JISUANJI ZUCHENG YUANLI
SHIYAN JIAOCAI

计算机组成原理实验教材

王春桃 / 主编

电子科技大学出版社
University of Electronic Science and Technology of China Press

计算机组成原理
实验教材

JISUANJI ZUCHENG YUANLI
SHIYAN JIAOCAI

王春桃 / 主编

电子科技大学出版社
University of Electronic Science and Technology of China Press

·成都·

图书在版编目(CIP)数据

计算机组成原理实验教材 / 王春桃主编. -- 成都：电子科技大学出版社，2019.8
ISBN 978-7-5647-7175-1

Ⅰ. ①计… Ⅱ. ①王… Ⅲ. ①计算机组成原理－实验－教材 Ⅳ. ①TP301-33

中国版本图书馆CIP数据核字（2019）第131451号

计算机组成原理实验教材

王春桃　主编

策划编辑　兰　凯
责任编辑　兰　凯
特约编辑　潘宗峰

出版发行　电子科技大学出版社
　　成都市一环路东一段159号电子信息产业大厦九楼　邮编 610051
主　　页　www.uestcp.com.cn
服务电话　028-83203399
邮购电话　028-83201495

印　　刷　四川煤田地质制图印刷厂
成品尺寸　185mm × 260mm
印　　张　10
字　　数　250千字
版　　次　2019年8月第一版
印　　次　2019年8月第一次印刷
书　　号　ISBN 978-7-5647-7175-1
定　　价　32.00元

前　言

“计算机组成原理”课程是计算机科学与技术学科的一门核心专业基础课程，它侧重于介绍冯·诺依曼体系结构型计算机各主要模块的组成结构及基本原理。该课程的先导课程是数字逻辑，它是计算机操作系统、数据结构、微机接口技术等课程的基础。

计算机组成原理课程的系统性强、内容广、细节多、偏硬件，使得学生普遍感觉比较难学，且通常觉得学习该课程难有实质性的收获而“用处”不大。编者经过近十年的教学观察、实践和研究发现，这其实主要是因为学生仅仅是吸收了计算机组成原理的基本知识，并没有理解掌握这些基本知识背后隐藏的系统思维，使得“知其然”而“不知其所以然”。鉴于该课程具有很强的系统性，若能运用“分工、合作、优化”式的系统思维遵循“需求分析→结构设计与分析→模块详细设计”的逻辑来阐述理论课知识点，并以此方式设计相应的实验，则能较好地解决这个问题。其中，“分工”是指把需求进行恰当分割后用独立的模块来实现，“合作”是指这些独立的模块相互传递各种相关信号进行协作而完成功能需求，“优化”是指对模块及模块间的互连结构进行优化调整以获得尽可能最优的系统性能。

对计算机组成原理的理论课而言，可以先梳理分析功能需求；然后用“分工”方式逐一落实各功能，从而最终覆盖所有功能需求；随后分析这些独立模块间的“合作”方式，从而最终实现该系统的具体功能；若有必要，则对所设计的模块及模块间的互连结构进行优化，以满足性能要求。采用这种方式，不仅能掌握课程知识点，也能清楚这些知识点究竟是如何得到的，还能理解、掌握如何针对实际需求开展系统设计与实现的思路与方法。

在此基础上，利用硬件描述语言进行具体的编程实现，进而用硬件实验箱进行硬件仿真测试，能让学生充分理解、熟悉并掌握“分工、合作、优化”式系统思维。这种思维模式本质上是一种系统工程式的思维模式，不是硬件系统特有的，因此当这种思维模式逐渐形成惯性后，就能针对实际应用中的现实需求问题开展设计和实现，从而提升实践问题的解决能力和创新能力。从这个角度看，即便是该课程的具体知识点一点都用不上，其背后隐含的“分工、合作、优化”式的系统思维却可以一直在硬件、软件系统工程或者其他日常生活工作的类似问题中发挥重要的作用。也就是说，如果在计算机组成原理的理论及实验课程中贯彻“分工、合作、优化”式的系统思维，将具有重要的意义和功用。

本书以“分工、合作、优化”式系统思维的运用为主线，首先在第1章简要介绍了如何运用此思维经过一定的逻辑推导得到各章节的主要知识点；然后在第2章简要介绍了VHDL编程语言，为计算机组成原理课程实验奠定基础；第3章通过图文方式介绍了如何使用实验

中要用到的软硬件平台；第4章至第7章基于“分工、合作、优化”式系统思维构造了4个基础实验，每个基础实验都按“功能需求分析→结构设计与分析→模块详细设计→VHDL编程实现→硬件仿真测试”的方式展开，因此每个基础实验就是一个相对完整的系统工程，这些基础实验实质上也构成了综合实验中设计与实现的实验CPU的重要组部分；第8章构造了面向实验CPU设计与实现的综合实验，详细介绍了实验CPU的功能需求分析、指令系统设计、数据通路基本结构设计、控制系统设计、各模块详细设计、VHDL编程实现、硬件仿真测试等硬件系统工程的环节。希望通过这种方式，让学生熟悉、理解和掌握“分工、合作、优化”式的系统思维，夯实及提升解决实际问题的能力。

上述构造的基础实验及综合实验，在近几年的实践中取得了一定的效果，表明了上述思路的可行性。因编者水平所限，书中难免存在一定的错漏，殷切希望广大读者批评指正。

编　者

2019年4月

目 录

第 1 章　计算机系统概述

“计算机组成原理”课程主要介绍冯·诺依曼体系结构计算机的基本原理。该课程首先介绍冯·诺依曼体系结构，如图 1.1 所示；然后再逐步详细介绍冯·诺依曼体系结构中的运算器、内部存储器、指令系统、控制器、系统总线、外部存储器及输入输出设备。

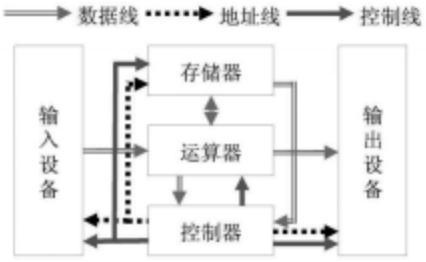


图 1.1　冯·诺依曼体系结构示意图

当前的《计算机组成原理》教材，侧重于介绍冯·诺依曼体系整体结构及其各部分的原理，以便让学生理解和掌握冯·诺依曼体系结构计算机的基本原理。这种呈现方式可以让学生理解和掌握计算机组成原理的基本知识，即可以“知其然”；但却难于让学生“知其所以然”，即理解冯·诺依曼体系结构及其各部分组成是如何设计得到的，以及如何针对实际需求开展相应的设计。事实上，冯·诺依曼体系结构及其各组件结构都可以借助于系统思维针对实际需求开展设计而得到，在此基础上对所设计结构的分析就可以形成计算机整体及各部件的组成原理。因此，对于“计算机组成原理”课程的相关知识点，若能首先梳理分析功能需求，然后运用系统思维开展设计，进而对所设计的系统或子系统结构进行原理分析，则学生在学习计算机组成原理的过程中就既能“知其然”又能“知其所以然”，既能掌握计算机组成原理基础知识，又能逐步熟悉甚至掌握如何运用系统思维解决实际问题，从而切实提升解决实际问题的能力。此种能力，本质上就是一种系统（包括硬件及软件系统）工程能力。因此，这种方式不仅能让学生掌握基础知识，还能让学生掌握针对实际问题开展设计进而切实解决问题的能力。

鉴于当前《计算机组成原理》教材的呈现方式并未按这种方式阐述知识点，因此本章按这种方式简要概述冯·诺依曼体系结构及其各个组成部件，以此让学生在理解掌握计算机组成原理基本知识的基础上，进一步理解并熟悉如何针对功能需求运用系统思维开展设计，从而达到逻辑上解决实际问题的目的。在此基础上，进一步按这种方式开展实验训练，通过硬件仿真的方式切实把冯·诺依曼体系结构计算机的各个主要部件及其整体模型机设计并实现出来，从而切实提升以系统思维解决实际问题的系统工程能力，进而提升创新能力。

本章侧重于以“功能需求分析→结构设计与分析→模块详细设计”的方式概述冯·诺依曼体系结构及其各个主要部件，后续再按这种方式构建各个基础实验及综合实验。本书将这

种方式称为“系统工程式呈现”。

1.1 冯·诺依曼体系结构概述

图1.1示例了冯·诺依曼体系结构。由图可知，冯·诺依曼体系结构主要包含运算器、控制器、存储器及输入输出设备等主要部件，通过这些主要部件的协作即可实现计算机的基本功能。本节的侧重点不在于介绍这些部件的基本原理，因为这些已经在《计算机组成原理》教材中进行了详细介绍；本节的侧重点在于采用“系统工程式呈现”方式展示冯·诺依曼体系结构是如何根据功能需求运用系统思维设计而得到的。

为了理解冯·诺依曼体系结构设计的出发点——功能需求，设想这样一种情景：设计一种硬件，以完成$z=ax+b-c$的计算。为了设计这种硬件，首先要搞清楚人是如何计算这个式子的。从符号计算角度来看，首先要获得a和x，然后计算它们的乘积，接着将乘积结果与b相加，随后再将相加结果减去c，最后将相减结果赋值给z。

符号运算过程解释了人是如何计算这个式子的，但符号运算得不到实际的数值结果。为了获得数值结果，就需要给出a、x、b和c的具体数值。考虑到设计一种硬件来完成式子的计算，从人的角度来看就相当于请其他的力量来计算这个式子，为此需要将a、x、b和c的具体数值以及待计算式子$z=ax+b-c$告诉对方，对方再按照“四则运算”规则来进行计算。基于这个情形，假设甲给出如图1.2所示的带横线的计算用纸，纸的开头部分写下了计算式子$z=ax+b-c$，并在上部4行写下了a、x、b和c的具体数值，然后请乙进行计算。乙接到计算用纸后，根据计算用纸提供的计算式子及相关变量数值，按照“四则”运算规则在纸上逐步写下计算过程，并将结果填到甲指定的位置，过程如图1.2所示。

计算式子：$z=ax+b-c$	
(1)	$a=5$
(2)	$x=2$
(3)	$b=-1$
(4)	$c=6$
(5)	$z=?$
(6)	
(7)	从第（1）行得到a的数值为5
(8)	从第（2）行得到x的数值为2，并与a的数值相乘，得到10
(9)	从第（3）行得到b的数值为-1，与前述结果10相加得到9
(10)	从第（4）行得到c的数值为6，与前述结果相减得到3
(11)	将计算结果3写到第（5）行
(12)	结束计算

图1.2 甲提供数据及计算式子、乙完成计算的过程示意图

把上述的计算过程抽象一下，甲相当于是人，乙相当于是待设计的硬件。鉴于硬件尚未有足够的智能根据计算式子直接解析成计算步骤，人还需要告诉待设计的硬件如何逐步进行

操作（即图1.2中第（7）~（12）行的具体操作）。人完成这两方面的任务后，就交由待设计的硬件来完成。待设计的硬件首先从存放人所给数据的地方提取数据，然后按人给出的操作步骤逐步进行计算，随后得到计算结果，最后写到人所指定的位置中。

为了将人解放出来，人和待设计的硬件不可能一直处于交互状态。也就是说，人只要把数据及计算步骤以某种方式交给待设计硬件后，就由待设计的硬件全程自动处理，无须人再参与其中。为了达到此目的，待设计的硬件显然必须要有一个模块来存放数据和操作步骤，否则人就无法得到解放。为简便起见，这个模块称为模块1。

模块1可以存放数据和操作步骤，因此待设计硬件可以从图1.2中的第（7）行开始逐步执行。第（7）行是从第（1）行中提取a的数值，即从模块1的第（1）行获得a的值。显然，模块1可以提供a的数值，但无法完成取数的功能。也就是说光有模块1还不够，还必须有额外的模块2来从模块1中获取a的数值。因此，必须再增加一个模块2，该模块告诉模块1，它需要获取哪一行的数据，即告诉模块1具体的行编号，并要求模块1把这行的数据取出来。这样利用模块1和2，就可以完成第1个操作步骤。

接下来进行位于第（8）行的第2个操作步骤。首先从模块1取到x的数值，方式与前述取a的方式相同，因此当前的模块1和模块2足以完成当前功能。随后，将a和x相乘得到乘积。此时可以发现，模块1和模块2均无法实现此乘积的功能，因此需要增加额外的模块3来完成此乘积计算功能。这样一来，模块1用于存放数据和操作步骤；模块2把行号和取操作告诉模块1，模块1取出a和x的数值后可以先后直接送给模块3；模块3在获得2个数据后，完成a和x的乘积，并可以把乘积放在模块3中。至此，利用模块1~3就可以实现第2个操作步骤。

随后，继续进行位于第（9）行的第3个操作步骤。与第2个操作步骤类似，首先由模块2把行号和读取操作告诉模块1，模块1取出b的数值。接下来是要将第2个操作步骤的结果与b相加，但截止到当前为止只有模块3能实现乘法功能，没有模块能实现加法功能。为了实现加法功能，要么新增加一个加法模块，要么让模块3增加加法功能。为了避免结构过于复杂，可以考虑将加法功能整合到模块3中，这样模块3就同时具备乘法和加法功能。因此，从模块1中取出的数据b可以送到模块3中；但因为模块3的运算功能不唯一，所以需要由模块2告知模块3进行加法运算。这样，模块1~3就可以实现第3个操作步骤了，其结构如图1.3所示。

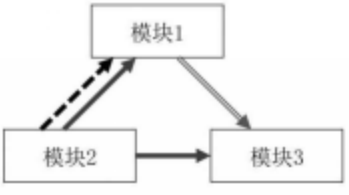


图1.3　模块1~3结构示意图

图1.3中，模块1主要用于存储数据和操作步骤；模块2把行号和读取操作告诉模块1以便模块1能够把所需数据读取出来，亦需要告诉模块3做什么运算以便模块3能够进行正确的运算；模块3主要实现乘法和加法功能。

接着进行位于第（10）行的第4个操作步骤。类似地，模块2把行号和读取操作告知模块1，由模块1把位于第（4）行的数据c取出来。与第3步操作类似，可以在模块3中增加减法功能，并由模块2告知模块3具体做减法运算。因此，模块1把取出来的c送给模块3，并由模块2告知模块3进行减法运算，得到的结果临时存在模块3中。

最后，进行位于第（11）行的第5个操作步骤。此时，模块2把需写入数据的行号及写操作告诉模块1，并由模块3把前一步的运算结果传给模块2，进而由模块1把运算结果写到模块2指定的行中。这就形成了如图1.4所示的模块结构。也就是说，利用如图1.4所示的结构，即能完成所有的操作步骤，从而实现目标计算式所需完成的计算。换句话说，如图1.4所示的结构，就是待设计硬件所需的结构。图1.4中，模块1与2的功能与图1.3相同，模块3则额外增加了减法功能，且模块3把计算结果保存回模块1中。

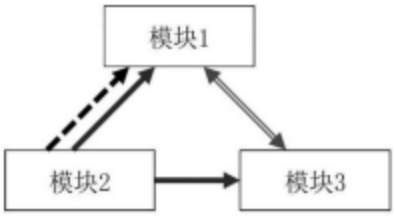


图1.4　模块1～3结构示意图

如果数据不是预先置于模块1，而是来自于待设计硬件与人的交互过程，那么如图1.4所示的所有模块尚不能满足此功能。为此，需要额外再增加1个模块4，用于实现数据从人与硬件的交互过程中得到。若最后的计算结果不仅仅保存到模块1，而是需要展现出来给人观看，那么当前已有的模块1～4都无法完成此功能，需要额外增加一个模块5来实现结果的展示。这时，也可以类似地让模块2把要读取数据的位置信息告诉模块4，实现数据从外界输入到模块3中；并让模块2把要显示的位置信息告诉模块5，从而使得模块5可以把来自于模块2的数据展现给人看。把这些模块组织一下，就能得到图1.1所示的冯·诺依曼体系结构。

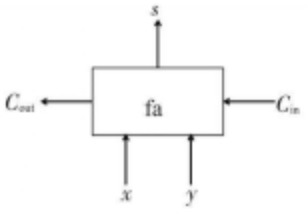
在上述的分析与设计过程中，首先给定设计目标，然后根据从小掌握的“四则运算规则”梳理待设计硬件的功能需求，接着根据运算过程运用逐步增加模块的方法来落实功能需求。在增加模块的过程中，会涉及如何运用这些模块来落实具体操作的分析，这个实际上也就是根据结构来分析功能和原理。这个过程，实质上就是一个运用系统思维进行设计的过程。具体而言，针对设计目标梳理功能需求，为了满足这些功能需求就通过设计一个系统来进行落实。对于这个待设计的系统，所需要做的就是用模块逐一落实各个独立的、不同的功能需求，即若没有模块能满足这个功能，则需要新增功能模块或者合并到某个相关模块中；若已有能实现该功能的模块，则无须新增模块，随后通过模块间的合作就能完成该功能需求。当把所有独立的、不同的功能需求都一一落实后，所有需要的模块都设置了，模块间的合作方式也理清了，因此整个系统也就设计好了，也就达到了设计目标。

为方便记忆和掌握，这个运用系统思维进行设计的过程，可以简称为“分工、合作”。其中“分工”就是指用模块来逐步落实不同的、独立的功能，“合作”就是指利用当前已有的模块进行合作来完成当前所需的功能需求。随着“分工、合作”的不断进行，当把所有功能需求都落实完毕后，应对设计目标的系统也就设计好了。通过这样的方式，就能针对设计目标及功能需求开展设计，也能够针对实际的问题着手予以解决，从而将能够提升解决问题的能力。当然，这样设计的系统虽然能够达到设计目标，但不一定是最优的，实际应用中需要进一步去优化系统结构。为此，可以将运用系统进行设计的过程简称为“分工、合作、优化”，此思维方式成为“分工、合作、优化”式系统思维。

上述分析表明，通过运用“分工”与“合作”，可以顺理成章地设计得到冯·诺依曼体系结构。因此，通过这种方式，既能理解掌握冯·诺依曼体系结构的基本原理，还能理解这种体系结构是如何来的，亦能了解熟悉其背后隐含的、有利于解决实际问题的系统思维。

1.2　运算器概述

对于运算器，《计算机组成原理》教材侧重于介绍定点及浮点数的“+”“-”“*”“÷”这4种基本运算在运算器内部是如何进行的，并介绍了对应的运算器组件。而为了实现运算，首先得在计算机内部表达现实世界的数字及事物，为此教材在阐述这4种运算前先介绍数字及事物在计算机内部的表示方式，包括定点、浮点数表示方式，以及为了方便计算而发展得到的机器码（原码、反码、补码、移码）。



x和y—1位的2个输入信号；C_{in}—输入的初始进位信号；s—全加器相加结果；输出C_{out}—输出的进位信号。

图1.5　1位全加器fa示意图

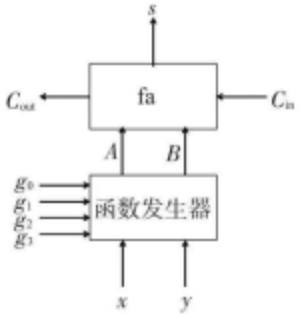
由于教材已经较为详细地介绍了上述的内容，所以本节不再进行重复介绍。本节侧重点在于运用系统思维开展运算器（即ALU，算术逻辑单元）的设计。根据《计算机组成原理》教材，ALU能够进行16种算术逻辑运算，且可以实现先行进位。这是一种相对比较复杂的功能，学生往往可以理解其功能，但通常比较难开展设计而得到此运算器。本节以1位的全加器（fa，如图1.5所示）为出发点，展示如何运用“分工、合作、优化”式系统思维逐步设计得到该ALU，详细的设计过程如下所示。

步骤1：用1位的fa，设计能进行16种算术运算的1位运算器。

对于1位的fa，能实现$s=x+y+C_{in}$的功能，并产生进位输出信号C_{out}。也就是说1位的fa只能完成1位的二进制加法。

对于设计目标——16种算法运算的1位ALU，是指利用输入的x和y，获得16种不同的运算结果。例如，产生$s=x-y-C_{in}$、$s=x$、$s=y$、$s=\overline{x}+y+C_{in}$等结果。

显然，现有的1位fa模块是无法实现除加法外的其他运算功能的。要实现其他功能，那么必须运用系统思维中的“分工”思维，增加额外的模块才有可能实现16种运算的功能。那么这个新增的模块应该要放到什么位置好？根据16种算术运算的功能可知，实际上就是对输入的x和y进行16种不同的组合，然后将组合结果再输入1位fa中进行加法运算，将能得到所需的16种不同算术运算结果。因此，这个新增的模块可以放置于输入信号x和y及1位fa之间，如图1.6所示。



其中，函数发生器在控制信号 $g_3\ g_2\ g_1\ g_0$ 的控制下将输入信号 x 和 y 组合成16种不同的 A 和 B，进而经由1位fa产生16种不同的结果 s 和进位输出 C_{out}。

图1.6　能做16种算术运算的1位ALU示意图

对于新增加的模块，教材中称之为“函数发生器”，即将输入的 x 和 y 映射为16种输出，即16种不同的 A 和 B 的组合。由于任何一个既定函数，只要给定输入就一定会产生确定的输出，所以为了产生16种输出，就必须要引入额外的变量。要组合产生16种情况，就需要4个变量，记作为 $g_3\ g_2\ g_1\ g_0$。从函数的角度来说，这是函数的4个参变量；从实际硬件角度来看，这4个变量就是4个控制信号。

利用这4个变量的排列组合，就可以形成16种不同的 A 和 B 的组合，从而可以得到真值表。获得真值表后，利用“数字逻辑”课程的基础知识（如卡诺图化简法），就可以推导得到函数发生器的逻辑方程，从而真正把函数发生器实现出来。为了进一步简化设计，尚可以进一步进行独立控制，如用 $g_3\ g_2$ 控制 A 的产生，获得4种情况；用 $g_1\ g_0$ 控制 B 的产生，获得4种情况。A 和 B 各自4种情况进行两两组合，就能获得16种情况。结果是一致的，但因为只有4种情况，可以大大简化其逻辑方程的推导。鉴于此过程教材中已有明确的描述，且本节侧重于运用系统思维开展ALU的组成结构设计，所以逻辑方程的推导在此略去。

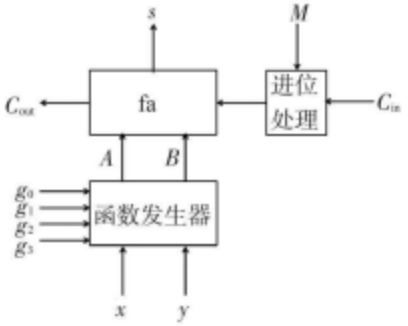
步骤2：在步骤1的基础上，设计1位的、能进行16种算术逻辑运算的ALU。

经过步骤1的设计，获得了具有16种算术运算的1位ALU。基于此，进一步叠加逻辑运算，以获得能进行16种算术逻辑运算的运算器。根据系统思维中的“分工”策略，当前的具有16种算术运算的1位运算器并不能完成16种逻辑运算，因此需要额外增加一个模块来实现16种逻辑运算。那么这个新增模块放到什么地方才较合理地实现设计目标呢？

为了准确放置新增加的模块，首先就需要搞清楚算术运算和逻辑运算的功能需求，或者说搞清楚它们之间的差别是什么。对于二进制的算术运算和逻辑运算，经过简单分析可知，它们的区别主要在于是否对初始进位进行运算，即算术运算需要对初始进位 C_{in} 进行运算（如加上初始进位），但逻辑运算则不需要考虑初始进位 C_{in}。也就是说，做算术运算时必须要把 C_{in} 输入到1位fa中，做逻辑运算时则不需要把 C_{in} 输入到1位fa中。根据这一特点可知，新增的模块应该放置于初始进位 C_{in} 与1位fa之间，如图1.7所示。

为简便起见，称该新增的模块为“进位处理”模块。该模块类似于第（1）步中的函数发生器，将输入的初始进位信号组合产生2种不同的输出 C'_{in}。为此，需要引入一个额外的变量或控制信号 M。不失一般性，假定 M=1时有 $C'_{in}=C_{in}$；而在 M=0时有 C'_{in}=0，即不再将输入的 C_{in} 输入到1位fa中。这样就可以形成真值表，进而获得逻辑方程。具体逻辑方程的推导略

去，结果可知进位处理模块实际上就是一个“与门”。



其中，“进位处理”模块在M=1时让$C'_{in}=C_{in}$，而在M=1时让$C'_{in}=0$。

图1.7　能做16种算术逻辑运算的1位ALU示意图

值得指出的是，通过进位处理模块，统一了算术运算和逻辑运算的功能。结合函数发生器的16种运算，产生16种A和B的组合，输入到1位fa后，就产生了16种的算术运算（当M=1时）和16种逻辑运算（当M=0时）。

步骤3：在步骤2的基础上，设计4位的、具备先行进位的ALU。

经过步骤2的设计，获得了具有16种算术逻辑运算功能的1位ALU。接下来在此基础上进一步设计4位的、具有16种算术逻辑运算功能的ALU。

为简便起见，将图1.7所示的具有16种算术逻辑运算功能的1位ALU用逻辑框图进行表示，如图1.8所示。要形成4位的、具有16种算术逻辑运算功能的ALU，显然只要把图1.8所示的4个1位ALU拼接起来即可得到，如图1.9所示。

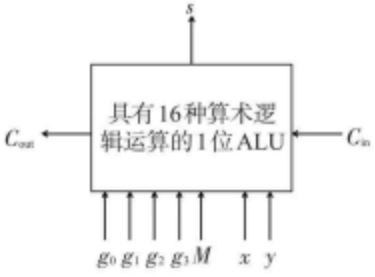
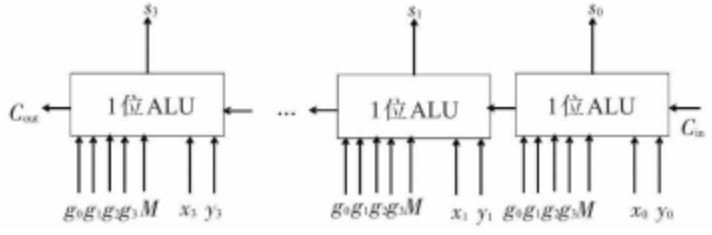


图1.8　具有16中算术逻辑运算的1位ALU逻辑框图



其中，“1位ALU”就是如图1.8所示的具有16种算术逻辑运算功能的1位ALU。

图1.9　具有16种算术逻辑运算功能的4位ALU逻辑推图

虽然图1.9实现了位数为4位、运算功能为16种算术逻辑运算功能的ALU，但该4位ALU之间各位的运算是串行的。因为只有低位的1位ALU完成运算产生进位后，相邻高位的“1位ALU”才能获得初始进位输入而开始进行运算。这就会导致4位ALU的运算速度比较慢。为了加快ALU的运算速度，需要进行“优化”，最好的优化方式就是4个“1位ALU”同时进行运算。考虑到必须获得低位的“1位ALU”模块的进位输出，才有可能进行高位的“1位ALU”的运算，如何才能解决这个问题而实现并行运算呢？

要实现并行运算，那么就必须解决处于高位的“1位ALU”模块的进位输入信号需要等待的问题。为此就需要分析进位输入信号需要等待的原因。假设图1.9中的4个“1位ALU”分别记为模块0、1、2、3，这4个模块的进位输出信号分别 $C_{out}^{i}(i=0,1,2,3)$，则根据1位FA的逻辑方程，4个不同模块的进位输出信号与进位输入信号之间的关系为

$$C_{out}^{0} = B_0 + A_0 MC_{in} \tag{1}$$

$$C_{out}^{1} = B_1 + A_1 MC_{out}^{0} \tag{2}$$

$$C_{out}^{2} = B_2 + A_2 MC_{out}^{1} \tag{3}$$

$$C_{out}^{3} = B_3 + A_3 MC_{out}^{2} \tag{4}$$

由上述公式可知，在计算 $C_{out}^{i}(i=1,2,3)$ 时，必须要等待来自于相邻低位的进位输出信号 $C_{out}^{i}(i=0,1,2)$，这就导致各个ALU模块不能同时运行。也就是说，对于公式（2）~（4）而言，$C_{out}^{i}(i=0,1,2)$ 属于未知变量，所以只有当 $C_{out}^{i}(i=0,1,2)$ 产生后变成已知变量才能进行计算。若能把这些未知变量都变成已知变量，则就不存在等待进位输入信号的问题，也就能实现并行运算。通过观察可知，把公式（1）代入（2）即可消除公式（2）的未知变量 C_{out}^{0}，公式（3）和（4）可类似地处理，即有

$$\begin{aligned} C_{out}^{1} &= B_1 + A_1 B_0 M + A_1 A_0 M^2 C_{in} \\ &= (B_1 + A_1 B_0 M) + (A_1 A_0) MC_{in} \\ &= G_0 + P_0 MC_{in} \end{aligned} \tag{5}$$

$$\begin{aligned} C_{out}^{2} &= B_2 + A_2 B_1 M + A_2 A_1 B_0 M^2 + A_2 A_1 A_0 M^2 C_{in} \\ &= (B_2 + A_2 B_1 M + A_2 A_1 B_0 M) + (A_2 A_1 A_0) MC_{in} \\ &\triangleq G_1 + P_1 MC_{in} \end{aligned} \tag{6}$$

$$\begin{aligned} C_{out}^{3} &= B_3 + A_3 B_2 M + A_3 A_2 B_1 M^2 + A_3 A_2 A_1 B_0 M^3 + A_3 A_2 A_1 A_0 M^3 C_{in} \\ &= (B_3 + A_3 B_2 M + A_3 A_2 B_1 M + A_3 A_2 A_1 B_0 M) + (A_3 A_2 A_1 A_0) MC_{in} \\ &= G_2 + P_2 MC_{in} \end{aligned} \tag{7}$$

其中，$\triangleq$ 代表定义式，因为 $M=0$ 或1，所以有 $M^K = M$，由公式（5）~（7）可知，此时 $C_{out}^{i}(i=1,2,3)$ 公式中的所有变量都是已知变量，所以按公式（1）、（5）~（7）进行计算时，图1.9中的4个ALU就可以并行进行运算了。为简便起见，定义

$$P^* = A_3 A_2 A_1 A_0 \tag{8}$$

$$G^* = B_3 + A_3B_2 + A_3A_2B_1 + A_3A_2A_1B_0 \tag{9}$$

分别为“成组进位传送输出”及“成组进位发生输出”。

根据（1）、（5）~（7）所表达的逻辑方程，即可形成先行进位输出的逻辑结构。根据《计算机组成原理》教材，这种结构称为74182CLA部件，逻辑框图如图1.10所示。利用先行进位产生部件74182CLA，即可形成具有先行进位的4位ALU，如图1.11所示。

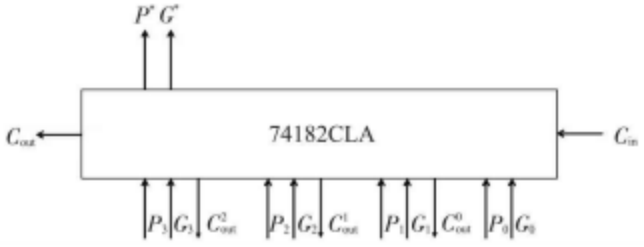
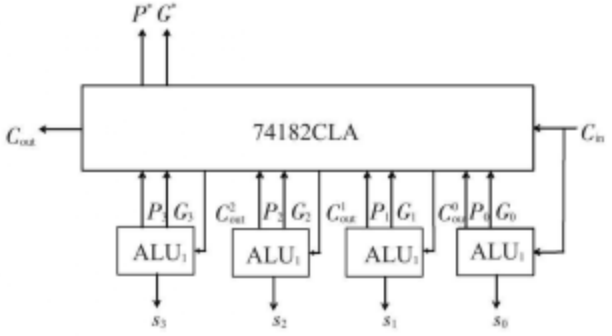


图1.10　74182CLA逻辑框图



其中，ALU_1是如图1.8所示的ALU的改进版，即在图1.8基础上增加了公式（5）~（9）中定义的号P_i和$G_i(i=0,1,2,3)$。

图1.11　具有先行进位的4位ALU，可进行16种算术逻辑运算

步骤4：在步骤3的基础上，设计8位的、具备先行进位的ALU。

上述步骤3中，获得了具备先行进位的、能进行16种算术逻辑运算的4位ALU。在此基础上，进一步利用2组如图1.11所示的结构可以扩展成8位的ALU。如果直接把2组图1.11所示的结构串联起来，那么组间还是属于串行进行的。这时可以加多1个74182CLA部件，以类似图1.11的方式进行组织，即可得到具备先行进位的8位ALU。具体结构与图1.11类似，在此略去。

步骤5：在步骤4的基础上，设计32及64位的、具备先行进位的ALU。

在步骤4的基础上，通过级联4组先行进行的8位ALU，即可获得32位的ALU。组与组之间进一步采用74182CLA实现先行进位，则可以获得具备先行进位的32位ALU。用类似的方法，可以设计得到具备先行进位、能进行16种算术逻辑运算的64位ALU。

通过上述的步骤3~5，可以逐步地设计得到能进行16种算术逻辑运算、具备先行进位的4、8、32或64位的ALU。这种待设计的ALU是一种功能相对复杂的部件，但通过上述若干步骤的分解，通过运用“分工、合作、优化”式的系统思维，通过从简单到复杂逐步叠加功能模块的方式，就能较好地设计一种比较复杂的部件。

1.3 存储器概述

存储器包括内部存储器和外部存储器。其中内部存储器包含静态随机存储器（SRAM）、动态随机存储器（DRAM）、只读存储器（ROM）、高速缓存（Cache）；外部存储器包含硬盘、光盘、磁带等。《计算机组成原理》教材中，简要介绍存储体系，侧重于介绍SRAM、DRAM、Cache等的读写原理。本节不再重复介绍《计算机组成原理》教材中详细介绍的内容，侧重于说明如何运用系统思维构建存储体系，并设计SRAM或DRAM的组成结构。

1.3.1 存储体系的构建

根据图1.1所示的冯·诺依曼体系结构，若把运算器和控制器组合在一起形成中央处理单元（CPU），则CPU和内部存储器之间的结构如图1.12所示。CPU给出待访问内存单元的地址，并发出读命令（即$R/\overline{W}=1$）；内存在接收这些信息后，从对应的内存单元读取数据，并将数据传送到数据总线上。对于写数据的情况，CPU相似地给出地址及写命令（即$R/\overline{W}=0$），随后内存把来自于数据总线的数据写入内存的对应单元中。

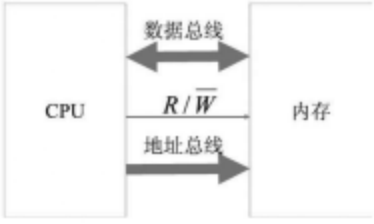


图1.12 CPU与内存组成结构示意图

现在来分析一下CPU和内存之间的速度差异。当前个人电脑普遍使用的CPU速度大概在3GHz或以上，DDR4内存等价速率在2400MHz或2666MHz。也就是说，即便在CPU速率多年极其缓慢增加而内存速率不断提升的今天，CPU和内存之间都仍然还存在着1~2倍的速度之差；十年以前甚至存在1个数量级以上的速度之差。在这种情况下，当CPU需要从内存取指令或取数据时，都需要等相对较长的时间才能得到所需的指令或数据。这就意味着如果CPU和内存之间的速率差异无法弥补的话，CPU的速率根本就无法发挥出来。若把CPU比喻成“兔子”，把内存比喻成“乌龟”，则CPU和内存之间的交互就属于“龟兔合作”问题，而不是“龟兔赛跑”问题。显然，让速度差别较大的部件之间进行合作，效率就是一个大问题。

图1.12所示的CPU与内存组成结构显然无法解决“龟兔合作”问题，这是因为短期内显然没有可能令内存的速度与CPU一致。为了解决该问题，从系统思维的角度，显然应该运用“分工”的思想增加额外的模块来解决。但这模块放置于什么位置会比较合适呢？

要想能发挥CPU的效率，那么就意味着CPU需要数据或指令时能够以接近CPU的速度快速地获得数据或指令。新增的模块显然就是需要实现此功能。考虑到缓冲部件在平衡不同速度部件之间的作用，并结合图1.12中CPU与内存的组成结构，那么把新增的模块放置于CPU和内存模块之间就是一个比较合适的方式，如图1.13所示。新增加的模块，数据或指令