回到课程资源

教学案例

第二章　数字信息的二进制表达（The Binary Expression of Digital Information）

案例 1：在数字系统中使用二进制的实例

温度传感器以二进制输出温度值，系统读出的温度值以 ASCII 码表示，用“F” 表示 冰点（freezing (0-32)）以下，“B” 表示沸点以上（ boiling (212 or more)），“N” 表示正常值（normal），显示器将 ASCII 码转换成对应的字母显示出来。该数字系统的目标是读取温度传感器的值，如果温度是 32 华氏温度或更低，显示字母“F”（冷）；如果温度在 32 和 212 之间，显示字母“N”（正常），如果是 212 甚至更高，显示字母“B”（热）。温度传感器的输出是 8 位，用二进制数表示温度在 0 到 255 之间的数。显示器的输入是 7 位，根据接收到的 ASCII 编码，显示对应的字符。

图2.1给出了温度传感器在数字系统中期望的输出和输入值。每根线上的值是1或0。数字系统的输出是7位，与显示器的7位输入相连。该数字系统的工作过程是：如果输入小于或等于二进制数00100000，即十进制数32，那么输出就是1000110，对应ASCII码中的字母F；如果输入大于或等于二进制数11010100，即十进制数212，那么输出就是1000010，对应ASCII码中的B。对于其他的输入值，输出就是1001110，对应ASCII码中的N。图中给出的例子是输入00100001，即十进制33，输出为N。

这个例子解释了输入是数字位—0和1—输出也是数字位的数字系统是如何工作的。后续章节会讲解如何构建实现这种数字系统的电路。

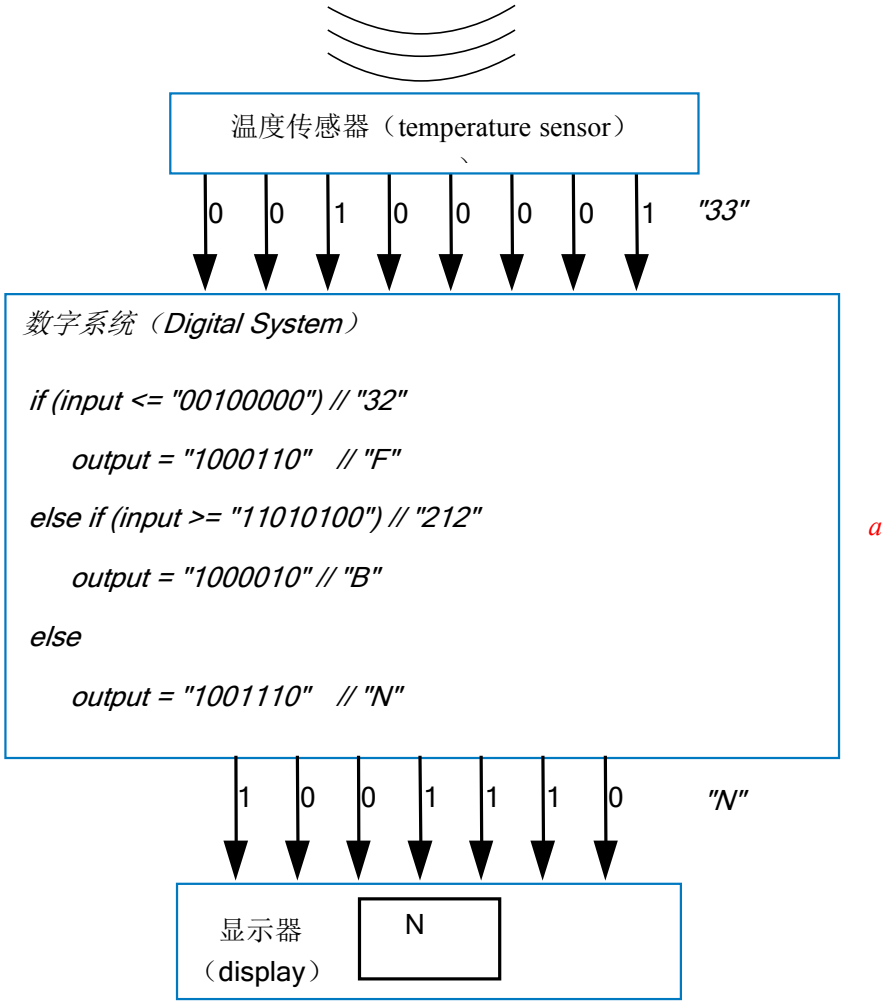


图 2.1 在数字系统中使用的二进制

案例2：十六进制应用的实例

无线射频识别器（RFID tag）是一个自动响应射频信号的芯片，即自动发送一个识别数字的唯一信号。从20世纪90年代以来，射频识别器就开始用在自动玩具转发器、奶牛（如图 2.2 (a)）和狗上(该识别器被移植到狗的皮肤下)。射频识别器典型的结构是有一个使用射频信号的电路，用来驱动芯片，这样就不需要电池了，从而减少了识别器的尺寸，延长了其使用寿命。

图 2.2 (b)解释了移植在牛里的识别器是如何构造的。该识别器存储了 32 位数据，前 8 位对应一个省的编号，接下来的 8 位对应一个城市的代码（这些表示了牛的出生地），最后 16 位表示了动物的特有编码。这种无线设备需要识别器的输入是十六进制形式，编程设备正确地把 0 和 1 组成的序列写到识别器的 32 位存储设备中—用 8 位十六进制数表示比 32 位二进制数表示错误率更小。图 2.2 (c)给出了对应的十进制数，图 2.2 (d)给出了这些十进制数对应的二进制数，图 2.2 (e)把这些二进制数转化成了十六进制数。最后，图 2.2 (f)给出了输入到编程设备的 8 位十六进制数，然后把这个数编程为与 32 位的二进制数相对应的特定的标签。标签的内容用二进制表示，程序员在读/写时用 16 进制表示。一旦这个数被编程到标签，标签与牛连接，射频识别器的阅读器进行连续读数，以确保每头牛每天被挤奶一次。

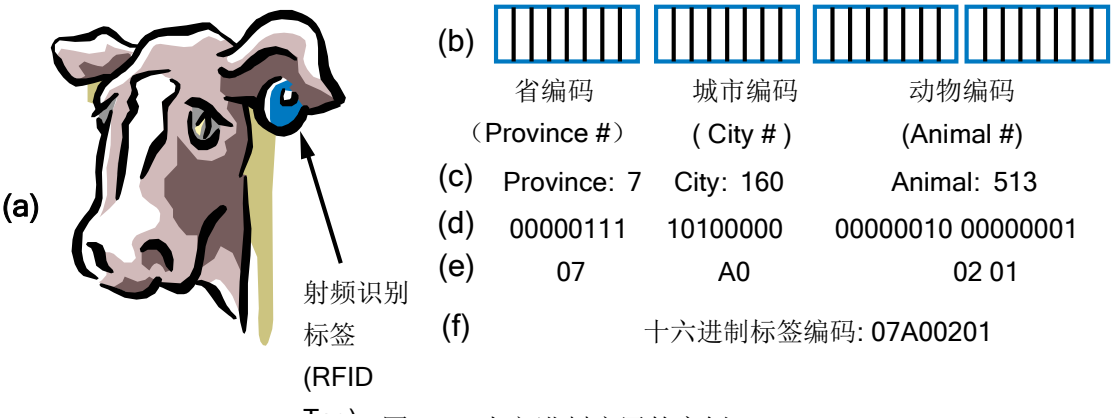


图 2.2　十六进制应用的实例

案例3：把十进制转化为二进制的实例

这个例子解释了把十进制转化为二进制来构造一个数字家用设备。吊扇就是最常见的家用设备，由一个远程控制器控制来打开或关上电扇，详细见图2.3(a)。所有的电扇和远程控制器都以相同的无线频率运行。因为，一个房子里可能同时有几个电扇，就需要一种方法来阻止一个电扇的远程控制器影响其他的电扇。常用的方法就是对无线信号的通道数进行编码，这样，每个电扇和远程控制器对就公用一个独立的通道。当一个电扇的模块检测到远程控制无线频率时，在响应之前，会先检查检测到的通道与自己固有的通道是否匹配。设置这样通道的常用方法就是在远程控制器中安装一个拨码开关，在其他的电扇里安装其它的拨码开关。一个8管脚的拨码开关有8个开关，每一个都可以置于向上或向下的位置，如果对应位置的开关是向上位置，那么8个输出位中对应的位是1；如果位置朝下，那么对应位是0.因此，这样一个拨码开关就表示了 $2^8=256$ 种不同的值。

假象电扇制造商希望给电扇/远程控制器设置一个特定的通道73，则首先需要把73转换为二进制数，见图2.3(b)中所示的01001001。然后，需要把拨码开关设置在电扇模块和远程控制器里，上下位置设置见图2.3(c)所示。当只有检测到远程控制器的频率的编码通道值与拨码开关的值相匹配时，电扇才会运作。

在下面的情况下，家庭正好买了两对电扇/远程控制器，其设置了相同的通道（通道值是256种的一个），这时，一个远程控制器会影响两个电扇，这时，主人可以不用撤掉一个电扇，只需要换一个就行。

图 2.3 为 DIP 开关控制风扇通道的例子。在工厂的车间里，天花板上的风扇接收器被设置成对通道 73 做出响应，先将 73 转换成二进制，再设置相应的 DIP 开关。

案例4：数字系统的几种输入

例如，一个键盘有4个按钮，如图2.4所示，颜色为红、蓝、绿、黑。设计者可以设计这样一个电路，当红色按钮被按下时，键盘的三位输出值是001；蓝色按钮被按下时，输出值是010；绿色按钮被按下时，输出值是011；黑色按钮被按下时，输出值是100；如果没有按钮被按下时，输出值是000。

更通用的数字现象是英文字母。每一个字符都来自于一组有限的字符集，因此，轻敲键盘产生数字化数据，而不是模拟化数据。通过给每一个字符分配一比特的编码，数字数据可以转化成比特。英文字母最流行的编码是ASCII编码，表示美国信息交换标准码，发音为“askey”。ASCII编码是用7个比特编码每个字符。例如，大写字母A的编码是“1000001”，大写字母B的编码是“1000010”，小写字母a的编码是“1100001”，小写字母b的编码是

“1100010”，因此，字母“ABBA”的编码是“1000001 1000010 1000010 1000001”。ASCII编码给所有的26个字符定义了7比特编码，还有0到9的数字、标点符号和运算控制符。ASCII里总共有128种编码。ASCII的编码子集见图1.9，由于国际语言的支持，另一种编码也越来越流行，即Unicode。Unicode编码采用每个字符16比特，代替了ASCII中的7位，表示世界上各种语言的字符。

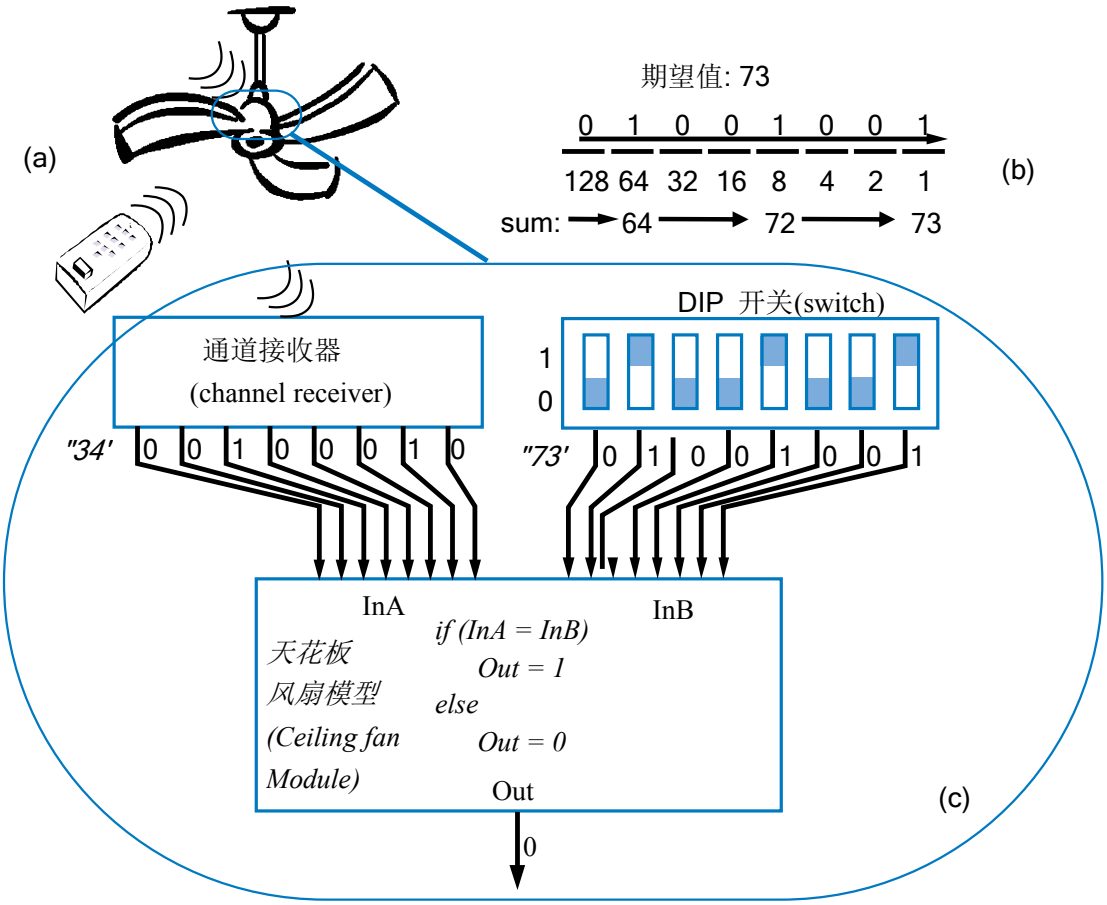


图 2.3 DIP 开关控制风扇通道

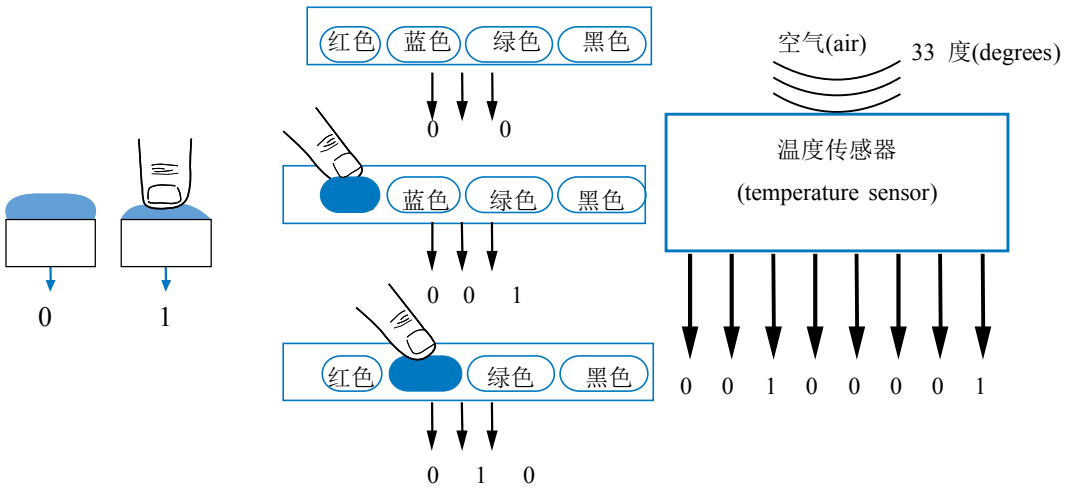


图 2.4 数字系统的几种输入

第三章 逻辑代数基础 （Basis of Logic Algebra）

案例 1：逻辑代数的发展

1847 年，英国数学家乔治·布尔（George Boole）首先提出了描述客观事物逻辑关系的数学方法——***布尔代数（Boolean Algebra）***。起初它并不是用来构建数字电路，而是用来处理人类的逻辑行为和思想的。代数是数学的一个分支，它主要用字母或者其它符号来表示一些数字，另外还可以根据某些规则对其进行组合。布尔代数使用布尔变量进行数学运算，该种变量只有 1 和 0 两种取值，其中 1 代表真，0 代表假。布尔代数使用运算符“与、或、非”对布尔变量进行运算，并返回 1 或 0 值。

1938 年，克劳德·香农（Claude E. Shannon）在他的麻省理工硕士毕业论文中阐述了布尔代数是如何被应用到基于开关器件的电路系统中的。他在论文中用“1”来代表开关的闭合状态，用“0”代表开关的断开状态，这样他就把布尔变量与开关的状态联系起来了。他将布尔代数用于设计电话继电器开关电路（Switching Circuit），因此布尔代数又称为***开关代数（Switching Algebra）***。他的理论被大家认为是现代数字电路系统发展的一块基石。正是由于布尔代数有很多公理、定理、推论和假设，我们才能用它来设计数字电路系统。换句话说，自从有了布尔代数以后，人们才能利用数学去设计数字电路。后来，布尔代数被广泛地应用于解决数字逻辑电路的分析与设计上，所以布尔代数也称为***逻辑代数（Logic Algebra）***。

第四章 逻辑门电路（Logic Gates Circuits）

案例 1：电子开关的发展过程

电子开关的发展非常迅速，尺寸在以惊人的速度缩小（shrinking）。从最初 1930 年代的继电器（Relays），到 1940 年代的真空管（Vacuum tubes），1950 年代的分立晶体管（Discrete transistor），1960 年代只有几个晶体管组成的集成电路（Integrated circuits，ICs），发展到由几十个、几百个、几千个...晶体管组成的集成电路，其发展过程如图 4.1 所示。

(a) 继电器（relay）　(b) 真空管（vacuum tube）　（c）分立的晶体管（discrete Transistor）　（d）集成电路（IC）

图 4.1 电子开关的发展过程

最早使用的电子控制开关是 1930 年代的电磁继电器开关，如图 4.1(a)所示，被电话生

产商应用于切换电话来电。继电器用磁体当作控制输入，当控制端具有高电压时该磁体被磁化。磁体将一片金属拉下，导致源输入到输出的接通——类似于拉下吊桥连通一条路到另一条路。当控制输入变回 0V，金属片再次弹起（或许被小弹簧弹起），从而隔断源输入和输出。在电话系统中，继电器使得来电从一部电话连通到另一部电话，不需要像以前那样由人工手动操作将电话线接通到另一部电话。

继电器是依靠金属部件的起、合工作的，因此速度相当慢。在 1940 年代到 1950 年代之间，真空管最初被用于放大如电报中那些微弱电信号，并开始在计算机中取代继电器。图 4.1(b)所示的真空管，在设计上类似于一个灯泡。在灯泡中，电子流过灯泡内部真空管的灯丝，使灯丝发热并发光——由于灯泡内缺乏氧气，所以不会燃烧。在一个真空管内，管内加入了一个终端，把真空和灯丝隔开。如果终端加上正电压，电子可以通过真空从灯丝运动到终端，终端的正电压在灯丝和终端之间会产生连接。真空管没有运动部件，因此要比继电器快得多。

尽管真空管比继电器要快，但他们会消耗大量能量，产生大量的热，并且故障频繁。

1947 年问世的晶体管，导致了更小更低能耗的计算机的出现。固态（分立）晶体管如图 4.1(c)所示，用小片的硅，掺杂一些额外的材料，形成一个开关。由于这些开关使用“固体”材料而非管子或者继电器中的起、合部件，他们通常被称为固态晶体管。固态晶体管更小，更便宜，更快，相比于真空管更稳定，因而成为 1950 到 1960 年代主流电脑的开关。

1959 年集成电路（Integrated Circuit, IC）问世，使计算能力真正有了革命性的改变。集成电路是将手指头大小的一片硅中封装了许多小晶体管的芯片。原来电路板上需要 10 个晶体管组成 10 个分立的电子组件，现在可以在一个组件，即一个芯片中实现了。图 4.1(d)显示了一个集成几百万个晶体管的 IC。尽管早期的集成电路只集成了数十个晶体管，如今的 IC 技术已经能够集成十亿个晶体管了。IC 技术将晶体管缩小到了完全不同的大小。真空管（大约 100 毫米长）相对于现代集成电路晶体管（大约 100 纳米），就像摩天大楼（大约 0.5 千米）相对于信用卡的厚度（大约 0.5 毫米）。

IC 晶体管更小，更稳定，更快，而且相比于分立晶体管能耗更少。因此，到目前为止，IC 晶体管更常用作计算机开关。1960 年代早期的集成电路能集成数十个晶体管，现在被称为小规模集成（SSI）。随着晶体管大小的缩减，1960 年代末和 1970 年代初，集成电路可以集成数百个晶体管，称为中规模集成(MSI)。1970 年代发展成大规模集成（LSI）电路，可集成数千个晶体管。超大规模集成（VLSI）芯片出现在 1980 年代。从那以后，集成电路不断增加容量，达到十亿晶体管。为加深你对这些数字的理解，我们不妨看看 2009 年笔记本电脑的处理器，像 Intel Atom 和 Celeron 处理器，只需要大约 5000 万个晶体管；手机处理器，比如 ARM 处理器，只需要几百万个晶体管。如今许多的高端芯片，比如互联网路由器中的芯片，包含几十甚至几百个微处理器，可以想象能够包含多少个晶体管。

案例 2：用开关和逻辑门构建电路的区别

CMOS 晶体管在简单的电路中可以被当作开关器件来使用，然而用开关器件来搭建复杂的数字电路无异于用小石头来建桥一样困难，如图 4.2 左边部分所示。当然，从原理上来说用这些基本的模块来搭建一些东西是可行的，但其过程确实十分复杂。如果用开关器件作为基本的电路模块的话，那么其工艺水平也实在过于低级了。幸运的是，布尔逻辑门的主要作用就是搭建数字电路，这比用单纯的开关器件要简单多了。

虽然逻辑门是由开关电路构成的，但是由于用开关电路直接构造的逻辑电路很难分析和设计，用逻辑门则比较容易创建方程并转换成逻辑电路，因此数字电路的设计一般是在门级，而不是在开关级实现的。图 4.2 右上部分给出了比较形象的说明。图 4.2 右下部分则是一个具体的用开关和逻辑门构建电路的实例。

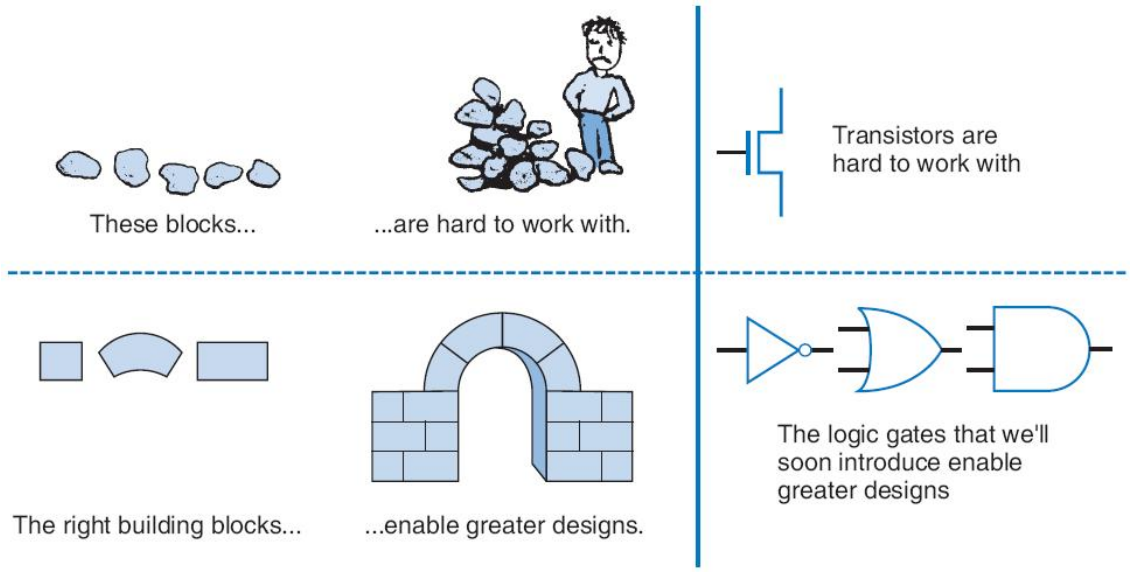


图 4.2 用开关和逻辑门构建电路的区别

案例 3：硅片上的晶体管

MOS 管是金属－氧化物－半导体场效应管 Metal-Oxide-Semiconductor- Field-Effect Transistor）的简称。MOS 管是一个受输入电压控制的器件。MOS 管的栅极（Gate）与源极（Source）、漏极（Drain）均无电接触，称为“绝缘栅极”。 根据导电沟道不同，MOS 管可分为 N 沟道和 P 沟道两类，简称 NMOS 管和 PMOS 管。图 4.3 所示为芯片上 NMOS 管的结构示意图。该类场效应管具有开关的三个部分：（1）源输入，称为源极（2）输出，称为漏极—也许是根据带电粒子流向漏极如同自来水流向下水道一样而命名的。（3）控制输入，称为栅极--也许是根据栅极阻止带电粒子流动就像门阻止人通过一样而命名的。源极和栅极之间形成一个沟道，栅极与沟道之间有一层很薄的二氧化硅绝缘层。由于是“绝缘栅极”，所以输入电阻很大（$10^{10}\Omega$ 以上），输入电流可以看作为零。当栅极—源极之间不加电压时，漏极—源极之间是两只背向的 PN 结，不存在导电沟道，也就不会有漏极电流；随着栅极—源极之间的电压加大，栅极产生的电场“场效应”促使漏极—源极之间导电沟道的阻抗相应减小，漏极—源极之间的导通电流也随之增大。漏极—源极之间不再具有 PN 结隔离，形成导电沟道，开关导通。

由于 MOS 管的导通是依靠栅极对载流子的吸引力实现的，因此在实际使用时，必须保障栅极具有最大吸引力，即在导通时，NMOS 栅极电平应高于其余电极，而 PMOS 栅极电平应低于其余电极。因此，利用此类开关器件传输状态时，NMOS 只能传递低电平（放在接地通道中），而 PMOS 只能传递高电平（放在接正电源的通道中）。

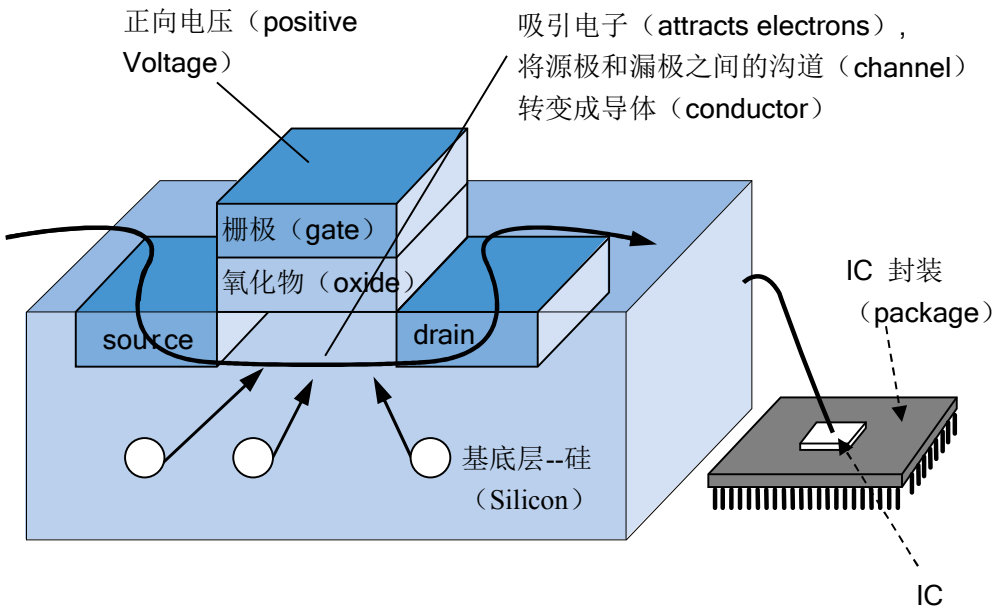
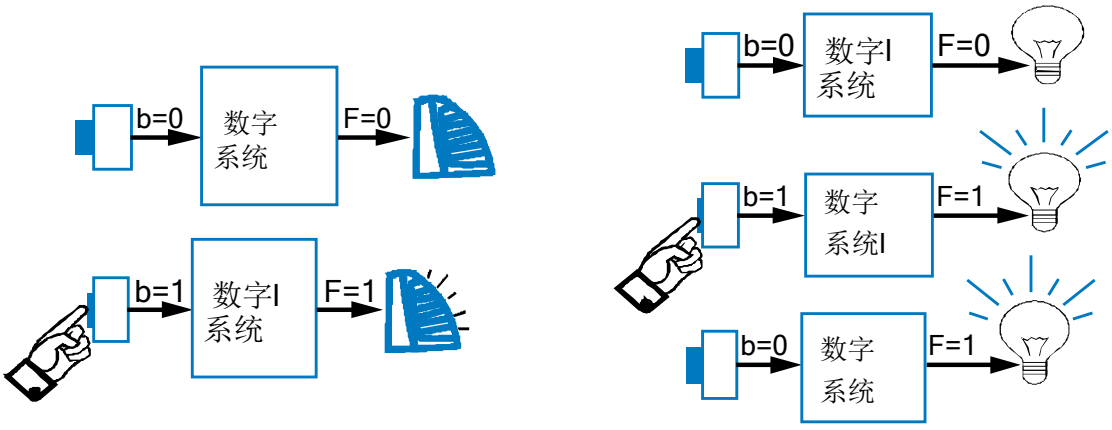


图 4.3 硅片上的晶体管

第五章 组合逻辑电路设计基础（Basis of Combinational Logic Circuit）

案例 1：组合逻辑电路和时序逻辑电路的区别

数字逻辑电路分为两大类，一类是组合逻辑电路(Combinational Logic Circuit)，一类是（Sequential Logic Circuit)。如果一个数字电路的输出只依赖于当前输入值的组合，则该电路称为组合逻辑电路。例如，图5.1(a)给出了一个门铃系统。如果按下按钮，输入b=1，则输出 F=1，门铃响。若没有按下按钮，输入b=0，则输出 F=0，门铃不响。然而，在时序逻辑电路中，输出值不仅取决于当前输入值的组合，还和过去的输入值有关，即电路中包含存储器件。例如，图5.1(b)中的切换灯；按下按钮，输入b=1，则输出 F=1，灯亮；此时即使释放按钮，输入b=0，灯仍然处于亮的状态，输出 F=1，这是由于数字系统中有以前存储的信息。只有重新按下按钮，灯才灭。因此，对于时序电路而言，不能只从当前的输入确定输出 F 的值。



(a) 组合逻辑电路　　(b) 时序逻辑电路

图 5.1 组合逻辑电路和时序逻辑电路

案例 2. 设计者实现一个数字系统的方法

设计者实现一个数字系统的常用方法就是选择两种的实现方法中的一种——编程一个微处理器，或者设计一个固定的数字电路，这被称为数字设计。

作为这种选择的一个例子，考虑一个简单的应用，即在黑暗的房间，无论何时有移动，都可以打开灯。假设运动检测器有1个输出线命名a，当检测到运动时，输出1；否则，输出0. 假设灯传感器有一个输出线b，当灯检测到时，输出1；否则，输出0. 当线F是1时，表示开灯；当是0时，表示灯灭。这样的一个系统图见图5.2。

设计的问题就是探测器如何设计。探测器的输入是a和b，输出是F，当在黑暗时检测到运动时，灯就亮。探测器应用是作为一个数字系统实现的，因为它的输入和输出很明显都是数字化的，而且只有两种可能值中的一个。设计者也可以通过一个微处理器来实现探测器模块（见图5.2）或者是设计一个数字电路（见图5.2）.

在组合逻辑电路(Combinational Logic Circuit)的设计中，逻辑抽象(Logic Abstract)是关键，需要仔细分析各种逻辑关系(Logic Relationship)和因果关系，必须包括所有情况，不能遗漏。现在考虑一个简单的应用，在黑暗的房间，无论何时有移动，都可以打开灯。例如，人在黑暗的房间中行走，有移动传感器探测是否有人到来，灯光传感器探测灯是否亮。现在要设计一个探测器，当探测到有人到来，而灯又没有亮时，则灯亮，如图 5.2 所示。这个探测器可以用两种方式实现。一种是用微处理器实现，一种是用门电路实现。如果用门电路实现，假设移动传感器有 1 个输出线命名为 a，当检测到有人移动时，输出 a=1；否则，输出 a=0. 假设灯传感器有一个输出线命名为 b，当灯亮时，输出 b=1；否则，输出 b=0. 当输出线 F=1 时，表示执行开灯的操作；当 F=0 时，表示执行关灯的操作。设计这样的一个系统我们可以得到输出 F 的表达式 F = a b'。这是我们设计的第一个数字电路。

探测器的应用是作为一个数字系统实现的，因为它的输入和输出很明显都是数字化的，而且只有两种可能值中的一个。

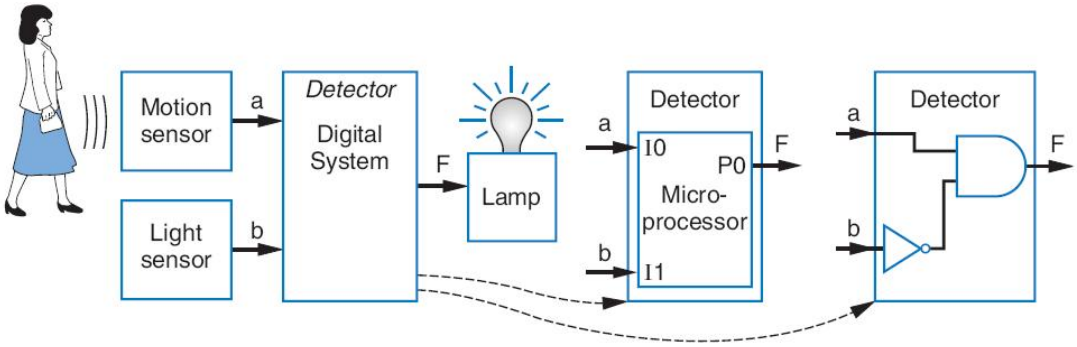


图 5.2 用逻辑门构建电路

案例 3 汽车安全带报警灯系统（Seat Belt Warning Light System）

该系统能够在汽车钥匙插入钥匙孔时，如果司机没扣紧安全带的情况下点亮警示灯予以警告。如图 5.3 所示。

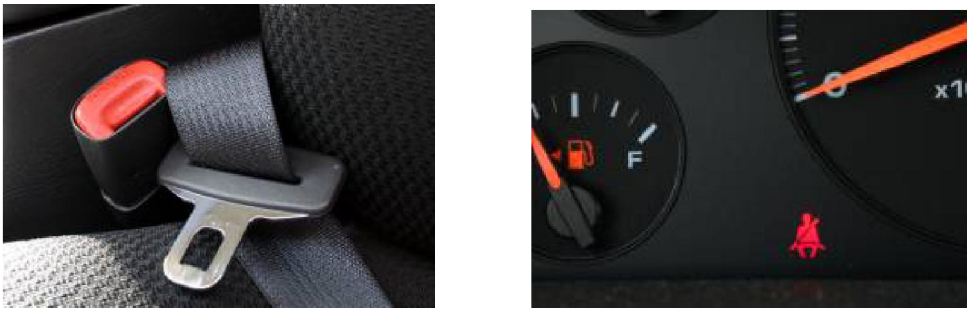


图 5.3 汽车安全带及警示灯

该系统应该包含以下几种传感器：

1.能够识别司机的安全带是否扣紧的传感器（假设其输出信号是 s(sensor)，如果 s 为高电平的话则表示安全带已经扣紧）；

2. 能够识别车钥匙是否已经插入车锁的传感器（假设其如果输出信号是 k(key)，如果 k 为高电平则表示汽车已经发动）。

假定警示灯的输入信号为 w(warn)，如果 w=1 则点亮警示灯。这样该数字系统的输入便是 s 和 k，输出则是 w。通过不断地检测传感器传过来的信号，选择是否点亮警示灯。如果 s=0,k=1 则 w 为 1。

案例 4：小键盘转换器（Keypad Converter）

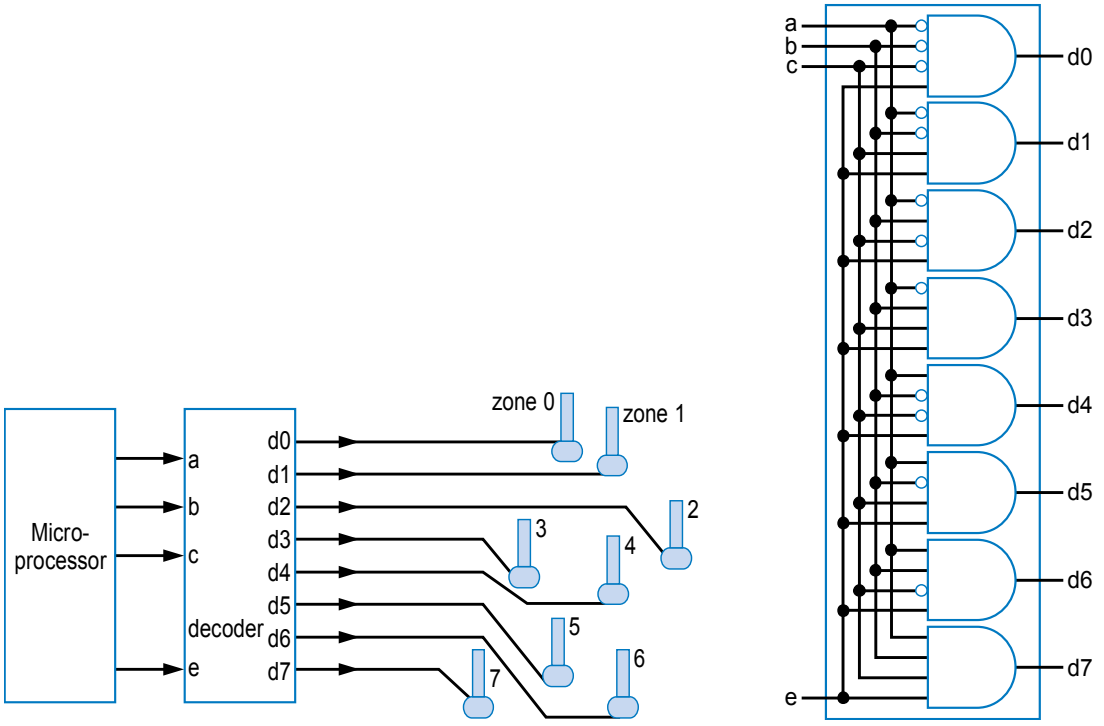
将键盘的 12 个按键转换成 4 位编码（4-bit code），如图 5.4 所示。其中 0-9 转换成 0000 to 1001，* 转换成 1010，# 转换成 1011，当什么按键也不按（nothing pressed）时，输出 1111。

读者可能在诸如电话，自动取款机等许多地方见过如图 5.4 的 12 按键键盘。第一行包含数字键“1”、“2”、“3”，第二行包含数字键“4”、“5”、“6”，第三行包含数字键“7”、“8”、“9”，第四行是混合键“*”、“0”、“#”。键盘的输出由 7 个信号来控制，每一行（row）对应一个输出端，每一列（column）对应一个输出端，其中 4 个信号(r1,r2,r3,r4)分别控制行输出，另外 3 个信号(c1,c2,c3)分别控制列输出。每当按下（press)一个按键时，便会有与之对应的行和列的输出变为高电平。例如当按下数字键“5”的时候，r2 和 c2 变为 1；而当按下符号键“#”的时候，r4 和 c3 变为 1。

图 5.4　8421BCD 码编码器

案例 5 全自动喷洒系统阀门控制器（Sprinkler valve controller）。

全自动草坪喷洒系统利用数字电路来控制其喷水阀门的关闭和开启。一个喷洒装置通常能够覆盖几个不同的区域，例如后面的草坪，左边的草坪，右边的草坪等等。该装置为了保证其管道内有足够的水压，某一时刻只能有一个方向的阀门被开启，本例假定该喷洒系统能够覆盖 8 个不同的区域（zone）。在通常情况下，该喷洒装置的控制电路由廉价的微处理器（Microprocessor 构成，该微处理器能够在一天的特定时间段内开启喷洒装置的每一个水阀。假定本例所使用的微处理器只有 4 个输出（outputs）引脚，而不是所要求的 8 个引脚。为了使其能够控制 8 个水阀，我们用微处理器的 1 个引脚来控制阀门的开启，其它 3 个引脚的 8 种组合来选择哪一个水阀被开启。这样我们就设计出了具有 4 个输入（e 使能端，a, b, c 三个选择端）和 8 个输出 （d0, d1, d2, d3, d4, d5, d6, d7）的组合逻辑电路。当使能端 e=1 时，电路将会按照输入，使相应的输出端口置为高电平 1，如图 5.5(a)所示。



（a）用译码器实现　　（b）用门电路实现

图 5.5　喷水器控制器电路

案例 6. 多路复用器实例

多路复用器(Multiplexer) 又称数据选择器，是继译码器以后，又一种用的比较频繁的电路模块（building block)。一个 Mx1 多路复用器有 M 路数据输入端，1 路数据输出端。某一时刻只允许 1 路输入数据从输出端口输出。 一个多路复用器就像如图 5.6 的铁路道岔一样，道岔主要是将多个铁轨中的某一条铁轨连接到另外一条铁轨输出。哪一条铁轨的火车可以通过另一条铁轨，取决于道岔控制器选择了哪一条铁轨连接到该铁轨。对一个多路复用器来说，控制开关并不是一跟杠杆，而是一个输入选择器，以二进制的形式选择要连接的输入。与某一条铁轨上的火车是否出现在输出的铁轨上不同，多路复用器输出 0 还是 1 取决于那路输入的数据是 0 还是 1。

多路复用器(Multiplexer)根据地址选择码从多路输入数据中选择一路到数据输出端输出，常用的多路复用器有二选一（如 74x157、74x257 等)、四选一（如 74x153、74x253 等)、八选一（如 74x151、74x251）和十六选一（如 74x150 等)。

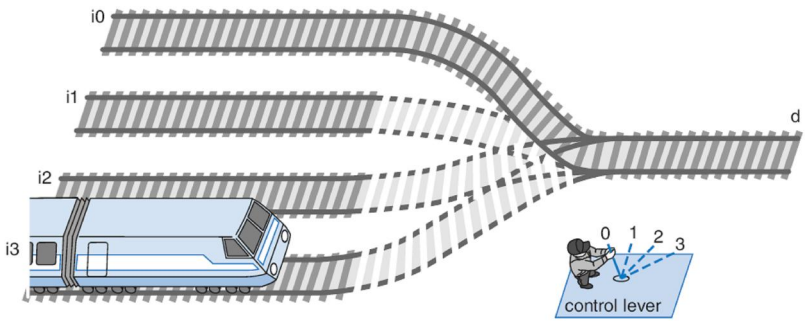


图 5.6　铁路站场上的控制开关

案例 7. 多路复用器构成的市长投票器

多路复用器(Multiplexer)的应用非常广泛。由多路复用器可以构成的市长投票器。假设你是某个小城市不怎么知名的一个市长。在市上每次开会时，市上的管理者们会给市长提出四条建议，市长会对这四条建议进行投票，从而决定采纳哪条建议。投票刚一结束，不论结果如何那些居民都会不约而同的向市长发出嘘的声音。大家可能会怀疑它滥用手中的权利，因此市长决定利用数字电路的知识制作一个简单的投票器。他找了四个能上下拨动的开关，这样便能输出高低电平。当市长在会议上对第一个建议进行投票的时候，他会把第一个开关向上拨动（表示赞成）或者是向下拨动（表示反对），不过没人知道开关所处的位置。当对第二个建议进行投票时，他会把第二个开关向上或向下拨动，接下来的投票以此类推。当投票结束以后，市长便会离开会议室回家。市长离开会议室以后，市上的管理者们便会将投票器的输出接到一个红/绿灯泡上。如果输入的低电平，则红灯亮起来，反之绿灯亮起来。那些管理者们通过两个选择开关来显示四个输入开关的状态。这些管理者们首先将选择开关拨动到 00（与此同时说出：“市长对这个建议的态度是……”），接着是 01，10,11，并根据输入开关的状态是红灯亮还是绿灯亮。该投票系统可以按图 5.7 很容易的制作出来，这个

电路可以用 4 选 1 的多路复用器实现。

市长的开关提议

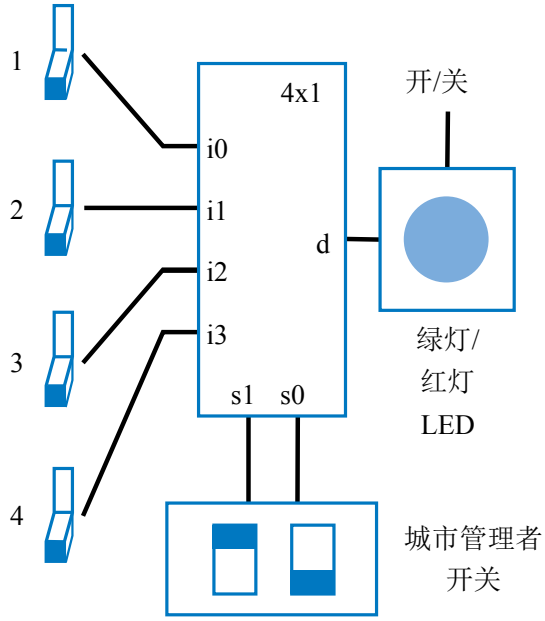


图 5.7　市长提议控制开关

案例 8 汽车显示控制开关

车载后视镜显示器的设计。一些汽车后视镜的正上方有一个显示器，驾驶员按一下按钮便能够让其显示室外温度，平均油耗比、当前的油耗比以及剩下的油所能跑的里程数。假定车载计算机向显示器传送的是 4 路 8 位数据，这四路数据分别为室外温度 T(Temperature)、平均油耗比 A(Average mile per gallon)、当前油耗比 I(Instantaneous mile per gallon)，以及剩余里程数 M(Miles remaining)， 如图 5.8 所示。我们可以设计其中的 T 由 8 位二进制数构成：t7,t6,t5,t4,t3, t2, t1,t0，同理 A，I 和 M 也包含 8 位数据。假定显示器有两个额外的输入端 x 和 y，无论按钮何时按下，这两个输入端都会按照 00，01，10，11 的规律进行变化。我们设定：当 xy=00 时，让 T 在显示器上显示；当 xy=01 时，数据 A 在显示器上显示；当 xy=10，I 在显示器上显示；当 xy=11 时，M 在显示器上显示。这里假定显示器的输出 D 具有将 8 位二进制数转换为人们可识别的十进制数的功能。

我们可以用 8 个 4 选 1 的多路复用器来设计该显示系统。

大家可能留意到将会有多根线从车载计算机引出来，最后可能有多达 8*4=32 根线连接到显示器上，这些线的数目确实很多。在随后的章节中，我们将会了解到如何减少这些线的数目。

从上面的设计中可以看出，合理地利用一些模块化的电路单元可以大大地降低电路的设计难度。如果使用常规的 4 选 1 多路选择器，那么需要 8 个这样的模块，并且需要很多线；然而如果使用门级电路来实现的话，那么将会使用多达 40 个这样的门电路。由此看来使用模块化的电路可以使我们的设计更加容易实现。

通过持续地按按钮选择两个输入 x 和 y 的取值，进而选择数据 D 的输出。

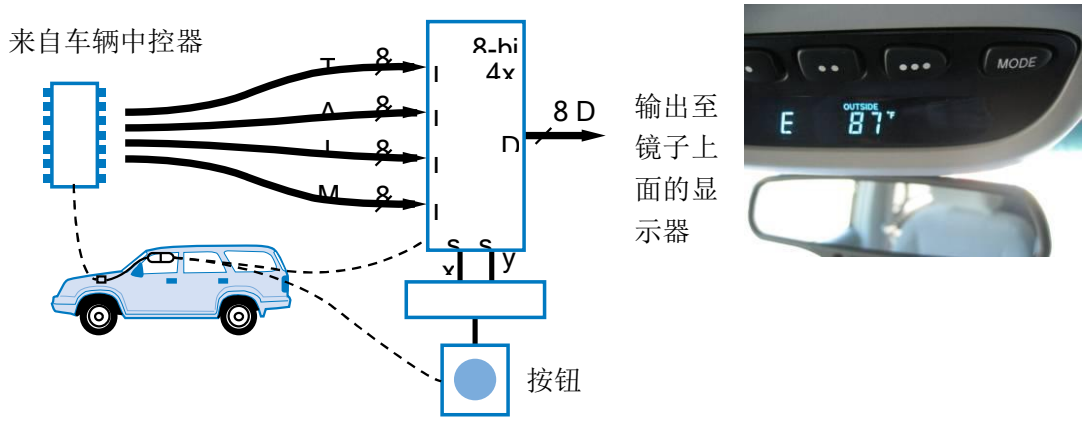


图 5.8　汽车显示控制开关

第六章　时序逻辑电路设计基础（Sequential Logic Design Foundations）

案例 1：SOS 检测器

摩尔斯电码(Morse alphabet)是美国人摩尔斯(Samuel Finley Breese Morse)于 1837 年发明的， 由点（.）、划（-）两种符号组成。一点为一基本信号单位，一划的长度等于 3 点的时间长度。在一个字母或数字内，各点、划之间的间隔为 1 点的时间长度。字母（数字）与字母（数字）之间的间隔为 3 点的时间长度的空格（Space）。SOS 是国际摩尔斯电码救难信号，编码表示为 3 点（S），Space，3 划（O），Space，3 点（第 2 个 S）。

现设计一个检测输入 SOS 的检测器。若一点用 1 个周期的高电平表示；一划用 3 个周期的高电平表示；在一个字母或数字内，各点、划之间的间隔用 1 个周期的低电平表示；连续出现两个周期的高电平或低电平视为不合理。则 1 个合理的 SOS 字符串是 10101000111011011100010101000.

我们设计 3 个有限状态机(Finite State Machine, FSM)用于检测点（Dot）、划（Dash）和空格（Space），然后这些 FSM 的输出连到检测 S 和 O 的状态机，最后设计一个 FSM 检测 SOS。实现该设计方案的原理图如图 6.1 所示。

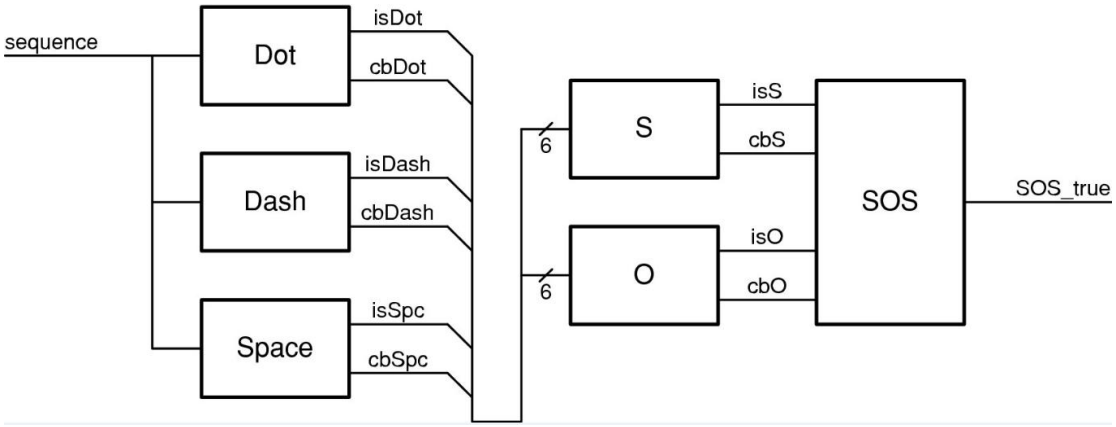


图 6.1 SOS 检测器原理图

输入序列（sequence）同时输入到 3 个 FSM（Dot, Dash, Space, 分别检测点、划和空格），这些 FSM 各有两个输出，一个输出表示检测到，另一个输出表示可能检测到，如：isDot 表示已检测到点（Dot），cbDot 表示可能（could be）检测到点（Dot），是否是点（Dot），则需进一步确认。

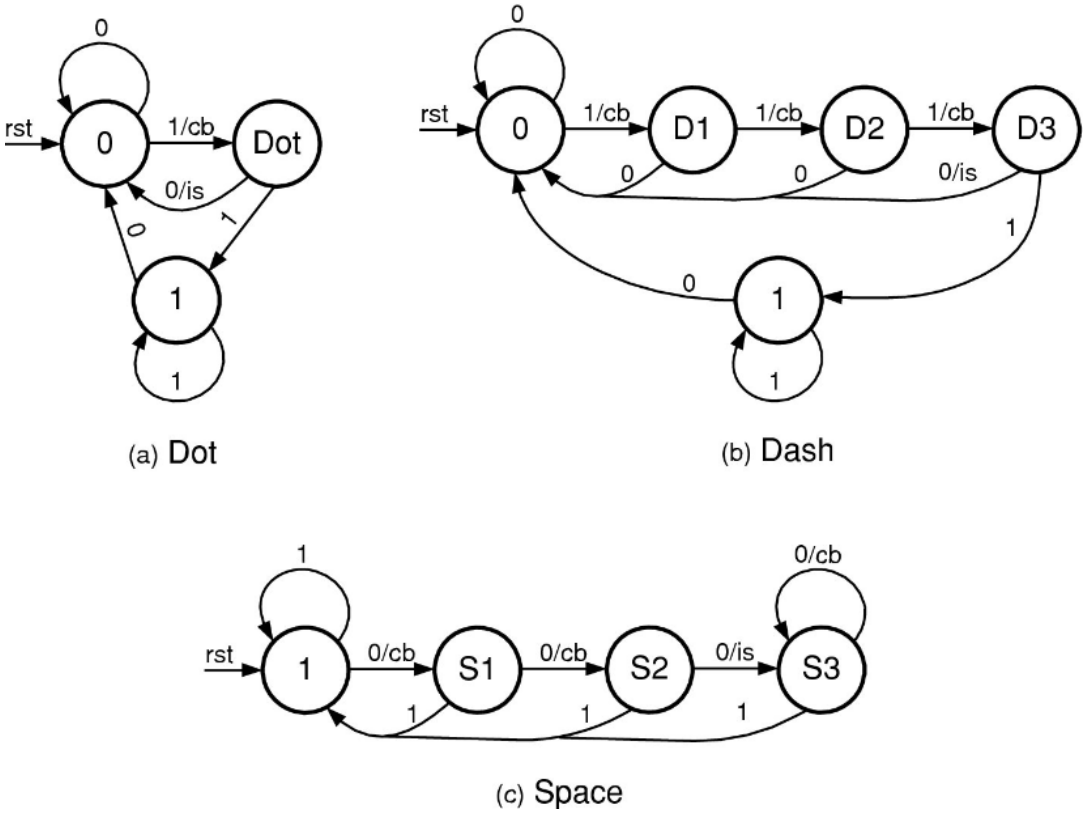


图 6.2 三个状态图分别检测点 Dot（图 a），划 Dash（图 b）和空格 Space（图 c）

图 6.2 是检测点、划和空格的 FSM，图中的 rst 表示复位信号，输出 cb 表示可能（could be）检测到，输出 is 表示已检测到。图 6.3 是检测 S 的状态图，图 6.4 是检测 SOS 的状态图。

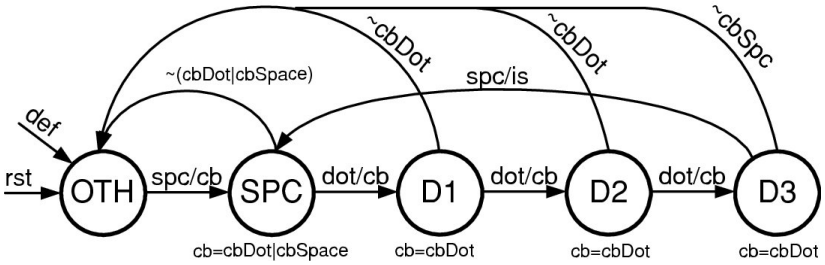


图 6.3 检测 S 的状态图

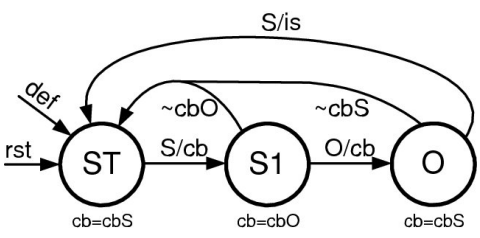


图 6.4 检测 SOS 的状态图

案例 2：交通灯控制器

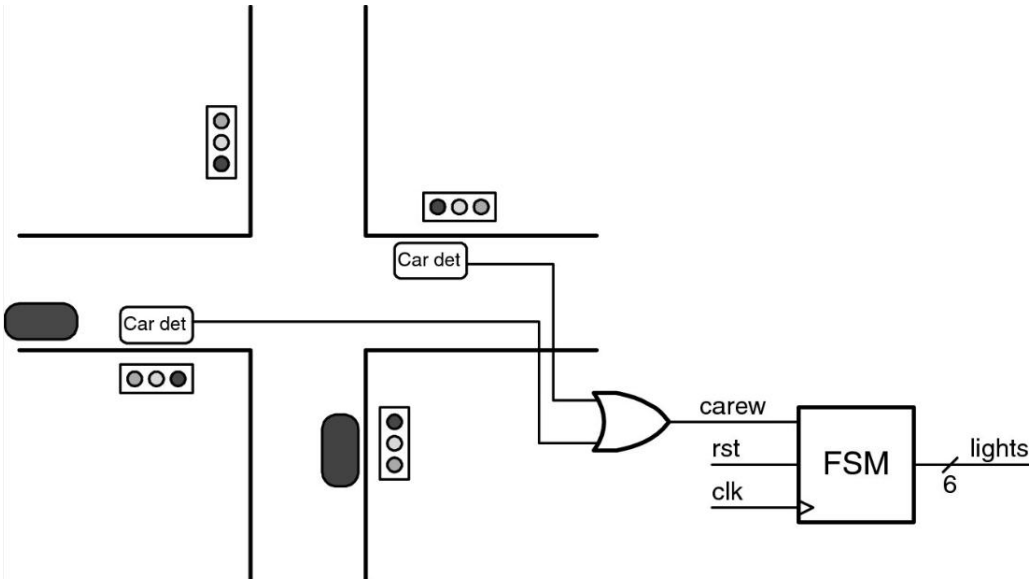


图 6.5 用 FSM 控制十字路口的交通灯

图 6.5 是一个用 FSM 控制十字路口交通灯的示意图，图中用 6 个灯来控制南北方向和东西方向的绿灯、黄灯和红灯；信号 carew 表示东西方向有车在等待，信号 rst 表示把 FSM 复位到一个已知的状态。

现要求：

用信号 rst 复位 FSM，使南北方向的灯是绿灯，东西方向的灯是红灯。

若检测到东西方向有汽车（carew=1），则下一个状态应置东西方向的灯为绿灯，经过预置的时间间隔后，南北方向的灯变为绿灯。

灯的变换顺序先是绿灯，其次黄灯，然后是红灯。

一个方向的灯变为红灯后，另一方向才变为绿灯。

满足上述要求的一个交通灯控制器的状态转换图如图 6.6 所示。

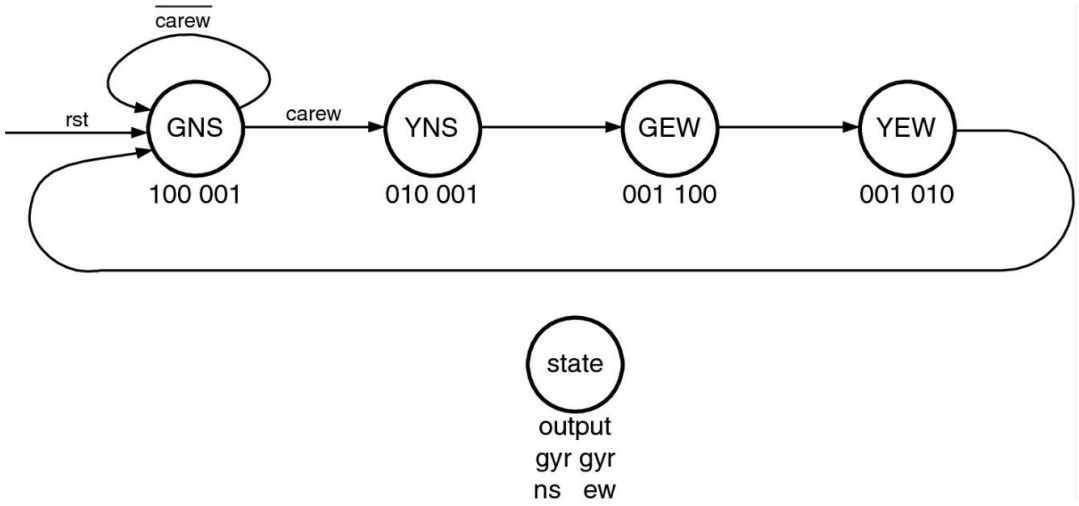


图 6.6 交通灯控制器的状态转换图

案例 3：自动售货机控制器

19 世纪 70 年代，自动售货机在美国、日本迅猛发展，如今已成为世界上最大的现金交

易市场。

自动售货机接受 3 种硬币 nickel，dime 和 quarter。当一个硬币投入到机器时，3 根线（各表示一种硬币）之一出现一个脉冲，表示收到 3 种硬币中的一种。物品的价格用 n 位二进制数存储在机器中（以 nickel 为单位）。

当投入的硬币价值足以购买一瓶软饮料时，信号 enough 有效。一旦 enough 有效，并且用户按了 dispense 按钮，信号 serve 就有效，这时状态机等待信号 done 有效，表示完成了一瓶软饮料的销售工作。然后若存在找零，则机器以 nickel 方式对用户进行找零。图 6.7 是自动售货机控制器的状态转换图，图 6.8 是自动售货机控制器的方框图。

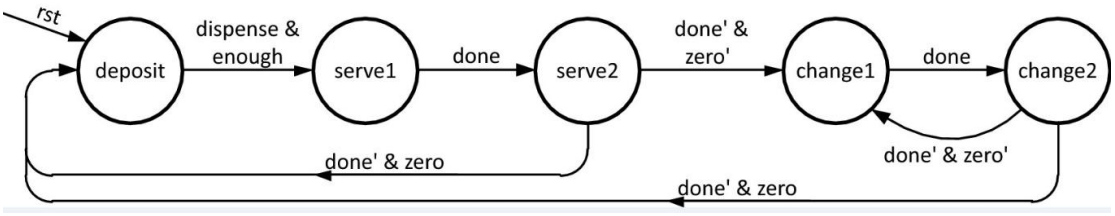


图6.7 自动售货机控制器的状态转换图

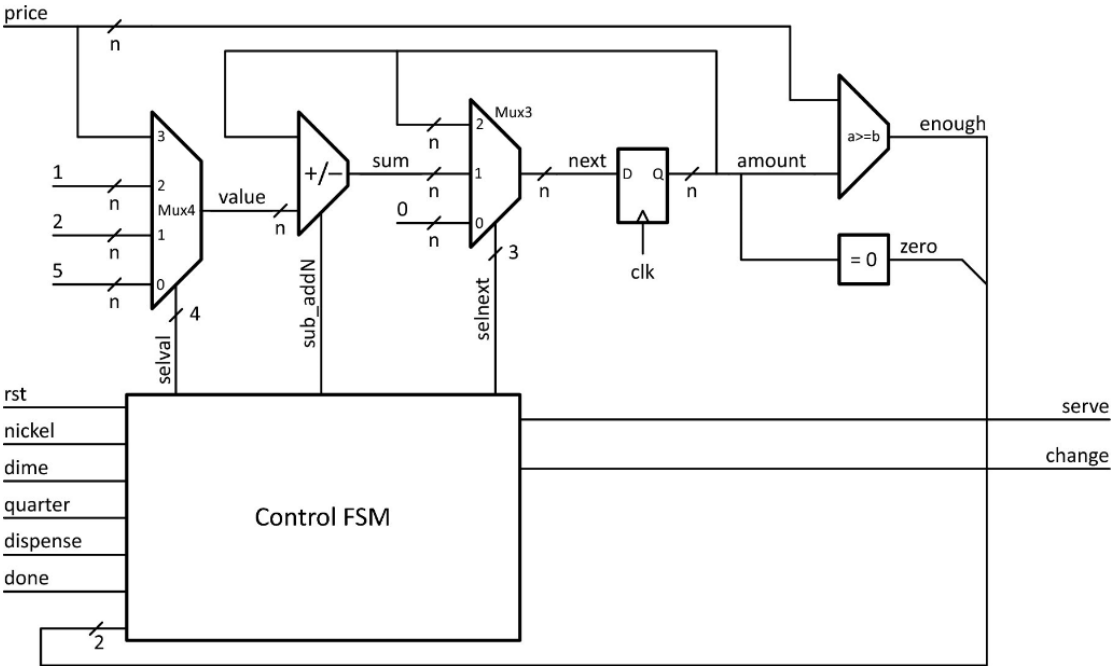


图6.8 自动售货机控制器的方框图

案例 4：数字密码锁

数字密码锁接收小键盘的输入，用户输入十进制数的密码序列，然后按 enter 键。若用户输入的密码序列正确，则 unlock 输出有效，表示门被打开。为了再上锁，用户按第 2 次 enter 按钮，若输入不正确的密码，则 busy 输出有效。图 6.9 是密码锁状态机的状态转换图，图 6.10 是密码锁控制器方框图。

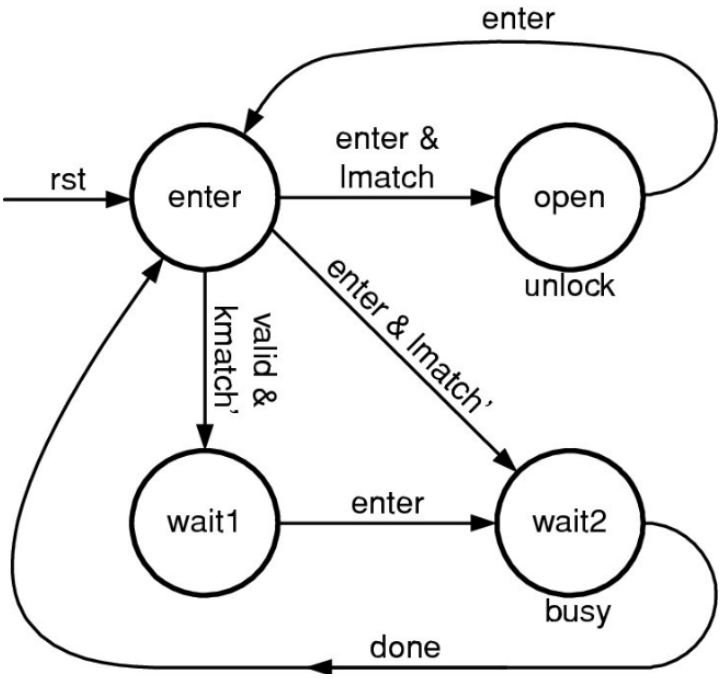


图 6.9 密码锁状态机的状态转换图

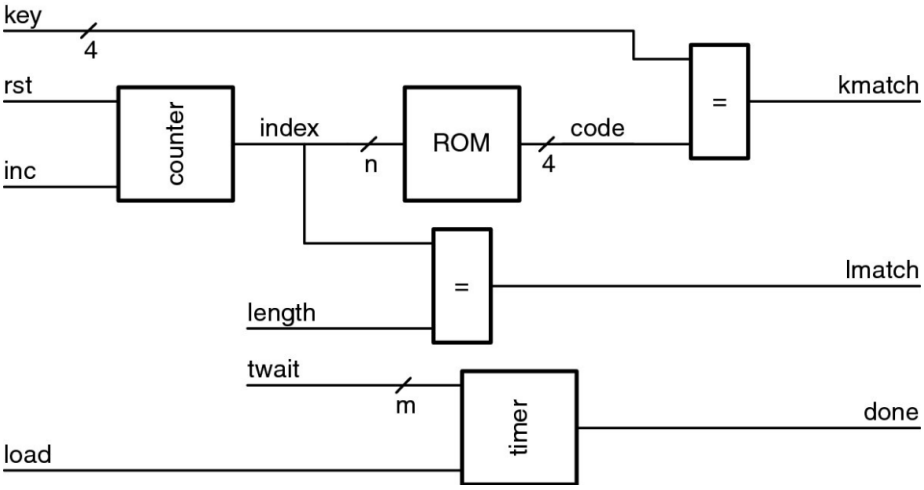


图 6.10 密码锁控制器方框图

案例 5：闪光灯控制器

闪光灯控制器有一个输入 in 和一个输出 out，当 in 出现一个周期的正脉冲时，触发一次闪光过程，在闪光过程中，输出 out 驱动 LED 灯的亮与灭。

闪光过程是：输出 out 持续 6 个周期的高电平（LED 灯亮），其次输出 out 持续 4 个周期的低电平（LED 灯灭），然后重复一次 LED 灯亮和灭，当第 3 次 LED 灯亮后，状态机返回到 OFF 状态，等待输入 in 的下一个正脉冲出现。图 6.11 是闪光灯控制器的状态转换图，图 6.12 是一个改进的闪光灯控制器的状态转换图。

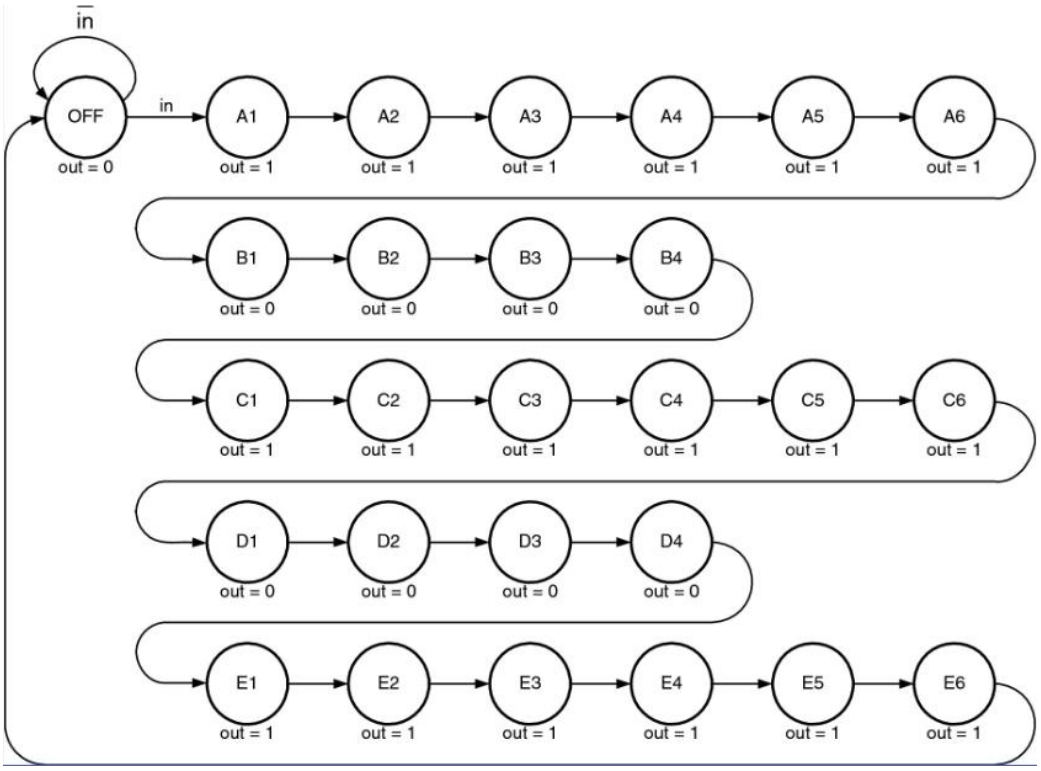


图 6.11 闪光灯控制器的状态转换图

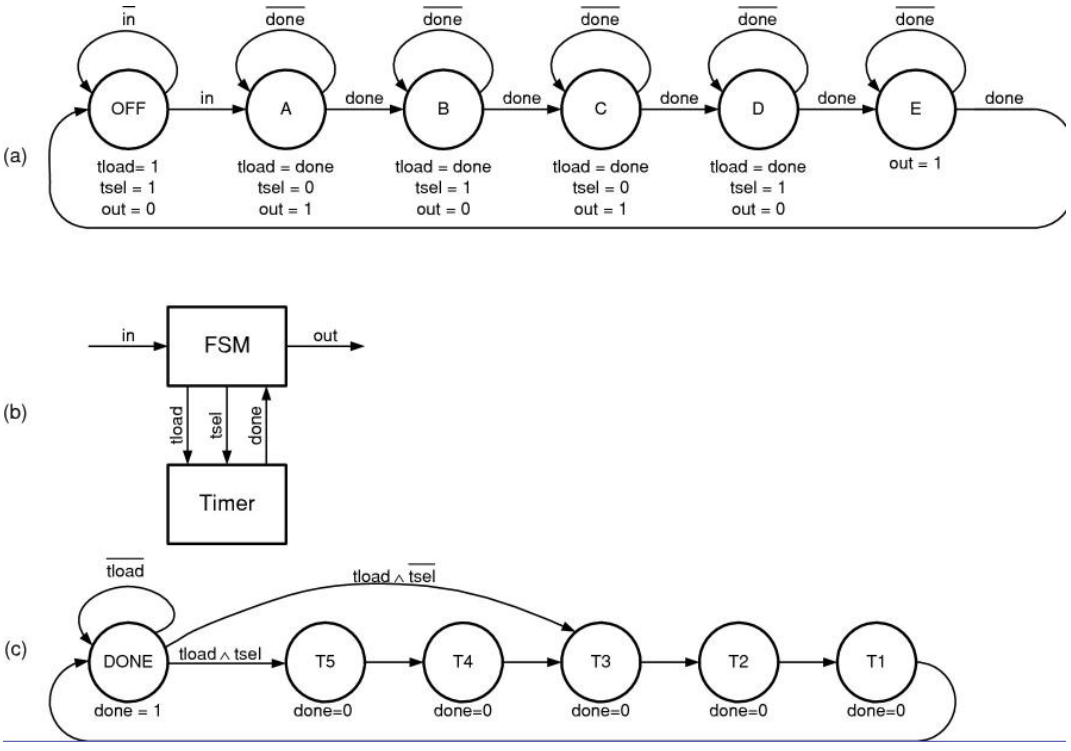


图 6.12 一个改进的闪光灯控制器的状态转换图

第七章　时序逻辑电路设计基础（Sequential Logic Design Foundations）

案例 1：可编程方波发生器

一个可编程方波发生器电路，根据变量 ON（即，逻辑“1”）和 OFF（即，逻辑“0”）间隔可以产生方波，如图 7.1 所示。间隔持续时间由 4 位控制信号控制，m 和 n，这被中断为无符号整数。ON 和 OFF 的间隔分别是 m* 100 和 n* 100 纳秒。设计必须完全同步。

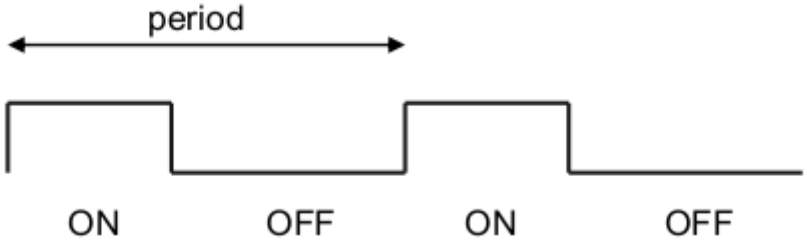


图 7.1 方波参数

案例 2：网络路由器

网络路由器的基本功能是去接收送来的包，读出包的目标地址，并发送包给相对应地址的其中一个端口，由包组成的数据就这样通过网络发送出去了。（例如：一份邮件或者一张图片）。包也能够组成地址，容错校验位和其他更多类型。该项目中，我们将采用一个高倍放大路由器，它的模块图如图 1 所示，该路由器有 24 位输入 I，组成了 3 个字节：字节 I（7..0）是包的目标地址，字节（15..8）是数据，字节 I（23..16）是总数校验。一个 1 位输入 IE 脉冲是当一个有效的包抵达目标即从 1 到一个时钟周期的时间。路由器有两个 24 位输出端口，即 P1 和 P2，每个端口和 P1E 和 P2E 相匹配，一个地址的包小于 128 时被分配给 P1 口，而一个有 128 的包或者有更大的包时应改分配给 P2。分配数据包给一个端口需要写满 24 位数据包到端口，并为相应的端口在一个时钟周期发送一次脉冲。

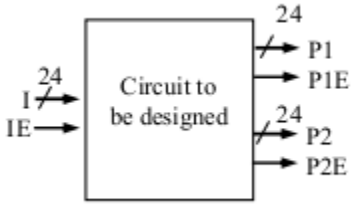


图 7.2 网络路由器模块

包的发送方按照其他数据包的双字节计算校验字节和。例如，如果一个数据包的目的地址是 15 和数据是 12，校验则是 27（当然是二进制）。如果路由器检测到一个数据包的校验和不正确，那么肯定有一个错误发生在数据包传输到路由器的途中；在这种情况下路由器将取消传输这个数据包。

图 2 提供了一个高级状态机描述路由器行为。路由器在 Waitpkt 状态等待数据包的到来表示为 IE= 1。然后路由器在寄存器保存包（因为包只能存在一个时钟周期在输入为 I 时），并在 Check1 状态计算校验和。在 Check2 状态，校验和被存储在寄存器中，如果 CS 不等于包总校验数字节（注：一个常见的错误将转到 Check1 状态，但这些转变将读取旧的 CS 值，在下一个时钟边沿没有新的值在该状态计算和更新。另一个常见的错误是当计算校验和时在 Check1 状态读 PKT，直到下一个时钟沿到来时 PKT 都不会更新）。如果校验和是正确的，那么路由器进入 Route 状态。如果地址少于 128 则从 Route 状态转换到 Route1 状态，否则转到 Route2 状态，这些状态写数据包到相应的端口，并设置相应的使能输出为 1。然后路由器回到等待其它一包状态。

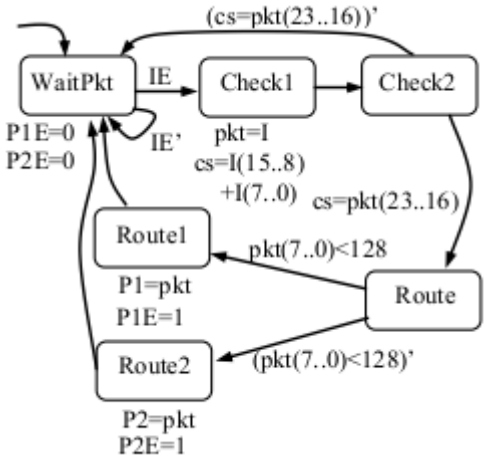


图 7.3 路由器的高级状态机描述

数字表示完成第一步的 RTL 电路设计过程。在这时模拟高级状态机。提供几个包，一些小于 128 的地址，和一些大于或等于 128，并注意包出现在适当的输出端口。包括一个数据包与一个不正确的校验和字节，并确保路由器不传输数据包。问题：这是路由器保证传输每一个数据包到达它的输入端吗？或可能路由器丢失（“下降”）一些数据包？如果数据包丢掉，什么情况造成这种丢失的，超过了其他错误校验吗？

第二步是创建一个数据通路。因此，创建一个路径能够实现数据的操作和条件在高级状态机图 2 中。提示：我需要记录 PKT 和 CS，一个加法器和比较器，所有我设计需要的。它是很好的对 P2 和 P1 输出实例寄存器的设计实践，P2 和 P1 也成为 P2 寄存器和 P1 寄存器。适当连接这些组件。确保名称唯一性以控制信号通路各部分。

现在，执行第三个 RTL 电路设计步骤连接上数据路径到控制器。确保所有的输入和输出已经连接上。下一步，执行控制器的有限状态机的第四个 RTL 电路设计步骤。记得，状态机应该有相同的状态和原始层次状态机图 2 的传输结构，但所有的数据操作和条件用布尔操作和条件取代，实现这些相同数据的操作和控制通路条件。

案例 3：激光测距仪

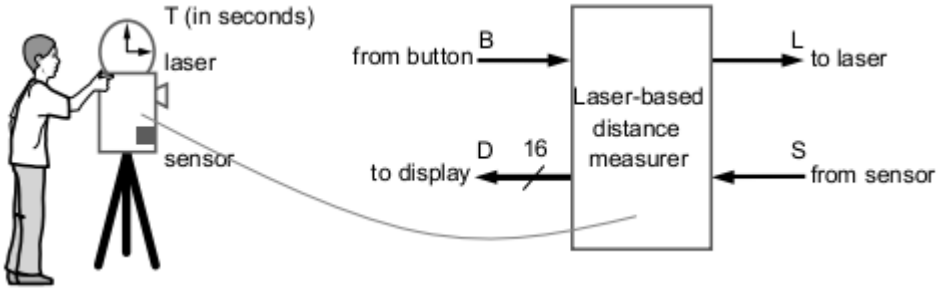


图 7.4 激光测距仪

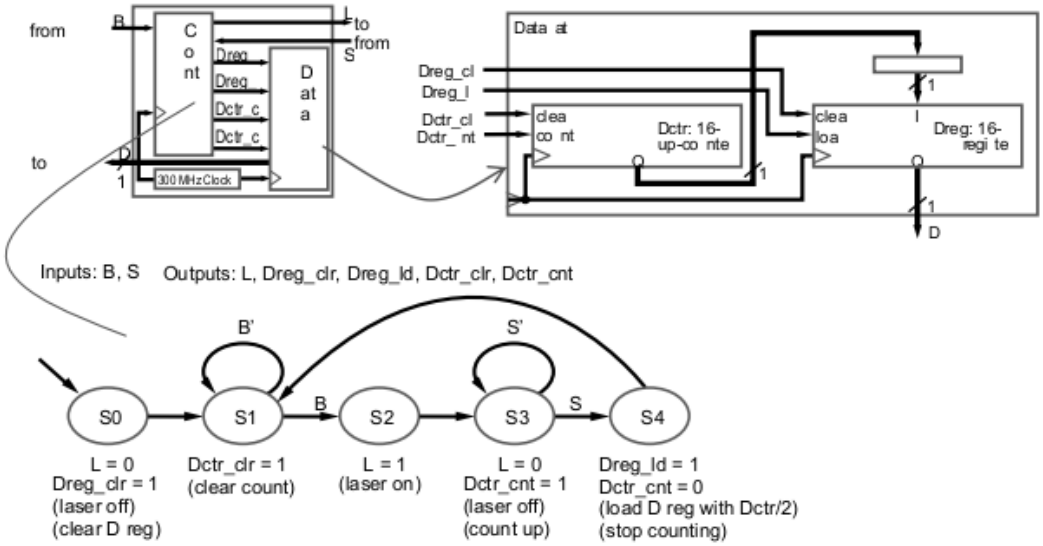


图 7.5 激光测距机和实现框图

输入与输出

- B:1 位输入，从按键到开始测量
- L:1 位输出，实现激光测距
- S:1 位输入，识别激光反应
- D:16 位输出，显示测量距离

案例 4：汽车转向灯控制

图 7.6 奥迪车模“四驱”跑车 2009 转向灯

奥迪已开发出了转向灯如图 7.6 所示。该项目的任务是制定一个低成本的转向灯，每一个转换信号包含 4 个 LED 灯。它的功能类似与福特雷鸟（T-bird）汽车 Wakerly 的转向灯。例如，当打开左转信号，左转信号表示方式为依次点亮 LED 向左边跑（1 发光二极管，2 个发光二极管，发光二极管，4……）和同样的右转向灯。实现有限状态图类似于一个文本显示方式在这里展示如下图 7.7，图 7.8

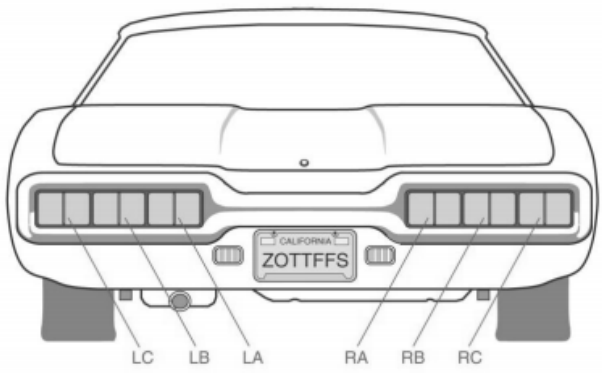


图 7.7　T-bird 尾灯命名

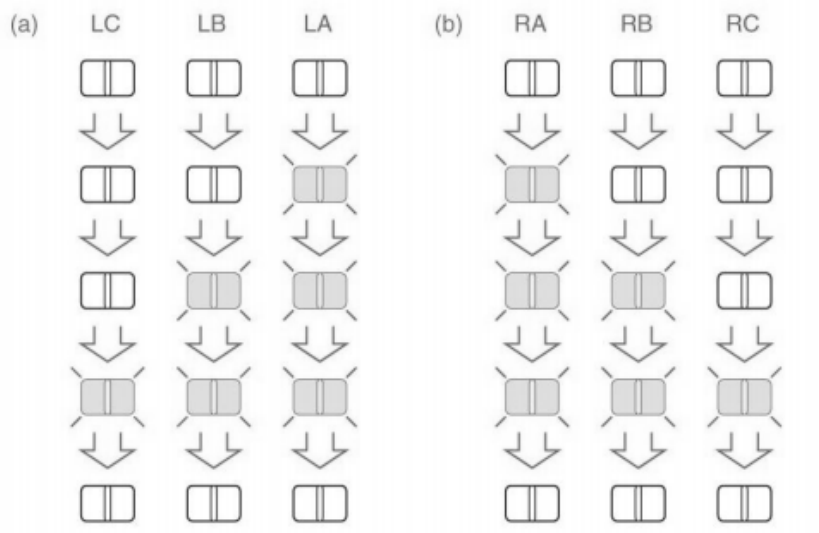


图 7.8　T-bird 尾灯转换序列

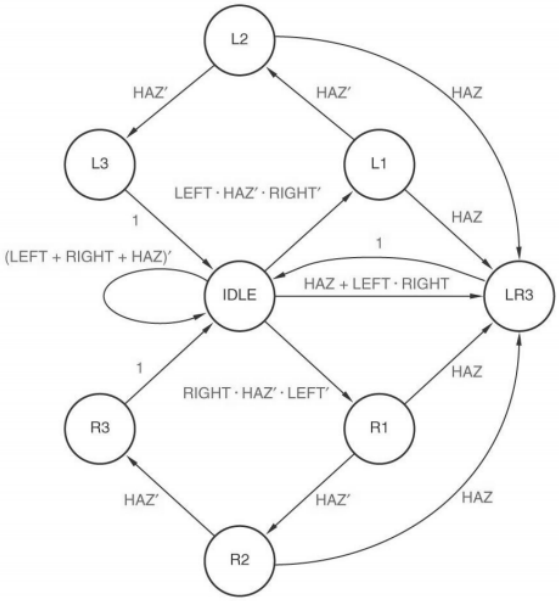


图 7.9　T-bird 尾灯状