32 位 ARM 指令集、16 位 Thumb 指令集或者 16 位与 32 位指令并存的 Thumb-2 指令集等，CPSR（Current Program Status Register，程序状态寄存器）的第 5 位（位 T）决定了 ARM 处理器内核执行的是 ARM 指令流还是 Thumb 指令流。

下面对几种典型处理器体系结构进行简要说明。

1）80x86 指令格式

80x86 指令格式如图 3.3 所示。指令前缀分为 4 组，每组用一字节编码。操作码表示该指令对应的具体执行动作，其长度可为 1 字节、2 字节或 3 字节。ModR/M 字节为寻址模式说明符，共有 32 个值，表示 8 个寄存器与 24 种寻址模式。ModR/M 中的 Reg/Opcode 表示寄存器号或者额外的 3 位指令码，其具体含义依赖于基本指令码，而 ModR/M 的位 5 表示第一操作数的寻址方式等。SIB 字节用于补充某些 ModR/M 字节表示的寻址模式，且仅在 ModR/M 有效时生效。

指令前缀	操作码	ModR/M (可选)	SIB (可选)	偏移	立即数
4字节	1/2/3字节	0~2:R/M 3~5:Reg/Opcode 6,7:Mod	0~2:Base 3~5:Index 6,7:Scale	0/1/2/4字节	0/1/2/4字节

图 3.3　80x86 指令格式

例如，80x86 处理器中将立即数写入寄存器的汇编指令格式为“MOV Reg，Imm”，从其主要指令表（Main Instructions）[25]可以查到，该指令对应的操作码为 1011wrrr。由于汇编语句“MOV DX，1234H”中目标寄存器为 DX 且立即数为一个字长，因此，可从 80x86 指令与操作码表中查出“w＝1，rrr＝010”，从该条指令翻译得到的可执行机器码为 BA3412（对应二进制码 10111010000110100000010010）。其中，立即数 1234H 采用了大端模式[②]存储。同理，可从汇编代码翻译得出其他机器指令的机器码。

2）ARM 指令格式

ARM 架构中有两种主要的指令体系，即 32 位的 ARM 指令以及 16 位的 Thumb 指令。Thumb 指令体系是 ARM 指令体系的压缩形式的子集，它并不完整，但可以减少代码量。以 32 位的 ARM 指令为例，典型编码格式如图 3.4 所示。其基本格式为

```
<opcode>{<condition>}{S}<Rd>,<Rn>,{<shifter_operand>}
```

其中，opcode 为操作码，condition 为执行条件（如 EQ、NE、CS/HS 等），S 表示指令操

① 2010 年，加利福尼亚大学伯克利分校研究团队需要设计一款新的 CPU，但 ARM、MIPS、x86 等商用指令集架构均涉及知识产权或高昂成本，为此自行设计了这套全新的 RISC-V 指令集。其特点在于：免费开源，技术中立；采用简洁至上的设计哲学，架构简单，基础指令仅 47 条；易于移植、可实现模块化设计。

② 数据在存储器中有两种主要存放模式。若数据的高字节保存在存储器的低地址中，而数据的低字节保存在高地址中，称为大端（Big-Endian）模式或 MSB（Most Significant Byte）模式；若数据的低字节保存在存储器的低地址，而高字节保存在高地址，则称为小端（Little-Endian）模式或 LSB（Least Significant Byte）模式。

作是否影响当前程序状态寄存器(CPSR)的值,Rd 和 Rn 分别为目标寄存器和第一个操作数的寄存器,shifter_operand 为第二个操作数;同时,< >表示该项在指令中不可或缺,{ }表示指令的可选项。同样,查询 ARM 指令表可知汇编语句“MOV R1, # 0x64”和“MOVEQ R0,R1”的机器码分别为 E3A01064 和 01A00001。

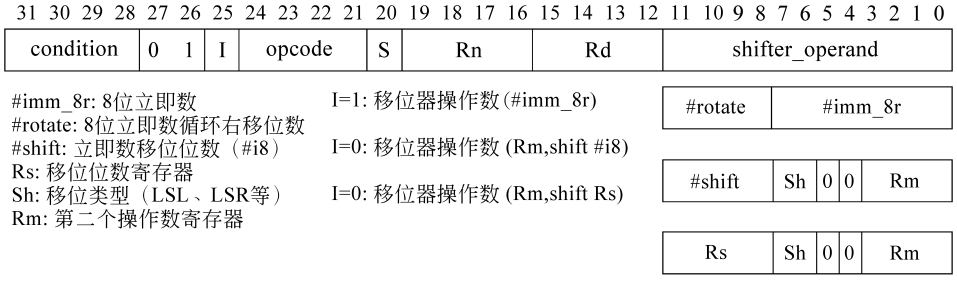


图 3.4 典型的 32 位 ARM 指令的典型编码格式

3) MIPS32 指令格式

图 3.5 为 3 个基本的 32 位 MIPS 指令格式,R 为寄存器指令格式,I 为立即数指令格式,J 为跳转指令格式。其指令格式中,opcode 为操作码,所有 R 型指令的 opcode 值均为 0;rs、rt 分别为第一个、第二个源操作数所在的寄存器编号,rd 为目的操作数的寄存器编号;shamt 用于指定移位指令进行移位操作的位数,在非移位指令中设为 0。同样,根据 MIPS32 指令表,可以翻译出汇编指令的机器码,如立即数加法指令“addi $r1, $r2,200”的机器码为 202200C8。

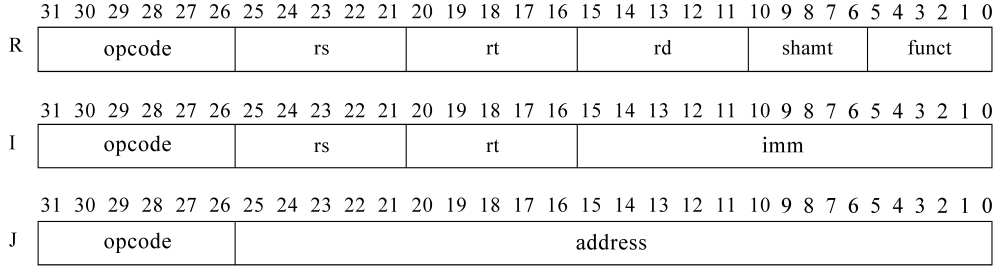


图 3.5 3 个基本的 32 位 MIPS 指令格式

4) RISC-V 指令格式[26]

基本的 RISC-V 指令具有固定的 32 位长度,且必须进行 32 位边界对齐。但是,标准的 RISC-V 编码机制设计扩展了变长的指令,每条指令长度为 16 位指令字的整数倍且必须在 16 位边界处进行对齐,如图 3.6 所示。图 3.7 给出了 RV32I 指令集中的 4 种基本指令格式。RV32I 中的 32 是指 31 个通用寄存器的宽度为 32 位,I 表示整数操作。R 类型为寄存器-寄存器操作指令,I 类型为短立即数和访存 load 指令,S 类型为访存 store 指令,U 类型为长立即数操作指令。其中,rs、rd 分别代表源寄存器和目的寄存器,imm 表示立即数,funct 代表具体操作的类型,如加、减、比较等。类似地,RISC-V 规范还定义了 RV64I、RV128I 和用于嵌入式系统的 RV32E 等基础指令集以及 RV32/64G① 等扩展指令集。

① G 表示通用指令集,包括了 IMAFD 特性。其中,I 表示整数,M 表示整数乘法,A 表示原子内存操作,F 和 D 分别表示单精度和双精度浮点数。

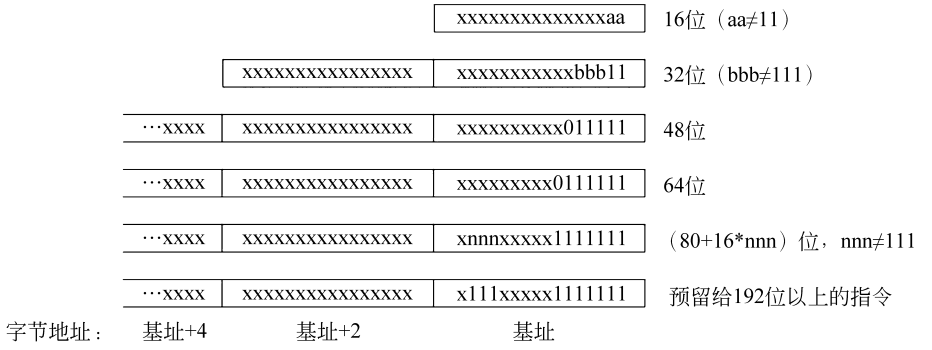


图 3.6　RISC-V 指令长度编码

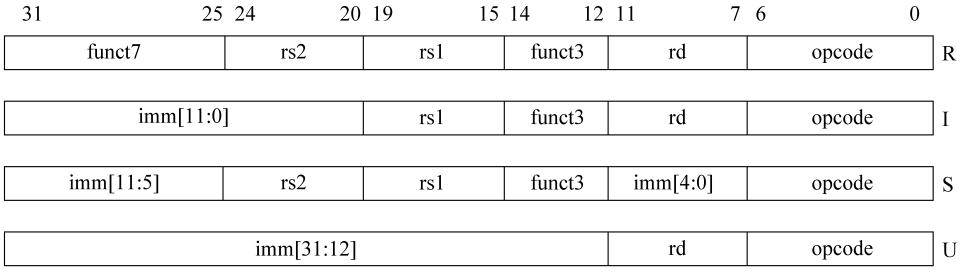


图 3.7　RV32I 指令集中的 4 种基本指令格式

需要说明的是，RV32E 采用与 RV32I 相同的指令集编码，但整数寄存器的数量被减少为 16 个，以减小核的体积和功耗。与这一改动相对应的是，需要采用一个不同的调用约定以及应用程序二进制接口（Application Binary Interface，ABI）。特别地，RV32E 仅与软浮点调用约定一起使用。相关组织也正在考虑设计跨 RV32I 和 RV32E 运行的应用程序二进制接口等。总之，RV32E 目前仍是一个处于演化过程中的架构。

3.1.2　典型处理器架构

1. 冯·诺依曼体系结构

冯·诺依曼体系结构（von Neumann architecture）是一种将程序指令和数据存储在同一存储器，并使用同一套总线传输的计算机体系结构，用于实现通用图灵机，也被称为冯·诺依曼模型（von Neumann model）或普林斯顿体系结构（Princeton architecture）。在冯·诺依曼体系结构中，一条指令的执行过程包括取指令、指令译码、执行指令，强调了顺序执行指令的特点。由于程序指令和数据分别存储在同一存储器的不同物理位置，因此，程序指令和数据具有相同宽度，且取指令和取操作数要以分时复用的方式在同一套总线上进行。冯·诺依曼体系结构的处理器是设计冯·诺依曼体系结构计算机的基础。对于任一冯·诺依曼体系结构处理器而言，其主要包括如下组件：

（1）一个存储器，实现程序指令和数据的存储。

（2）一个控制器，控制程序的运行。

（3）一个运算器，用于完成算术和逻辑运算。

（4）输入和输出设备，用于进行人机交互和通信。

早期的微处理器大多采用冯·诺依曼体系结构设计，典型的如 Intel 公司的 x86 系列微

处理器。在嵌入式处理器领域，TI公司的MSP430处理器、MIPS公司的MIPS处理器也都采用了冯·诺依曼体系结构。ARM7处理器采用的ARMv4T结构就是冯·诺依曼体系结构[27]，如图3.8所示，其核心逻辑包括32位的运算器(ALU、乘法器、移位器)、控制程序运行的控制器及一组寄存器，同时具有一套32位的地址总线和32位的数据总线。

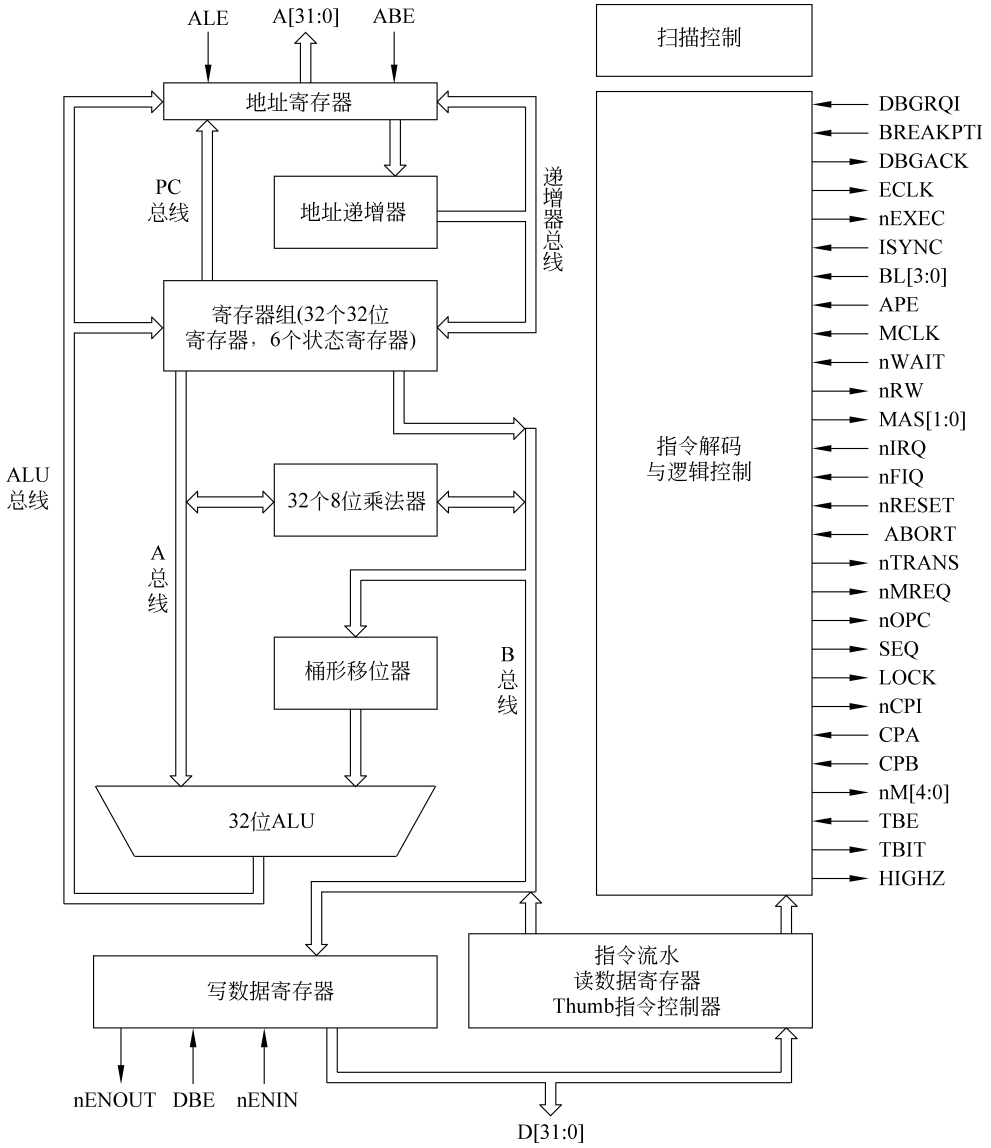


图3.8 ARM7处理器采用了冯·诺依曼体系结构

复用总线进行串行的指令、数据访问会导致冯·诺依曼瓶颈(von Neumann bottleneck)，数据吞吐量将制约整个计算系统的性能，而且在处理器速度越快时问题越是严重。约翰·巴科斯①曾在图灵奖获奖感言《能否将编程从冯·诺依曼风格中解放出来?》

① 约翰·巴科斯(John Warner Backus，1924—2007)，美国计算机科学家，FORTRAN语言发明人之一，提出定义形式语言语法的记号法BNF，发明函数级编程概念及FP语言，1977年获图灵奖。

（*Can Programming Be Liberated from the von Neumann Style?*）中曾提到：**“瓶颈不仅是数据传输的瓶颈，更重要的也是使我们的思考方法局限在‘一次一字’模式的智力瓶颈，不能鼓励我们关于任务本身更广泛地思考。由此，编程工作主要是大量字符数据通过冯·诺依曼瓶颈的规划和细节描述，字符传输更关心的是如何找出数据而不是数据本身的特征。”**近年来，高速缓存、多线程、RAMBUS以及分支预测算法等技术已被广泛应用于缓解冯·诺依曼瓶颈问题。

2. 哈佛体系结构[①]

哈佛体系结构（Harvard architecture）是指采用独立的物理存储器分别存储程序指令与数据，并通过两套总线进行独立传输，是冯·诺依曼体系结构的延伸。哈佛体系结构的计算机由CPU、非易失的程序存储器和易失的数据存储器组成。由于程序存储器和数据存储器独立编址且采用独立的总线，该体系结构可以提供更大的存储器带宽，实现更为高效的指令流水机制，也使得数据的移动和交换更加方便。哈佛体系结构的这一优点使其非常适合设计运算量大、运算速度和数据吞吐量要求高的数字信号处理器。

目前，采用哈佛体系结构的微处理器非常多。例如，Intel公司的MCS[②]-51系列处理器内部有独立的程序存储器和数据存储器，类似的还有Microchip的公司PIC系列、摩托罗拉公司的MC68和MPC860系列、Zilog公司的Z8系列、ATMEL公司的AVR系列以及ARM公司的ARM9、ARM10、ARM11处理器等。图3.9、图3.10分别为采用哈佛体系结构的ARM Cortex-M3嵌入式处理器和摩托罗拉MPC860嵌入式处理器。由图3.9可知，Cortex-M3处理器采用两套独立的地址总线和数据总线访问外部程序存储器和数据存储器；

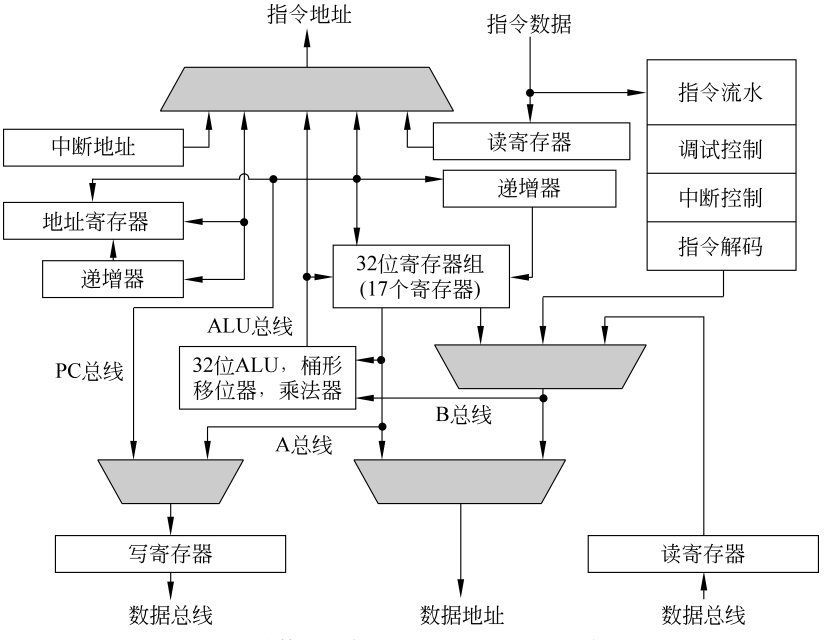


图3.9　采用哈佛体系结构的ARM Cortex-M3嵌入式处理器

① 哈佛体系结构源于哈佛大学1944年研制的Harvard Mark Ⅰ机电型可编程计算设备。

② MCS：Micro-Control System，微控制系统的缩写。

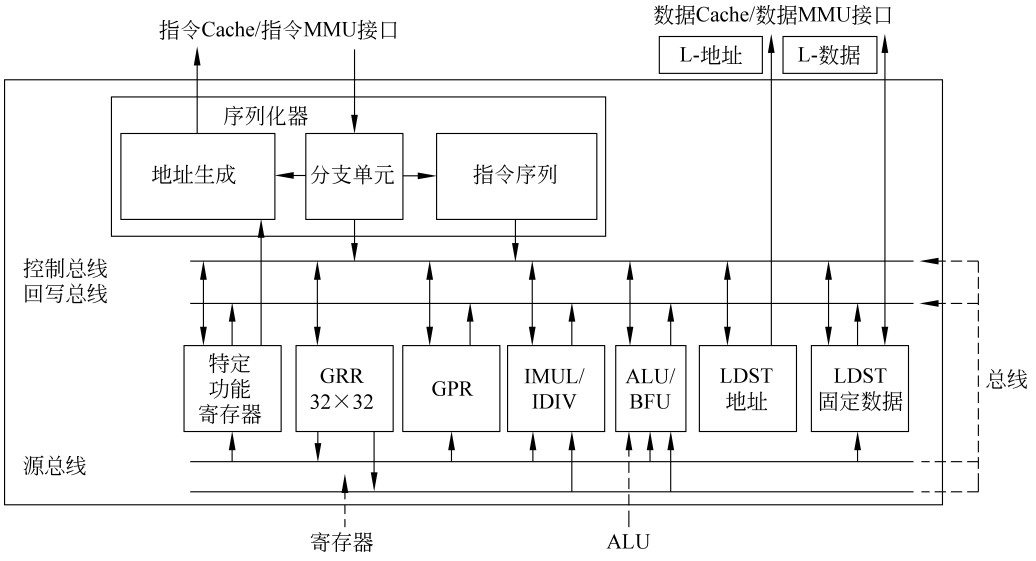


图 3.10 采用哈佛体系结构的摩托罗拉 MPC860 嵌入式处理器

MPC860 处理器中不但具有两套存储器和总线，还采用了独立的指令 Cache、指令 MMU[①] 和数据 Cache、数据 MMU。

当然，哈佛体系结构也存在非常明显的缺点。哈佛体系结构较冯·诺依曼体系结构需要更多的地址线和数据线，这使得处理器的设计以及外设连接与扩展的难度都大为增加。因此，在复杂处理器的设计中，常常通过在处理器内部使用独立的数据 Cache 和指令 Cache 提高指令执行的效率，即，在处理器内部逻辑中使用哈佛体系结构，而外部仍然使用冯·诺依曼体系结构。

图 3.11 给出了一种改进的哈佛体系结构。它保持了哈佛体系结构的独立存储特点，为程序指令和数据采用两个独立的存储单元，同时采用了分时复用的地址总线和数据总线。显然，改进的哈佛体系结构在存储特性上与哈佛体系结构一致，实现了指令、数据的独立存储；而在总线特性上，它则与冯·诺依曼体系结构相似，降低了硬件逻辑复杂性以及对外围设备连接和处理的要求。在部分改进型哈佛体系结构处理器中，还在两个存储器之间增加了公共总线，允许数据存放在程序存储器中，提高了处理器访问数据的灵活性。在改进型哈

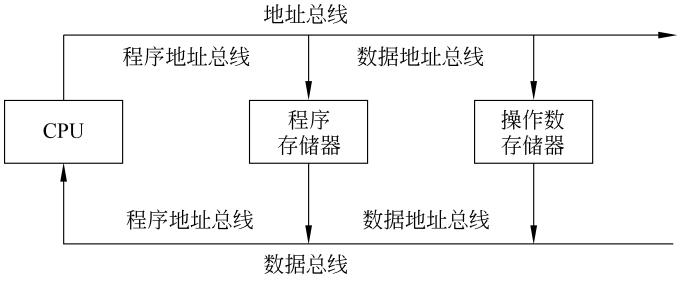


图 3.11 一种改进的哈佛体系结构

① MMU：Memory Management Unit，存储管理部件，CPU 中用于实现虚拟内存管理的组件，提供硬件机制的内存访问权限检查。

佛体系结构基础上，进一步增加数据总线的数量，还可以形成更为先进的改进型哈佛体系结构。例如，TI公司的TMS320VC5404/5407 DSP处理器的内核逻辑中就采用了一套独立的程序存储总线和三套数据存储总线。

另外，对于多核处理器还有**超标量体系结构**（Super Scalar Architecture，SSA）、**超长指令字**（Very Large Instruction Word）等支持指令并行的处理器体系结构。在保证结果正确的前提下，超标量体系结构的处理器采用了复杂的硬件逻辑，可以非顺序方式发射、执行多条指令，典型方式有顺序发射顺序完成、顺序发射乱序完成、乱序发射乱序完成等。在超长指令字体系结构中，更为复杂的编译器在编译程序时可以将多条可并行执行的指令组合成一条长的指令，执行时再将长指令分解并将各个部分分配到相应的执行单元上执行。

3.2 嵌入式处理器的类型及特点

作为微处理器家族的一个重要分支，嵌入式处理器自20世纪70年代中后期诞生以来已经经历了飞速的发展，而且其体系结构和形态也在随着嵌入式系统应用领域的丰富而不断演化和细分。如前所述，集成电路技术是微处理器设计的基础，而微处理器的出现则是嵌入式系统发展的一个重要里程碑。

自20世纪70年代以Intel 4004和4040为代表的第一代微处理器出现以来，微处理器在体系结构的演化、工艺的发展（如COMS、HCMOS）、集成度和处理速度的提升、低电压节能运行等方面取得了令人叹为观止的进步。例如，早期Intel 4004仅是具有2200多个晶体管、每秒执行90 000次指令操作的4位微处理器，而目前新型的微处理器则已经单片集成数十亿个晶体管，具有32位、64位字宽，执行速度可达数千MIPS[①]。目前，嵌入式处理器范畴呈现出性能各异、用途多样、成百上千种嵌入式处理器并存的局面。本节将对主要类型的嵌入式处理器及其特性进行归纳和分析。

3.2.1 嵌入式微控制器

单片微型计算机（Single Chip Microcomputer，SCM，简称**单片机**）是一种将CPU、RAM、ROM及I/O等主要功能部件集成于一块集成电路芯片上的处理器类型，其随着Intel公司1971年制造出的4004微处理器芯片而诞生，几十年来已演化出很多架构并在各个领域得到广泛应用。单片机又称**微控制器**（MCU），是为快速构建控制类型的设备提供的一种核心控制单元。早期的代表性微控制器有National Semiconductor公司的4位COP 400、Intel公司的8位8048和Fairchild公司的8位F8微控制器等。经过几十年的发展，现在各种类型的8位、16位、32位微控制器已被广泛使用，典型的有源自Intel公司的MCS-51系列、意法半导体（ST）公司基于ARM Cortex-M核的STM32系列、微芯科技公司的PIC32、ATMEL公司的AVR、Imagination公司的MIPS M6200以及M6250、IBM公司的PowerPC 440GP等。

从构成的角度，微控制器中除了CPU以外，在片内通过内部总线连接、集成了丰富的外设资源，如程序/数据存储器、定时器/计数器（Timer/Counter）、中断控制器以及串行/并行通信接口与I/O接口等，为快速设计系统提供如图3.12所示的处理单元。在实际中，面

① MIPS（Million Instructions Per Second，每秒百万条指令），是评价处理器速度的重要指标。

向不同领域、不同层次应用的微控制器，其内部集成的组件类型和数量存在差异。以 MCS-51 系列 8 位微控制器为例，其内部逻辑与图 3.12 基本一致，较为简单。而 PIC 等 32 位微控制器等还可能提供了 ADC（Aualog-to-Digital Conversion，模数转换）、PWM（Pwlse Width Modulation，脉冲宽度调制）模块，以及更为丰富的 I/O 接口。图 3.13 给出了 ATMEL 公司的 32 位微控制器 AT91SAM7L 的逻辑结构，其以 ARM7 TDMI 处理器为核心并集成了多种类型的逻辑组件与外设[28]。微控制器内部集成的资源越丰富，设计硬件时所需的电路扩展就越少，就越简洁高效。

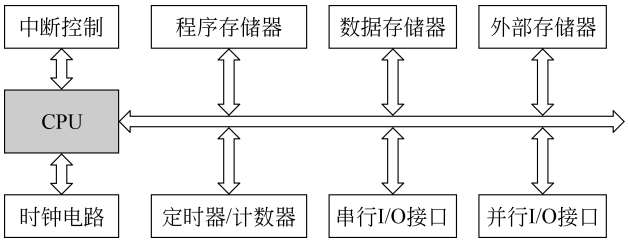


图 3.12　微控制器基本逻辑结构

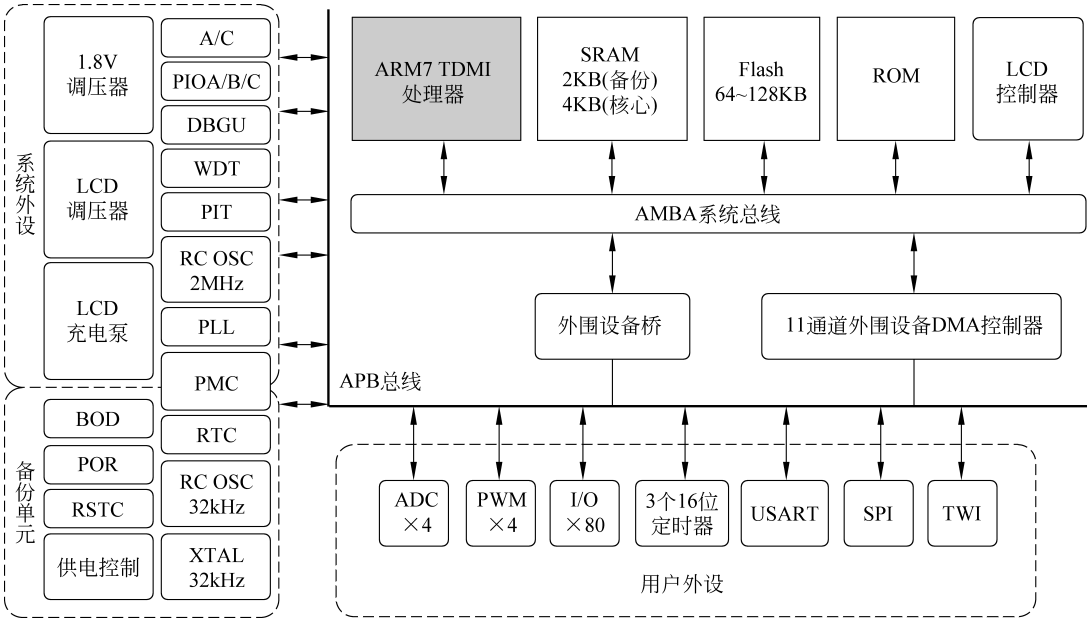


图 3.13　32 位微控制器 AT91SAM7L 的逻辑结构

在分类上，可根据存储及总线结构将微控制器分为冯·诺依曼体系结构和哈佛体系结构。微控制器根据用途又可分为通用型和专用型。通用型微控制器具有开放的开发资源，允许用户根据应用需要进行软硬件逻辑的扩展和定制；而专用型微控制器则更类似于一个专用芯片，如打印机控制器等。另外，片内程序存储器类型也是对微控制器进行分类的一个重要依据。在 MCS-51 系列微控制器中，就分为 8031（无 ROM 型）、8051（ROM 型）、8751（EPROM 型）、8951（E2PROM 型）以及 89S51（Flash 型，支持 ISP①）等。总体上，微控制器

① ISP：In-System Programmable，在线可编程，是一种在芯片正常工作时直接对其进行逻辑重写的技术。

具有一些共性特点，可概括为**计算核的指令数较少、位处理指令较丰富、组件集成度较高、提供多种I/O接口、低功耗、低成本、使用方便以及快速构建应用系统等**。同时，微控制器也存在供电电压较单一、计算性能通常不高、引脚数量较少等缺点。

在目前的嵌入式应用中，微控制器也是使用最为广泛的一类处理器，应用领域覆盖消费电子、信息家电、工业控制和汽车电子等。不同领域对微控制器的特性要求有所不同。例如，对于智能玩具等功能较为单一的消费电子产品而言，需要低功耗、低价格的微控制器；信息家电领域需要可靠性高的微控制器；而在工业控制、汽车电子等领域，主要要求微控制器具有良好的可靠性、安全性和高性能。在实际硬件选型过程中，设计者一般要坚持“成本优化、功能/性能够用、简洁易用”的原则，尽量选择芯片性能及其片内资源接近功能需求的微控制器，以提高设计效率和产品的性价比。

3.2.2 嵌入式微处理器

不同于面向快速构建控制系统需要，提供集成度高的计算核心的微控制器设计理念，由通用计算机的微处理器演化而来的嵌入式微处理器（MPU）主要面向结构更为复杂、功能更为丰富或性能要求更高的嵌入式应用，具有良好的通用性及可扩展性。几十年来，嵌入式微处理器的发展也呈现出百花齐放的局面，有数十上百种体系结构和成百上千种类型，典型的如x86（386EX）、ARM、MIPS、PowerPC、SPARK、SuperH（SH）以及基于RISC-V的相关系列等。

嵌入式微处理器以中央处理单元为核心，其通常采用RISC指令集、多级流水体系、32位/64位字宽、数据Cache和指令Cache，主频可达数百乃至上千兆赫兹。在多数嵌入式微处理器内部，通常还根据领域需要扩展、集成了增强计算能力和可扩展能力的组件，如浮点协处理器单元、MMU、DMA、高性能总线控制器、在线调试逻辑等。总体上，嵌入式微处理器的设计思想、逻辑结构以及性能指标等方面都与通用微处理器比较接近。二者的主要区别在于，通用微处理器以计算速度和稳定性为主要性能评价指标，而嵌入式微处理器的设计还必须针对一定的领域特征和应用特性，如移动终端、通信服务、流媒体处理等。不同的嵌入式微处理器常常拥有不同的体系、资源和性能，而且，即使在同一体系中也可能具有不同的时钟频率和数据总线宽度，集成不同的外设与接口。另外，嵌入式微处理器的处理架构和计算资源的数量与类型首先是面向领域裁剪（或定制）的，可以以更小的体积、功耗、重量乃至成本满足某类应用或某个领域的系统需求。其次，嵌入式微处理器具有可靠性、功耗、工作温度、湿度、抗电磁干扰、体积、封装形式等多维度的设计指标，较通用微处理器的要求更高。

除了不断提升芯片工作频率、优化处理逻辑、改造制程工艺之外，现代微处理器的体系结构也在不断演化。在单一芯片内集成多个计算核形成的多核并行计算体系已成为微处理器的主流设计架构。在多核嵌入式处理器中，多个同构或异构的计算核独立运行，各计算核可以具有独立的高速缓存，计算核之间通过共享的二级缓存、内存及总线进行数据交互和操作同步。这种多核的处理器体系结构有效抵消了加工工艺、工作频率以及芯片发热等方面的瓶颈和限制，较多处理器体系结构具有更高的集成度、更小的体积和功耗，也使得摩尔定律得以有效延续。总体上，嵌入式微处理器的特点可归纳为**指令丰富、位处理指令较少、处理速度快、可扩展性好、支持内存保护机制、寻址能力强、支持实时多任务、具有良好的中断处理能力、可靠性高、功耗较低、通用性强、适合中高端应用**等。

以32位MIPS M 6250[29]低功耗处理器为例。M6250是一款中高端的嵌入式微处理器，其逻辑结构如图3.14所示。M6250内部采用了6级流水结构，提供一个协处理器、支持32位MIPS32指令集和精简的microMIPS指令集、整合SIMD和DSP模块、支持64KB的一级指令/数据Cache，具有可选的SPRAM(ScratchPad RAM，高速暂存存储器)接口、可选的数据纠错组件ECC(Error Correct Code)，具有快速重编址缓冲器TLB(快表)的MMU和APB(Advanced Peripheral Bus，高级外围总线)、AXI3高速可扩展总线接口等。M6250的架构和资源配置表明嵌入式微处理器的逻辑和组件设计主要是围绕提升处理器计算性能并增强计算系统架构的可扩展性进行的，而且并未像微控制器一样集成其他面向I/O操作的硬件组件。

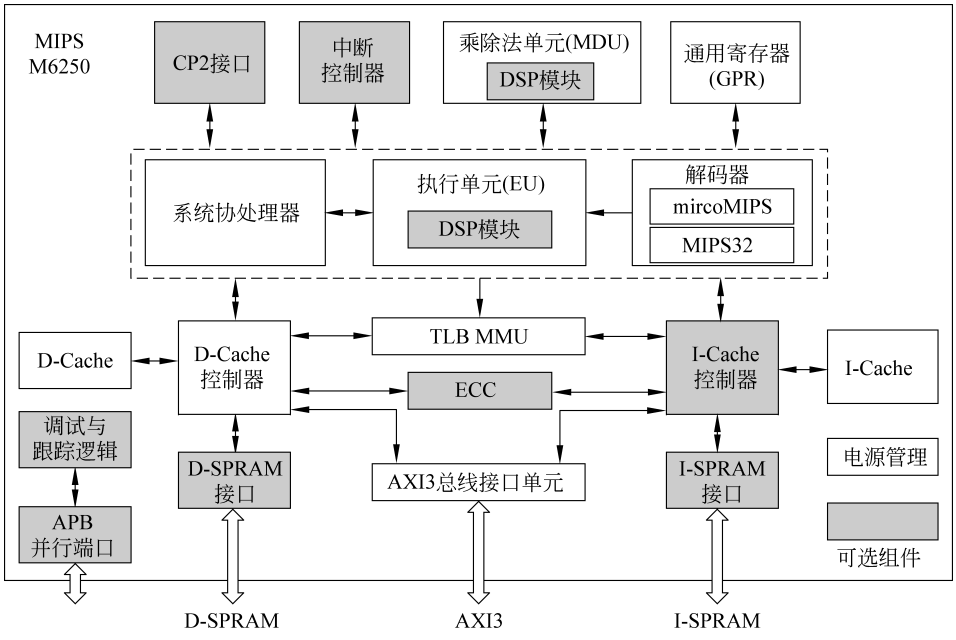


图3.14　32位MIPS M6250嵌入式微处理器的逻辑结构

又如，SiFive公司基于RISC-V指令集设计的U74-MC复杂核包括一个S7(基于RV64IMACB指令集)和4个U74(基于RV64GCB指令集，且提供一个符合IEEE 754—2008标准的浮点处理单元)的64位RISC-V核，以及支撑这些计算核所需的核级本地中断管理器(Core Local Interruptor，CLINT)、平台级中断控制器(Platform-Level Interrupt Controller，PLIC)、物理内存保护(Physical Memory Protection，PMP)、基于JTAG的调试单元、64位/128位AXI4总线等。具体请参阅文献[30]，这里不再赘述。

在实际设计中，嵌入式处理器的选型仍要以应用需求为主要依据，需要综合考虑功能、性能、功耗、尺寸、封装、成本及生命周期等诸多因素，在够用原则下择优选用。由于嵌入式微处理器的I/O集成度低，因此，硬件设计的重点是以嵌入式微处理器为核心进行外部I/O的扩展。嵌入式微处理器有两种主要应用形式：一种是以微处理器为中心，设计面向具体产品型号的嵌入式系统，如智能手机、机顶盒等；另一种则是面向工业控制、信号处理等某个行业或某一类应用，设计为集成了特定接口、软件可编程并具有一定领域通用性的单板计算机模块。如前所述，这种形式较传统的工业控制计算机而言具有体积小、重量轻、成本低、可

靠性高等优点。

3.2.3 数字信号处理器

信号是一个特定形式的能量单元，是信息的物理表现，或者说载体。在物理世界中，任何随时间或空间变化的量都是潜在的信号，其或者表示一个物理系统的状态信息，或者是不同观察对象之间传达的消息，其形式可为声、光、电等。在电子与通信技术领域，信号被抽象表示为传递有关现象的行为或属性信息的函数，在电子设备间进行传输收发。按照因变量对时间的取值是否连续，信号一般被分为模拟信号和数字信号。数字信号通常是对模拟信号进行采样、量化和编码后获得的。信号处理是很多电子装置的重要功能，其通过对变换到数字域的信号进行压缩、编解码、分析或识别等操作，从输入信号中滤除噪声或将信号变换为易于处理、传输、分析和识别的形式。数字信号处理是信号处理的重要分支。

由模拟信号处理器（Analog Signal Processor，ASP）和离散逻辑发展而来的数字信号处理器（Digital Signal Processor，DSP）是一种专门用于数字信号处理的微处理器，最早以20世纪70年代末德州仪器（TI）公司的Levinson滤波器芯片和贝尔实验室的Mac 4型DSP为标志。DSP的主要特点是针对数字信号处理特性设计、优化了专门的硬件逻辑。图3.15给出了一个DSP的基本逻辑结构。图中，DSP的计算核不仅包括传统微处理器中的ALU，还增加了乘法器（Multiplier）、移位器（Shifter）及丰富的数据寄存器、确保多级指令流水正确性的程序定序器、两套分开的数据总线和地址总线（可访问独立的数据存储器和程序存储器）以及专门的数据地址产生器（Data Address Generator，DAG）等。这种设计方式允许DSP在单个时钟周期内就可完成一个在MCU或MPU上需要多条指令和多个周期才可以完成的操作。这些特殊的硬件设计都充分体现了对数字信号处理的密集数据、SIMD、矩阵操作等数学计算特征的考虑。当然，作为一款嵌入式处理器，DSP芯片内也可集成定时器/看门狗、DMA等外围电路组件和接口，以提高芯片的集成度、可扩展性和可靠性。

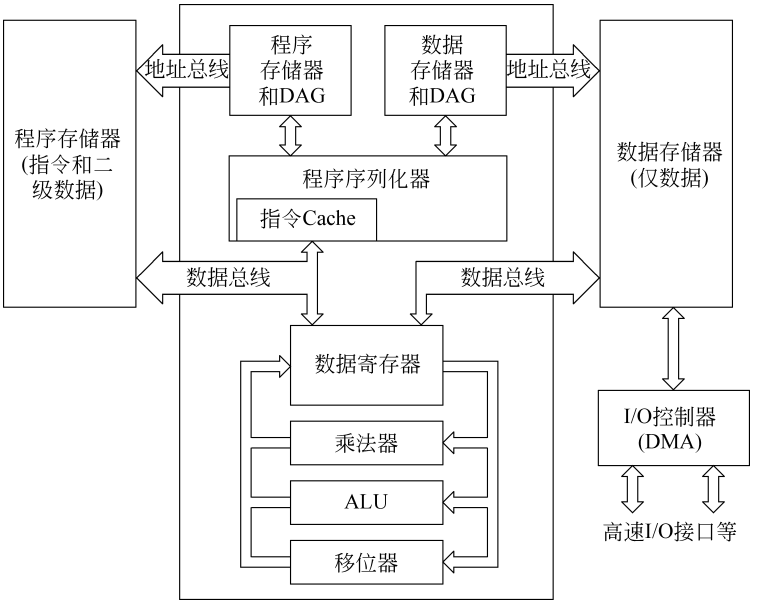


图3.15 DSP的基本逻辑结构

图 3.16 是 ADI 公司 ADSP-21532 DSP 的片内逻辑结构[31]。该芯片中集成了一个 300MHz 的高性能 MSA(Micro Signal Architecture,微信号体系)DSP 计算核,片上存储资源包括 16KB 指令 SRAM/Cache、32KB 指令 SRAM、32KB 指令 ROM、32KB 数据 SRAM 以及 4KB 的 ScratchPad SRAM,一个 DMA 控制器和一个用于内存保护的 MMU。同时,该 DSP 还集成了 3 个定时器/计数器、实时时钟(RTC)、看门狗定时器(WDT)、并行外设接口(PPI)、支持红外(IrDA 标准)的异步收发接口(UART)、通用 I/O 接口(GPIO)以及片上 JTAG 测试与仿真接口等资源。在如图 3.16 所示的 MSA DSP 计算核逻辑中,有两个 16 位的乘加器(Multiplier/Accumulator,MAC)、两个 40 位的 ALU、4 个 8 位的视频 ALU 以及一个 40 位的桶型移位器,可以处理 8 位、16 位、32 位格式的数据。地址计算单元提供了两套数据地址生成器(DAG),以实现并行的内存操作。一个多端口寄存器文件包括 4 组 32 位的索引(I)、修改(M)、长度(L)及基数(B)寄存器和 8 个附加的 32 位指针寄存器。控制单元提供了定序器、对齐、解码及循环缓冲器。从这些资源配置和逻辑设计就可以看出 DSP 硬件在数字信号处理方面的特点与优势。

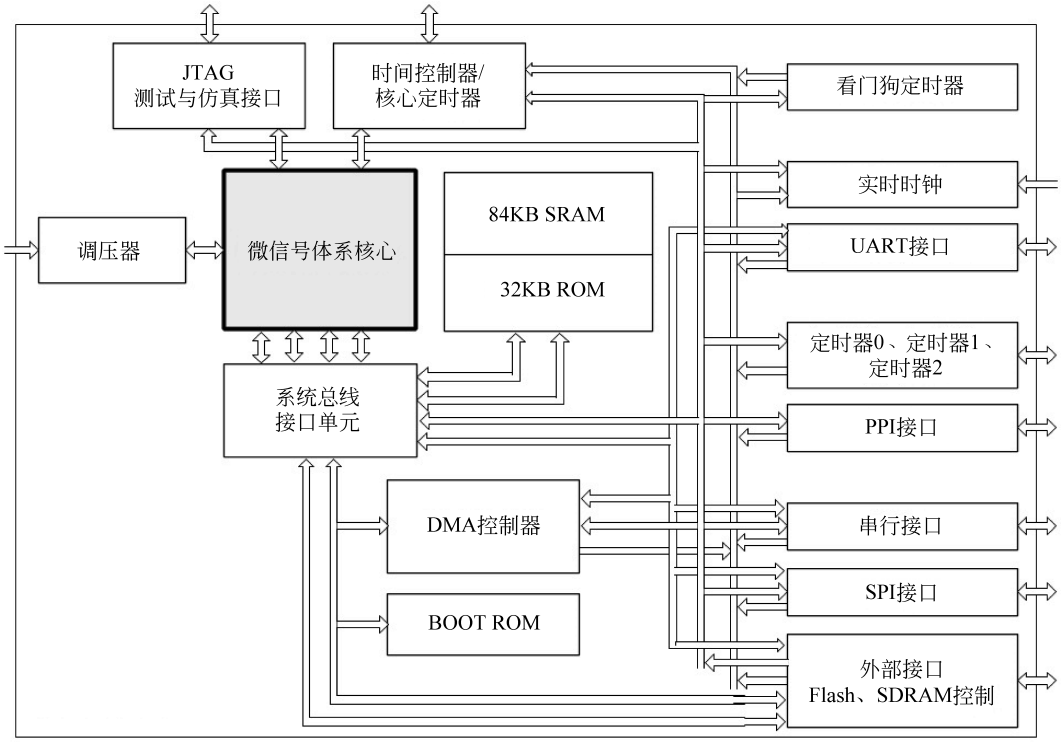


图 3.16 ADSP-21532 DSP 的片内逻辑结构

总体来说,DSP 芯片通常都采用诸多优化的软硬件设计方法和技术,以提高处理能力,具体主要包括如下几方面。

(1) 计算核基于多级指令流水体系结构设计;采用多硬件乘法器单元,可在一个时钟周期内并行完成多个乘加运算;提供硬件控制循环,降低循环操作的开销。

(2) 指令集中采用 SIMD、VLIW 以提升指令级的并行性;提供 MAC 操作指令以及与模寻址相关的指令等;面向数据处理提供了饱和算子、定点算子及单周期操作指令。

（3）存储体系主要采用哈佛体系结构并支持多数据存储器，可以实现并行的多指令和多数据读取；可提供DMA接口，以实现快速存储器访问。

（4）在寻址机制上，计算核采用专门的数据地址产生器；支持常见的寻址模式，也提供了特定的优化寻址模式，如自动增量寻址、循环寻址以及用于快速傅里叶变换（Fast Fourier Transform，FFT）的位翻转寻址，具有非常好的直接数据访问能力。

就分类而言，根据DSP所支持的数据格式是定点型还是浮点型，可以将其分为定点DSP处理器和浮点DSP处理器，字长可以有16位、24位和32位。DSP所能支持的数据格式和字长决定了其处理的精度和动态范围，同时也决定了DSP的成本、功耗以及软件开发复杂度。常见的DSP主要为定点DSP，如TI公司的TMS320C1x/C2x、AD公司的ADSP21xx系列等，这是因为信号处理一般不需要浮点型的精度。对于定点DSP，编程时要仔细考虑信号幅值和中间结果，在避免溢出和尽可能减小舍入误差的前提下，使精度和动态范围最大化。表3.1给出了16位定点数在不同小数点位置（Q_n）时的数值范围和精度。浮点DSP，如TI TMS320C3x、Motorola MC96002，较定点DSP具有更高的处理精度和复杂度，适合科学运算及其他高精度应用。不同的DSP可采用的浮点数格式有IEEE 754标准定义的（扩展）单精度、（扩展）双精度格式以及TI公司定义的TMS320C3x浮点数格式等。大多数32位浮点DSP具有40位的运算单元，并可以进行扩展单精度格式的浮点运算。根据用途，可以将DSP分为**通用**DSP和面向数字滤波、卷积及FFT等特定运算的**专用**DSP。例如，将专门用于图像处理的DSP称为**图像信号处理器**（Image Signal Processor，ISP）。根据核数量，还可以将DSP分为**单核**DSP和**多核**DSP。总体上，DSP的特点可归纳为计算核体系优化、采用哈佛体系结构、突出的数学计算性能、高集成度等。

表3.1　16位定点数在不同小数点位置时的数值范围和精度

Q_n	数值范围	精　度
0	−32 768～32 767	1
1	−16 384～16 383.5	0.5
2	−8192～8191.75	0.25
3	−4096～4095.875	0.125
4	−2048～2047.9375	0.0625
5	−1024～1023.968 75	0.031 25
6	−512～511.984 375	0.015 625
7	−256～255.992 187 5	0.007 812 5
8	−128～127.996 093 75	0.003 906 25
9	−64～63.998 046 875	0.001 953 125
10	−32～31.999 023 437 5	0.000 976 562 5
11	−16～15.999 511 718 75	0.000 488 281 25
12	−8～7.999 755 859 375	0.000 244 140 625
13	−4～3.999 877 929 687 5	0.000 122 070 312 5
14	−2～1.999 938 964 843 75	0.000 061 035 156 25
15	−1～0.999 969 482 421 875	0.000 030 517 578 125

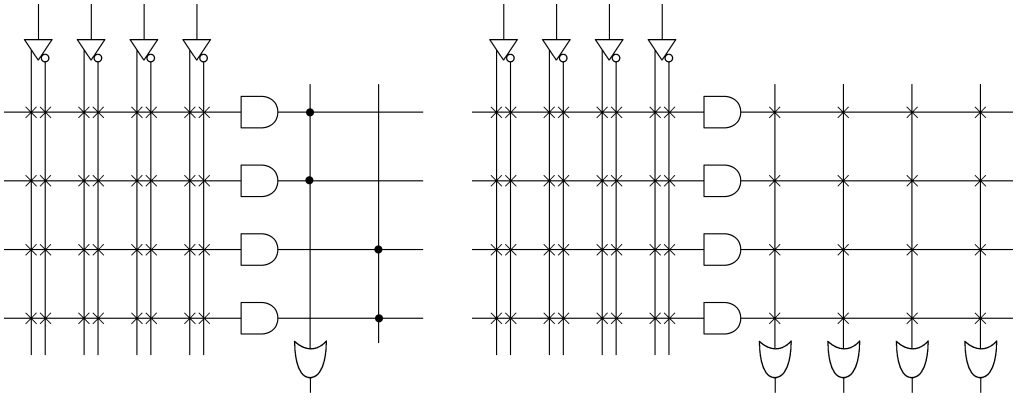
3.2.4 可编程逻辑器件

顾名思义，可编程逻辑器件(Programmable Logic Device，PLD)是电路逻辑可被动态编程和改变的一类通用集成电路。典型的 PLD 包括可编程阵列逻辑(Programmable Array Logic，PAL)、可编程逻辑阵列(Programmable Logic Array，PLA)、通用阵列逻辑(Generic Array Logic，GAL)、复杂可编程逻辑器件(Complex Programming Logic Device，CPLD)以及现场可编程门阵列(Field Programmable Gate Array，FPGA)等。相较于 CPU、GPU 及各类 ASIC 专用器件，其在片内提供了面向应用需求进行逻辑电路编程的定制能力，是集成电路技术的一次重要飞跃，使得**硬件逻辑的快速研制、动态重构、智能演化**等成为可能。

1. PAL 与 PLA

PAL 是一个逻辑门的可编程阵列，门阵列采用 AND-OR(与-或)逻辑进行配置。在 PAL 中，通常具有一个可编程的 AND 阵列和一个固定的 OR 阵列，OR 阵列中的每一个或门从一组与门中获取输入。这意味着，所有与门的输出并不会对应到任何一个或门。与 PAL 不同，PLA 中既包括一个可编程的 AND 阵列，还提供一个可编程的 OR 阵列。其中，AND 阵列用于实现 SOP(Sum of Product，积之和)表达式的乘积项，OR 阵列则用于对乘积项求和。显然，PLA 比 PAL 更为复杂，其逻辑编程能力也更强大。PAL 和 PLA 技术为复杂可编程器件的发展奠定了重要基础。

图 3.17 给出了 4 路输入的 PAL 和 PLA 结构。其中，PAL 有 4 个与门和两个或门。以 BCD 码转格雷码逻辑设计为例，图 3.18 给出了一个 PAL 4 位译码器的逻辑结构。编程所得的译码器以式(3.1)所示的逻辑运算规则将 BCD 码(4 位，分别用 A、B、C、D 表示)转换到格雷码(4 位，分别用 W、X、Y、Z 表示)。



(a) PAL　　(b) PLA

图 3.17　4 路输入的 PAL 和 PLA 结构

$$\begin{cases} W = A + BD + BC \\ X = B\overline{C} \\ Y = B + C \\ Z = \overline{A}\overline{B}\overline{C}D + BCD + A\overline{D} + \overline{B}C\overline{D} \end{cases} \tag{3.1}$$

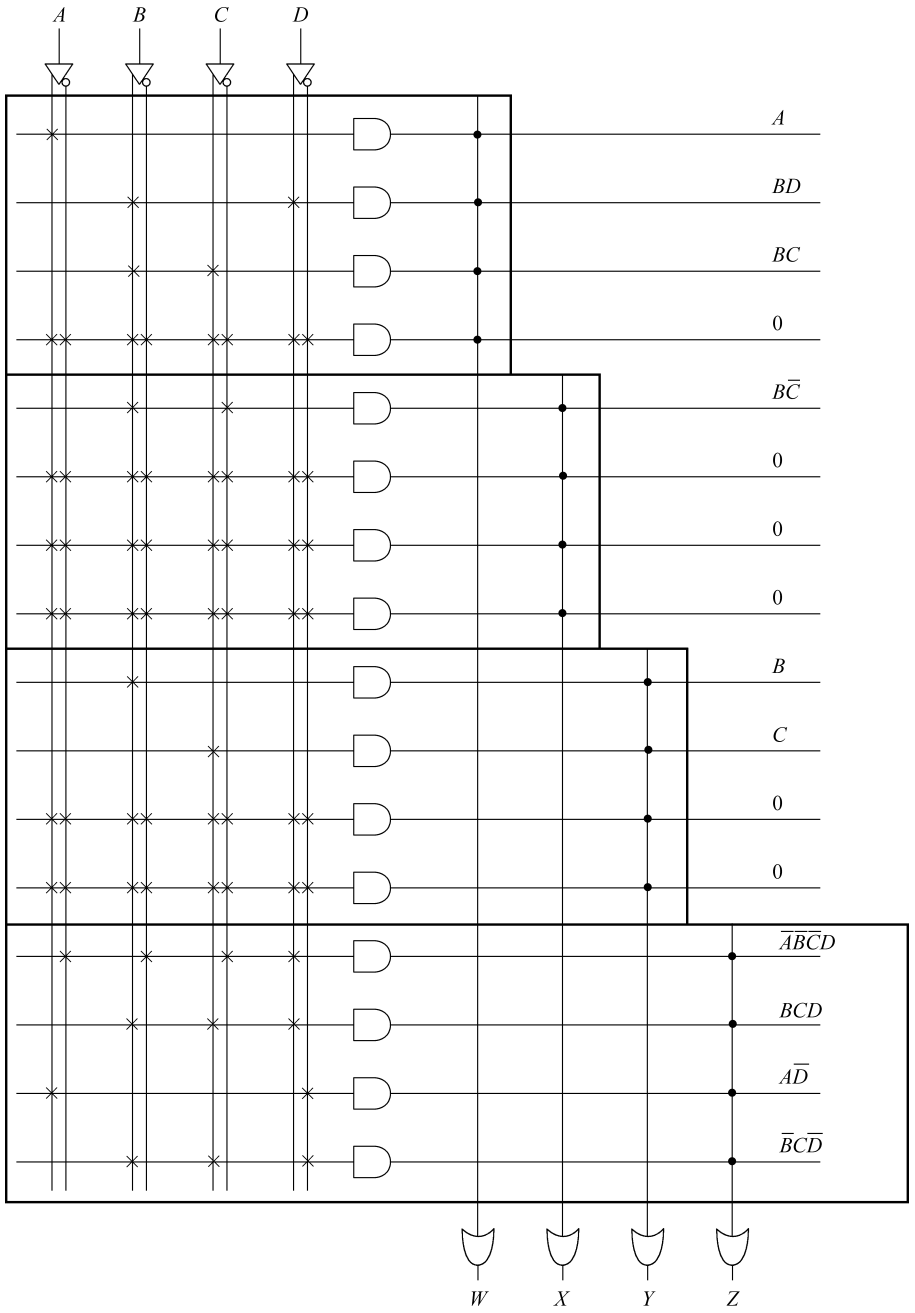


图 3.18　PAL 4 位译码器的逻辑结构

2. CPLD

CPLD 是采用 CMOS EPROM、EEPROM、Flash 和 SRAM 等编程技术设计的可编程逻辑器件，它是以 PLD 为宏单元的可编程器件，门电路数量更多，也更为复杂。在 CPLD 内部，多个类似于 PAL 和 PLA 的逻辑块宏单元通过可编程互连通道进行连接，并通过可配置的 I/O 控制块提供对外交互接口。一般而言，CPLD 器件的核心包括逻辑阵列块、可编程互连通道和 I/O 控制块 3 个主要部分。具有 4 个 PAL 块的 CPLD 的典型逻辑结构如图 3.19 所示。

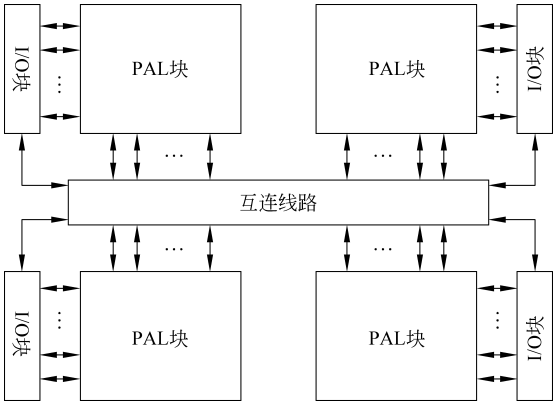


图 3.19　具有 4 个 PAL 块的 CPLD 的典型逻辑结构

以图 3.20 所示 Altera MAX 系列的 CPLD 为例，其逻辑包括了多个逻辑阵列块(Logic Array Block，LAB)、片内互连通道(可编程互连阵列，Programmable Interconnected Array，

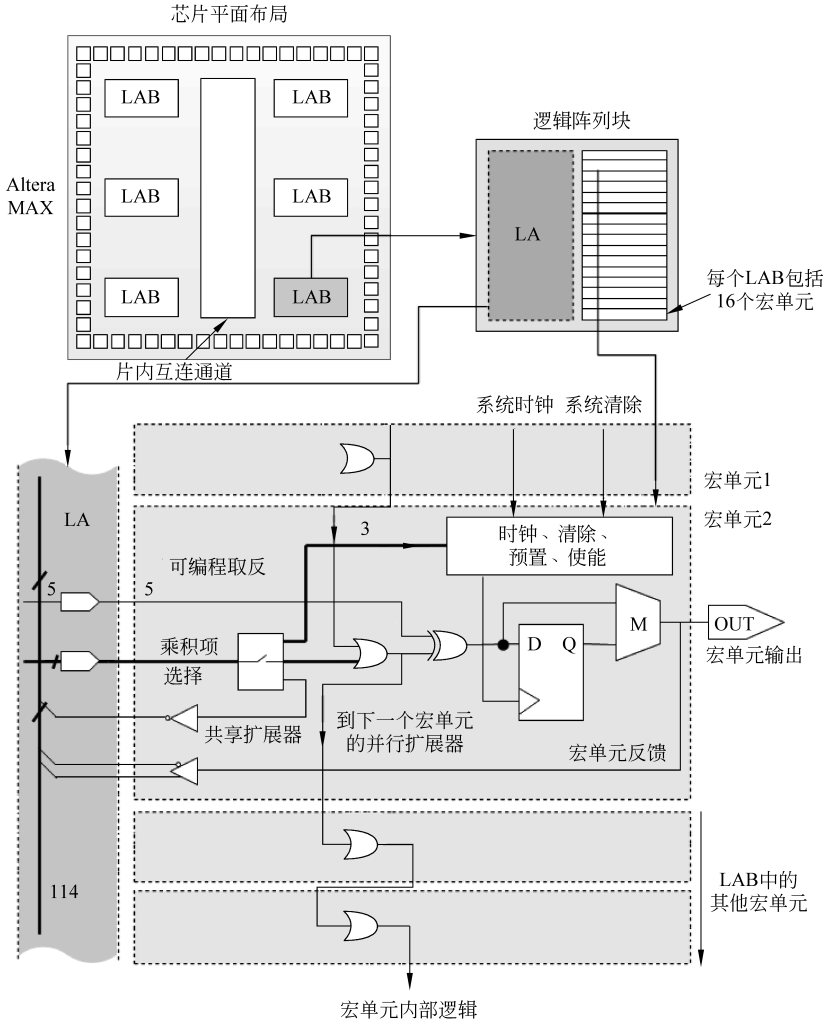


图 3.20　Altera MAX 系列 CPLD 的体系结构

PIA）以及I/O块。其中，逻辑阵列块由局部阵列（LA）和宏单元（marcocell）组成。LA提供了大量可编程AND阵列，而类似于PAL的宏单元则由固定的OR阵列、用于生成额外逻辑项的逻辑扩展器（logic expander）以及用于减少所需乘积项数量的可编程取反逻辑（Programmable Inversion）等组成。

表3.2给出了Altera MAX Ⅱ系列CPLD的特征参数[32]。CPLD的特点在于**逻辑非易失、启动速度快、较高密度和低功耗**，常用于通用CPU、DSP等难以完成的时序组合逻辑、大运算量的数学计算等。

表3.2　Altera MAX Ⅱ系列CPLD的特征参数

特征参数		EPM240/Z	EPM570/Z	EPM1270	EPM2210
密度和速度	宏单元	192	440	980	1700
	引脚间延迟/ns	4.7,7.5	5.4,9.0	6.2	7.0
体系特征	用户Flash/kb	8			
	边界扫描JTAG	有			
	快速输入寄存器	有			
	上电可编程寄存器	有			
	JTAG ISP	有			
	实时ISP	有			
I/O特征	I/O多电压/V	1.5,1.8,2.5,3.3	1.5,1.8,2.5,3.3	1.5,1.8,2.5,3.3,5.0	1.5,1.8,2.5,3.3,5.0
	I/O电源线数	2	2	4	4
	最大输出数	80	160	212	272
	32位，66MHz PCI	无	无	有	有
	其他特性	LVTTL/LVCMOS，可编程转换速率，施密特触发器，可编程上拉电阻，可编程GND引脚，开漏输出，总线保持			

3. FPGA

FPGA也是被广泛使用的典型可编程半导体器件，其内部用户可编程的单元包括输入输出模块（Input/Output Block，IOB）、可配置逻辑块（Configurable Logic Block，CLB）和内部布线资源（Interconnect），同时可提供专用的块内存（Block RAM，BRAM）以及数字时钟管理（Digital Clock Management，DCM）模块、时钟延迟锁相环（Delay-Locked Loop，DLL）和DSP、CPU核等系统级的功能组件。图3.21所示为Xilinx FPGA的逻辑结构。FPGA以集成电路形式存在，同时以硬件描述语言（Verilog或VHDL）编程的方式进行逻辑电路设计和布局，这使得FPGA具有显著的软硬件一体化的特性，主要包括：允许通过下载配置位流改变功能，定制外设接口；支持有限状态机、数字逻辑、定时和存储器控制；可实现与其他处理器的高速通信接口；支持高速且硬件级并行的信号处理和控制算法；等等。FPGA的出现为嵌入式处理器、嵌入式硬件以及嵌入式系统的设计开创了新的模式。例如，基于FPGA的复杂电路逻辑设计已经成为广泛采用的集成电路设计、验证手段。

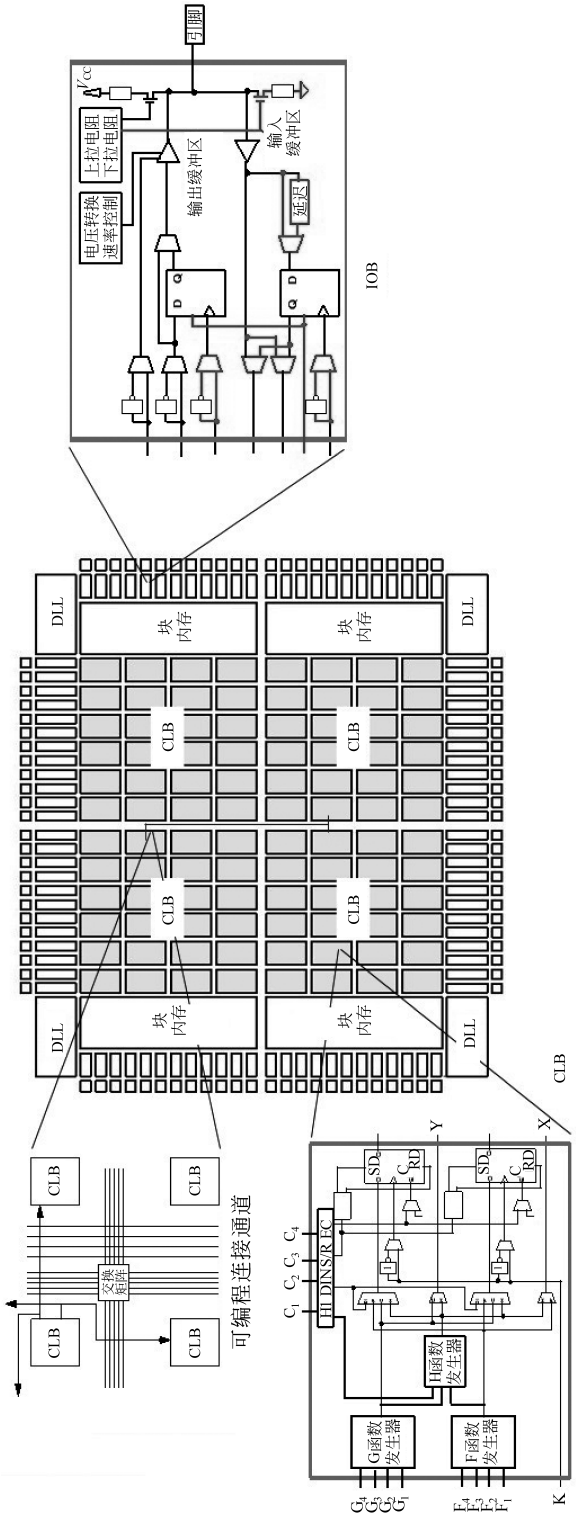


图 3.21 Xilinx FPGA 的逻辑结构