从二进制到现代计算机

————《穿越计算机的迷雾》阅读报告

学号：P02114014，姓名：何忻咏

（2021 级，电子科学与技术）

摘　要： 本文讲述了阅读《穿越计算机的迷雾》过程之中思考的若干问题和个人的一些想法；并重新阐述了书中作者的一些富有价值的观点。本文由数制和布尔代数引入，然后追溯了计算机的发展历程——从由继电器制成的逻辑门到累计加法器的原型机，最终到了现代版本的计算机。同时，我们也在文章之中介绍了一些最新的技术。

关键词： 微机原理，计算机

From Binary System to Modern Computers

Xinyong He

Abstract: This article represents some thoughts and questions when reading Traveling Through the Mists of Computer, and pharaphrases several important parts in this book. The article begins with Boolean algebra, numerical system and then traces the developing process of the computer – from logic gate made up of relays to the prototype of a cumulative adder, and finally to the modern type of computer. Also we introduce some related state-of-the-art technologies.

Key words: principle of micro computer, computer

0 引言

《穿越计算机的迷雾》[1]是这个学期推荐书目之中的第一本书。幽默风趣的语言让人第一次阅读的时候，很容易使人产生“说的这些我都会”，抑或是“数电之中都讲过”的感觉，然后匆匆翻阅、不知其所旨。那么不妨先搁置一下现有知识带来的先入为主的傲慢，从最基础的物理知识开始，慢慢翻阅这本书，就会感受到书里逻辑上环环相扣奥妙。我认为，本书可以分为以下的几个阶段：

1. 从模拟信号转向数字信号，引入了二进制（莱布尼茨）和逻辑运算（布尔），并将电路中的开关的状态和逻辑运算的不同算子之间实现了对应（香农）。笔者认为，这是计算机发展史中最基础、但是最不可或缺的一步，具有里程碑式的意义。
2. 在电路-数字逻辑能够相互对应的时候，将若干个逻辑门组合产生最基础的组合逻辑电路（加法器、译码器）和时序逻辑电路（触发器）。第二部分类似于搭积木的过程，在已有电路的基础上将其进行组合，得到更加复杂的电路。
3. 在实现“全自动计算”的过程之中，为了控制计算的行为引入了计算机“指令”的概念。自此我们的电路不再是只有单一逻辑功能的电路，而是具有着由软件来控制硬件的计算机。在此基础上，介绍了现代计算机中的操作系统和中断、外设等。

《微机原理与接口技术》为《数字电路与逻辑设计》的后续课程，而这本书更像是这门课程的绪论，在逻辑上首先为我们理顺了计算机大致的发展流程。让我们拨开表面的枝枝叶叶——那些来自数电知识的疑问“这个在电路上是如何实现的？”，而抚清最深处的树干“是如何从 A 到 B 的？”。抓住主干，而这，同样与本书标题的含义不谋

而合。

1 二进制和数字逻辑的引入

在书的第一章中，首先介绍了一些有关电学的物理知识，例如电流、电压、导体等。这也是在为引出贯穿整个第二章的问题留下铺垫：“如何使用已知的电学知识，使用电路中的一个物理量来表示二进制数？”

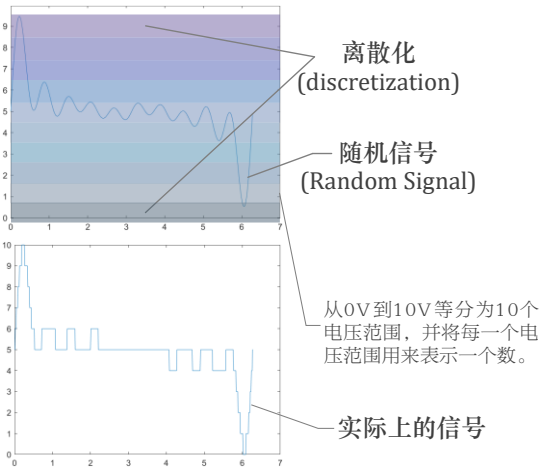


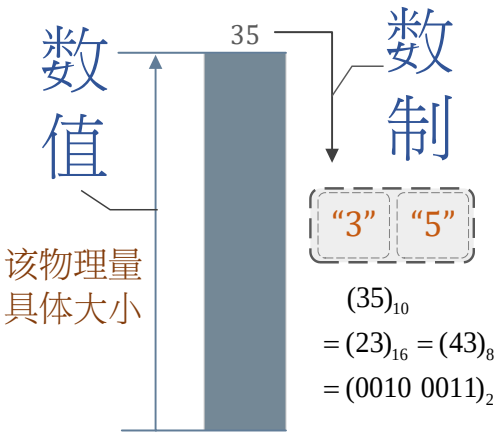
图 1-1 对于实际的数值进行离散化

很自然地，在最初我们会使用一个模拟量的实际大小来对于一个数值进行表示。例如，为了表示 35 这个数字，使用 3.5V 或者是 0.35V 的电压来表示；为了表示 35+27 我们就需要在输出的电压中，将 3.5V 和 2.7V 的电压相加输出 8.2V 的电压。在书中也提到了，“如果只使用一根导线来表示这个数的话，那么首先会产生一个电死人的电压，其次只要收到一点点的干扰，那么表示的数就会受到改变了。”为了解决这个问题，如图 1-1 所示，在十进制之中，我们采用了“离散化”的概念，将 0~10V 的电压中的每个整数值的邻域用来表示这个数。例如，0.6V 到 1.4V 之间的电压将其认为代表着“1”，这也与后续 CMOS 和 TTL 电路中规定高电平和低电平的电压范围的想法相似。

实际上，“将‘125’分别用三根线‘1’、‘2’、‘5’来表示”，看起来非常自然，但这实际上是由“数制”来表示“数值”的一个巨大的飞跃。从这里到二进制也仅仅是不同进制数下、进制之间的转化。

如图 1-2 所示，数制与数值概念上的巨大不同，在于：数值是用其本身的大小来反应一个真实的物理量的大小；然而数制的本质是使用一个对应规则(numeral system)[2]和一组符号的集合来表示实际数值的大小[1]。数值量“35.71”和“35.68”在数制上只有着 0.03 的差距，非常微不足道；但是在十进制数制之中，确实两个数字之间的差别。

图 1-2 数制与数值概念上的不同



我们从小在十进制数的环境之中长大，于是就会很不假思索的从“1”数到“9”然后进位变成“10”。但是实际上，我们在学数数的过程中，在做的有两件事：首先，我们认识了从“0”到“9”的一整套符号集（所以会有孩子学数数的时候发问“为什么 1 后面是 2 啊”）；其次，我们学的是进制数下给每一个“数制位”赋上幂次方的计数方式，即进位制（另外一种是投票时画“正”字的累积数制）。对于 n 进制的数制而言，我们需要定义其符号集 $DIGI\_SET$ 和 n 个进制下的进位运算规则 $F(Vol)$，有：

$$DIGI\_SET=\{D_0,D_1,\cdots,D_{n-1}\} \quad (1)$$

$$Vol=\overline{(D_{a_2}D_{a_1}D_{a_0}.D_{a_{-1}}D_{a_{-2}}D_{a_{-3}}\cdots)}_n \quad (2)$$

$$F(Vol)=\sum_{i=-k}^{k}a_i\cdot n^i \quad (3)$$

通过将 n 的不同幂次数乘上不同的值然后进行相加，只要 n 的幂次数 k 足够小就能够表示极小的数，使得这个记号能够表示的值可以覆盖整个实数域。不难联系到傅里叶变换之中的“正交函数集”和线性代数之中的“基向量”，而他们各是函数和向量意义之下的“数制”；甚至，这种“进位相加”的规则都是一脉相承的。

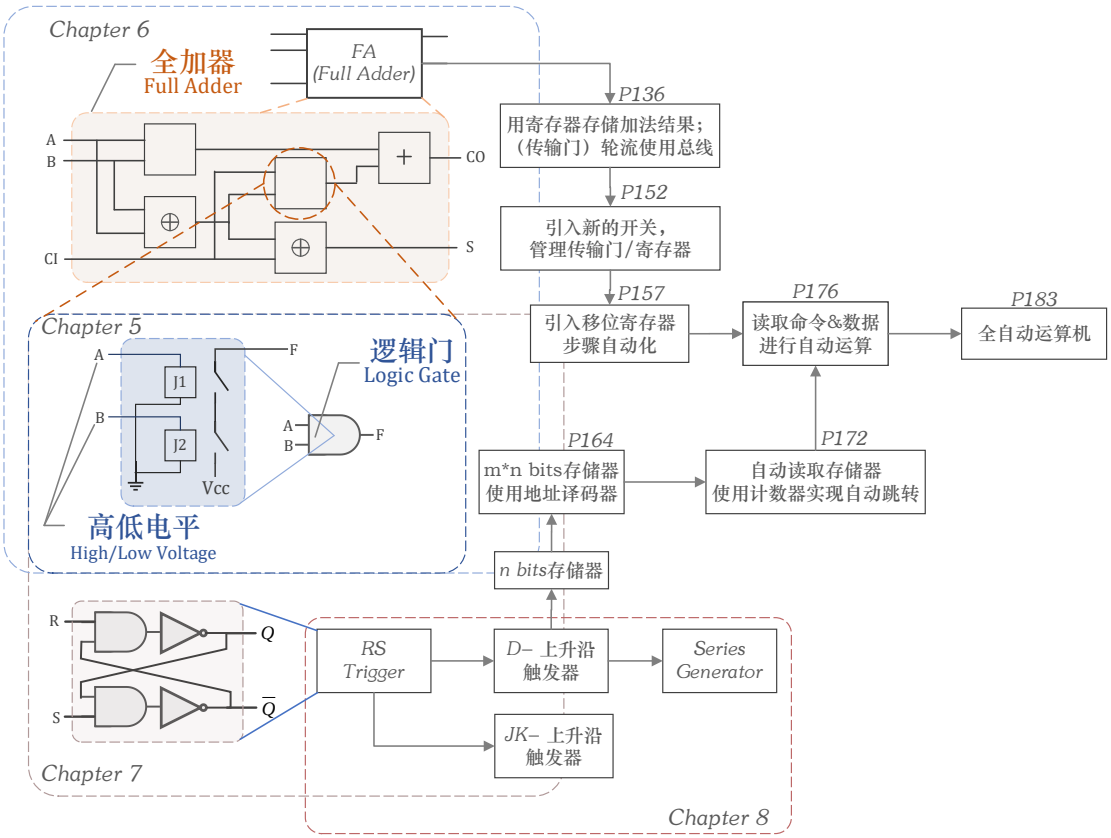


图 2-1 书第五章至第十二章逻辑框图

那么，应该如何表示 a_i 呢？采用“电压”带确实是一个方法，并且它与一开始的直接用电压值来表示数的想法相比，系统的抗干扰能力得到了大大增强。但是它存在的问题是，这一套系统具有着非常麻烦的运算规则。当“99”和“33”相加的时候，在个位数上需要“3V”和“9V”进行相加得到“12V”，然后将“1”进行进位使得各位剩下的数为“2V”，并使十位数这条电线上增加“1V”。实际上，这样的运算需要较为复杂的电路设计。

二进制的伟大之处在于以下几点：

1. 仅使用了两个状态，使得在电路中实现二进制能够较为简单，可以将”1”和”0”用电路之中开关的闭合和断开来进行表示。
2. 采用二进制之后系统的鲁棒性和抗干扰能力得到了进一步增强。
3. 每个状态的使用频率更高。能够减少电路设计的冗余。
4. 与人的认知逻辑相似。在布尔的逻辑代数之中，规定了二进制数的与、或、非等运算，给二进制这个计数的方式赋予了逻辑学上的意义。

在未来的量子计算机之中使用的是三进制——对应着量子的三种不同的状态。在蔡超，金翊[3]等人的工作之中，使用了对称三进制 $[\underline{1},0,1]$ 来表示数,并采用光的三种状态来分别表示对称三进制中的基数. “$\underline{1}$”即”-1”的引入，使得在进行减法时无需经过补码的运算，可以直接将负数输入加法器实现.

从人类文明的最初产生，到拥有“数制”和“二进制”的概念，再到基于二进制的逻辑学和布尔代数的产生，其中走过的几千年的时间都在为计算机的产生奠定了最基本的理论基础。

2 计算机电路的逐步发展

“电”的概念其实产生了数百年了，从莱顿瓶的时代开始人们便一直孜孜以求的研究它；但是，直到 1936 年香农的《继电器和开关电路的符号化分析》将布尔代数和电路理论结合在一起，方为数字电路的发轫之时。从这里开始，逻辑门到组合逻辑电路，到触发器寄存器再到一些存储单元构成了

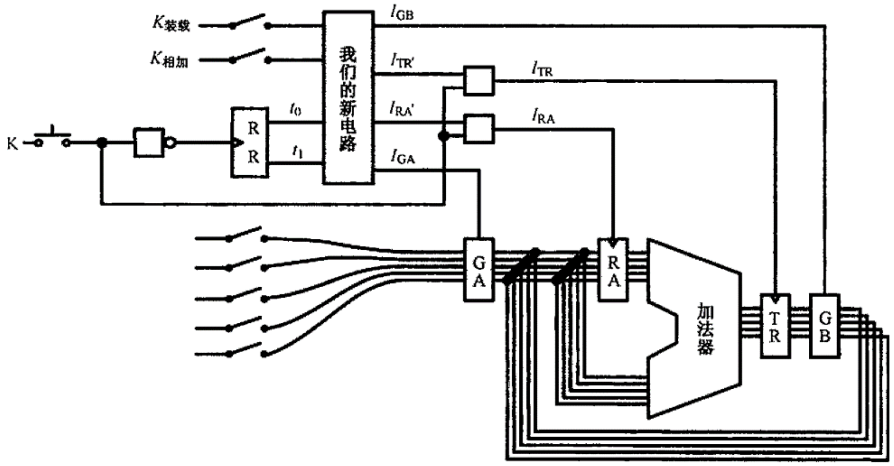


图 2-2 使用循环移位寄存器来简化装载和相加过程

一脉相承，并逐渐构成了具有现代意义的计算机。

书中第五章至第十二章在逻辑上环环相扣，具体的演变过程如图 2-1 所示。在阅读这一部分的内容的时候，我们上学期所学的数字电路的知识被贯穿在了一起。图 2-1 上半部分讲述的是数字电路之中的组合逻辑部分；下半部分所述的是时序逻辑电路的演进。但是与数电相不同的是，数字电路讲究“如何分析这个电路之中的可能输出”或者是“为了这样的输出应该如何设计电路”；但是容易给人一种将“组合逻辑”和“时序逻辑”相割裂开的错误观点。而在这本书之中，我们想知道的是，“根据我们已知的知识，将很多‘黑箱’组成一个具有计算功能的机器？”

这几章的展开顺序是先由逻辑门引出的组合逻辑电路和时序逻辑电路，然后最终在“自动加法器”的部分将两者的具体应用电路合二为一。到这里，实际上已经站在了高于数字电路的角度来窥探计算机组成原理的奥妙了。

为在加法器的后端加上一个寄存器，并在输入端的前面加入传输门，从而达到轮流使用总线的效果，以防止结果和新数据之间电平高低产生冲突。这是后续了解总线技术之中“数据总线”[4]留给我们的最初的印象。这一步改进其实在人所设想的情理之中，并不难想到。到了下一步的改进，从 P141 图 10-9 之中，采用铡刀开关和按键开关和自定义的新电路来控制传输门和寄存器的使能，到 P145 图 10-10 之中引入了移位寄存器和与门的设计，是设计上的一个巨大的飞跃。在阅读这个章节的时候，首先会被作者的定义弄的有些发晕；但是到了最后会因为这一步巧妙的设计而“伸颈、侧目、微叹、以为妙极”。

表 2-1 基于循环移位寄存器优化的累计加法器的设计

K(相加)=1，进行重复累加的情况下：

1. 初始状态下 GA 传输门始终打开，$I_{RA}^{'}$ 为高电平，但是 I_{RA} 需要一个来自开关的高电平才能 $0\to1$。所以此时的开关数据并不会被触发器锁存。
2. 第一次按下开关 K，由于非门作用，移位寄存器的 CP 端收到下降沿。但是 I_{RA} 被使能，被加的数据被送入寄存器之中。
3. 第一次松开开关 K，移位寄存器移动一位，使得“我们的新电路”进入下一个状态。此时 I_{GB} 打开，$I_{RB}^{'}$ 为高电平，但是需要一个高电平使得连接到寄存器 RB 的两个输入端都为‘1’，才能让 I_{RB} 输出为 1.
4. 第二次摁下开关 K，此时 I_{RB} 的输出为高电平，得到的加法结果被加法寄存器 TR 锁存
5. 第二次松开开关 K，移位寄存器得到一个上升沿，重新进入初始状态（1 状态）。TR 之中的数制被送入寄存器 RA 之中。

在这一步之中的设计非常巧妙，使得操作非常便捷的同时，解决了前文之中描述的

“时间差”的问题。通过图 2-2 之中的非门，使得开关的按下和松开分别赋予了不同的意义。当开关松开的时候，进入一个新的状态；这个新状态的标志就是重新打开一个新的传输门，使得数据能够稳定的加载入电路之中。然后在按下开关的时候，已经达到稳定的数据随着使能信号的来临被存储进寄存器之中。这个做法的关键是将两件客观上需要一定时间间隔的动作：“打开传输门”和“使能寄存器”这两件事情分别对应到按键 K 的按下和松开，在这个想法的基础上来设计“我们的新电路”。

在这里，我们的电路之中一个重要的部分，控制器(Control Unit, CU)已经逐渐从运算器之中逐渐的分离了出来了。在这个具有半自动功能的连续加法电路之中，这个控制器的部分控制着加法器在何时打开传输门传入数据，何时做加法并将加法的结果锁存入寄存器之中。表 2-1 之中所描述的一个时序流程其实是被写在了“我们的新电路”之中。但是由于逻辑门的限制，如果想要改变运算的逻辑，就必须重新设计电路之中的部分。这里，也是后续的“全自动计算机”之中区分“软件”和“硬件”的一个引子。

同样的，采用这样的“时序”的概念，也可以设计这样的控制器，自动逐条读取存储器之中的数据。

图 2-1 之中体现的另外一个思想是，从继电器中的高低电平，到逻辑门，再到中规模逻辑电路，其本质实际上是逐层封装的过程。在最初的第五章中、使用继电器实现逻辑门电路的设计时，我们关注的是继电器上开关的是否闭合，以及其产生的高低电平。到了书本的第六章之中，当我们需要分析全加器的输入和输出之间的关系时，我们只需要专注在真值表的逻辑关系上，使用卡诺图来对于逻辑关系进行化简。然后，我们只需要将封装好的逻辑门应用于全加器的电路之中，内部继电器的开关状态被忽略了，需要关注的仅仅是其输入和输出。这种“封装”的思想使得我们能够将已有的电路组成一个更大规模的电路，或者对于电路的结构进行更新，而不是重复着造轮子的过程。在意法半导体的 Cubemx 软件之中，可以直接通过图形化的界面来配置时钟和 GPIO 的输入输出属性，系统便可以自动生成配置其中的寄存器所需要的代码；在 pytorch 之中，我们想要定义新的神经网络时，只需要对于继承一个父类并对其进行重写。这样的逐层封装能够为开发者提供更加清楚的开发逻辑，也能够为开发带来更多的便利。

最后，我们将自动加法（图 2-1）和自动取数（图 2-2）的电路合二为一，使之能够执行自动读取一段内存之中所存储的数据并对其进行相加，然后对于停机的指令做出反馈。这里，一段指令能够被我们的电路进行译码，转化为寄存器不同的存取状态和传输门的开合，从而对于整体的硬件电路实现了控制。而在现实之中，我们所写的代码在编译之后，会被转化为二进制的代码并存储在计算机的只读存储器(Read Only Memory, ROM)之中，在需要阅读的时候会被载入随机存储器(Random Access Memory, RAM)。程序计数器(Program Counter, PC)也会逐条的从存储器之中来读取程序。

在上课的时候，老师引出的问题是，“计算机如何识别指令和数据？”，在冯 •诺伊曼结构之中，指令和数据同时都是以二进制的形式存储在计算机的相同位置，所以如果将指令和数据混淆将导致较为严重的后果。在实际中，计算机在开始执行程序的时候，会从固定的地址开始读取指令集，并且我们需要确保计算机开始读的内存地址是一段指令。

在指令的设计之中，系统对于指令就能够识别出其所需的数据位的数量，那么，系统就可以认为接下来从存储器之中读取的数据存储的也都是“数据”而非“指令”。当我们使用汇编语言输入：

```
1. MOV  X1，23
```

首先计算机处于取指令阶段，根据输入的二进制数，计算机首先接收到的指令为”MOV”。然后计算机进入执行指令的阶段，然后在译码器之中判定接下来的两个内存单元存储的都是需要访问的内存的地址。通过这样“取指令”和“执行指令”使得计算机能够区分程序之中的指令和数据。

在有了指令和数据之后，我们能够写更多更复杂的程序。在此时就会有了“机器语言”和“高级语言”的区分了；目前的高级语言将更加的类似于人类自然语言的习惯，

方便人们的理解。但是他的本质还是对于运算器进行控制。在面向对象编程(Object Oriented Programming, OOP)之中，我们编辑对象的“方法”，从更加宏观的角度来改变指令集的组织方式，从而对于运算器的运行方式进行了改动。

3 成为完整意义上的“计算机”

本文第二节所述的过程曲折漫长，经过了数十年的反复迭代更新，终于能够使得计算机从最基础的逻辑门电路演进到能够识别最简单的指令码。到了这里，计算机之中最基础的两个结构——“控制器”和“运算器”已经基本上确定了他们的结构，我们也称其为 CPU(Central Processing Unit)。下面需要做的事情是，完善 CPU 和输入、存储、输出端口之间的联系，从而使它能够成为一个完整意义上的“电脑”。

为了使得 CPU 之中的运算器能够尽可能满负荷的运作，我们需要保证 CPU 和存储器之间的读写速度与 CPU 的运算速度相匹配。然而显示的情况是，存储器的速度无法达到 CPU 运算的速度。为了解决这个问题，采用了流水线的方法，将运算器的运算步骤进行分级的处理，并在级间使用触发器来保存结果。这样子做能够对于运算器的运算用统筹的方法进行规划，可是会导致计算机的跳转指令对于程序的运行产生影响。因此最终的选择是采用除了流水线之外的第二种办法，在 CPU 和主存储器之间，采用一小块高速缓存（SRAM）来存取计算机即将要访问的内容。这样以来，高速缓存能起到蓄水池的作用，从而平衡了 CPU 和主存储器之间读写速度不同的问题。

为了实现更好的人机交互，后来的计算机之中采用了各种各样的外设——鼠标、键盘、显示器，通过这些外设来方便人们向 CPU 之中发送数据，或者是理解由 CPU 发送而来的数据。但是实际上我们向计算机的内部输入的数据是一串二进制代码，并且在计算机的内部也是使用二进制对于信号来进行处理的。IO 口，或者叫 GPIO(General Purpose Input & Output), 是计算机和其他外围设备交换的物理接口。通过 GPIO 的引脚上的高低电平的变化，我们需要的数据便源源不断的输入进了计算机之中。

为了规范电平的高低应该如何变化，让电平高低的变化能够代表的意义得到确定，因此各种硬件通信协议应运而生，例如 UART，I2C，SPI 等. 这样以来，在不同的计算机之间如果采用同样的通信协议，那么信息是可以畅通无阻地传递。

在今天，只有在树莓派这样的设备上面才能见到裸露的物理针脚的存在了，在目前主流的设备上，将通信协议对应的信号线和电源线进行封装，从而得到了目前，在你我的电脑上的各种各样的接口——他们是属于昨天的 RS232、VGA，正在大范围使用的 HDMI、USB 和逐渐壮大的 Type-C。

计算机需要收到一个电平的信号之后，需要立即高速访问键盘等外设暂存在直接存储器(Direct Memory Access, DMA)中的信息。因此，“中断”的机制产生。

随着计算机的功能和外设越来越多，需要拥有一个程序来管理内存的分配，文件的管理和各种各样的中断的事件的轻重缓急。因而有了操作系统。

外设的到来使得 CPU 的有限的计算量受到挑战；因此在现代的电脑中使用了声卡和显卡(Graphic Processing Unit). GPU 采用了数量众多的计算单元和超长的流水线，但只有非常简单的控制逻辑并减省了 Cache。使得 GPU 能够胜任大规模的并行运算，从而可以胜任今天的图像处理等新兴的算法.

[1]. 李忠. 穿越计算机的迷雾[M]. 北京；电子工业出版社；2011.1.

[2]. 维基百科. https://zh.m.wikipedia.org/wiki/%E8%AE%B0%E6%95%B0%E7%B3%BB%E7%BB%9F

[3]. 蔡超,金翊,包九龙,汪宇涛.三值光计算机的对称三进制半加器原理设计[J].计算机工程,2007,(17).

[4]. 陈惠鹏, 徐冰，刘松波. 新编微机原理与接口技术[M]. 北京；电子工业出版社；2015.8.