**数字系统课程设计线上学习报告**



**课程名称： 数字系统课程设计（EDA与PLD）**

**任课教师： 万国春**

**姓 名:**

**学 号:**

**专 业： 微电子科学与工程**

目 录

数字系统课程设计线上学习报告 [1](#_Toc173424291)

[一、课程设计目的 3](#_Toc173424292)

[二、课程设计基本工具 3](#_Toc173424293)

[三、Vivado的学习和使用（基于资料学习） 4](#_Toc173424294)

[四、Verilog HDL的学习和使用（基于资料学习） 5](#_Toc173424295)

[①基本结构 5](#_Toc173424296)

[②数据类型及变量、常量 6](#_Toc173424297)

[③运算符 6](#_Toc173424298)

[④基本语句 7](#_Toc173424299)

[五、数字系统设计学习（基于视频学习） 8](#_Toc173424300)

[①绪论 8](#_Toc173424301)

[②数字逻辑结构 9](#_Toc173424302)

[③有限状态机 14](#_Toc173424303)

[④通用处理器 16](#_Toc173424304)

[⑤数字系统接口 19](#_Toc173424305)

[六、学习总结与心得 22](#_Toc173424306)

# 一、课程设计目的

通过企业课程设计实践资料以及企业线上课程，掌握Vivado数字系统设计工具，主要是进行针对FPGA设计的学习，并利用硬件描述语言学习资料，对Verilog HDL语言进行学习，主要是针对Verolog HDL的基本结构、数据类型及变量、常量、运算符以及语句结构进行学习，并完成企业线上数字设计任务。

另外通过浙江大学《玩转数字设计》线上学习课程，对数字信号、数字逻辑结构、有限状态机、通用处理器、数字系统接口等五部分进行学习，并将理论知识应用到具体实践当中，系统化学会一个数字系统设计的一套流程理念和方法。

# 二、课程设计基本工具

本次课程设计主要使用的基本工具为Vivado，主要使用的编程语言为Verilog HDL，以下是简要介绍：

Vivado是Xilinx公司开发的一款集成设计环境（IDE），专为其FPGA（现场可编程门阵列）和SoC（片上系统）产品提供全面的设计和验证工具。Vivado支持多种硬件描述语言（如VHDL、Verilog和SystemVerilog），并提供图形用户界面（GUI）和IP Integrator，方便设计人员通过图形方式或集成现成IP核快速构建复杂系统。

在设计流程中，Vivado的综合工具将HDL代码转换为门级网表，并通过布局布线（Place and Route）等步骤将设计映射到具体的FPGA资源上，生成比特流文件。同时，Vivado的仿真工具支持行为仿真和时序仿真，确保设计在功能和时序上的正确性。

Vivado集成了多种调试工具，如集成逻辑分析仪（ILA）和虚拟输入输出（VIO），并通过静态时序分析（STA）验证设计的时序约束。此外，Vivado HLS（高层次综合）允许设计人员使用C/C++等高级语言进行设计，加速开发过程。SDx工具则面向嵌入式系统和加速器应用，支持软件工程师快速开发硬件加速应用。

Vivado还提供了详细的用户手册、教程和应用笔记，帮助用户快速上手并深入了解工具的使用。Xilinx社区和论坛也是用户交流和解决问题的重要平台。综上所述，Vivado是一个功能丰富且灵活的设计工具，广泛应用于通信、消费电子、工业控制和汽车电子等多个领域。

Verilog HDL（硬件描述语言）是一种用于描述数字系统和电路的硬件描述语言，广泛应用于集成电路（IC）和现场可编程门阵列（FPGA）的设计。Verilog由Gateway Design Automation公司于1984年开发，后来被IEEE（美国电气电子工程师学会）标准化为IEEE 1364标准。它通过类似编程语言的语法，提供了一种便捷的方式来描述和模拟硬件行为。

Verilog支持多种抽象层次的设计描述，包括行为级、RTL（寄存器传输级）和门级描述。这使得设计人员可以在较高的抽象层次上进行设计，然后逐步细化到实现层次。Verilog的语法类似于C语言，易于学习和使用，这也是它被广泛采用的一个重要原因。

仿真是Verilog的一大优势，通过编写测试平台，设计人员可以在模拟环境中验证设计的功能和时序行为，提前发现并修正问题。Verilog仿真器可以对设计进行功能仿真和时序仿真，确保设计在实际硬件实现之前就能满足预期的性能指标。

Verilog还支持模块化设计，通过模块和实例化机制，设计人员可以构建复杂的系统，并重复使用已经验证的模块。模块化设计不仅提高了设计效率，还增强了设计的可维护性和可扩展性。

Verilog HDL在EDA（电子设计自动化）工具中得到了广泛支持，如综合工具、仿真器和调试工具等。这些工具帮助设计人员将Verilog代码转换为物理硬件，实现从设计到生产的无缝过渡。由于其强大的功能和广泛的支持，Verilog HDL已成为数字电路设计领域的重要工具。

# 三、Vivado的学习和使用（基于资料学习）

Vivado在Xilinx官网进行下载，如图1所示为Vivado安装界面。



图1 Vivado安装界面

图2所示为Viv阿斗进入后的主界面，主要由四个部分组成，分别是——“Quick Start”，用于打开例程；“Task”，用于IP管理等；左侧用于打开最近使用的工程；下方为TCL控制台。

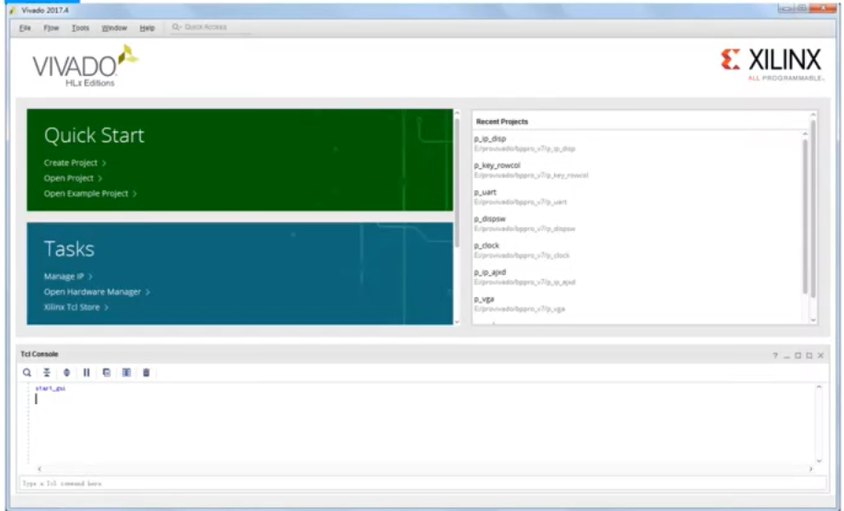


图2 Vivado主界面

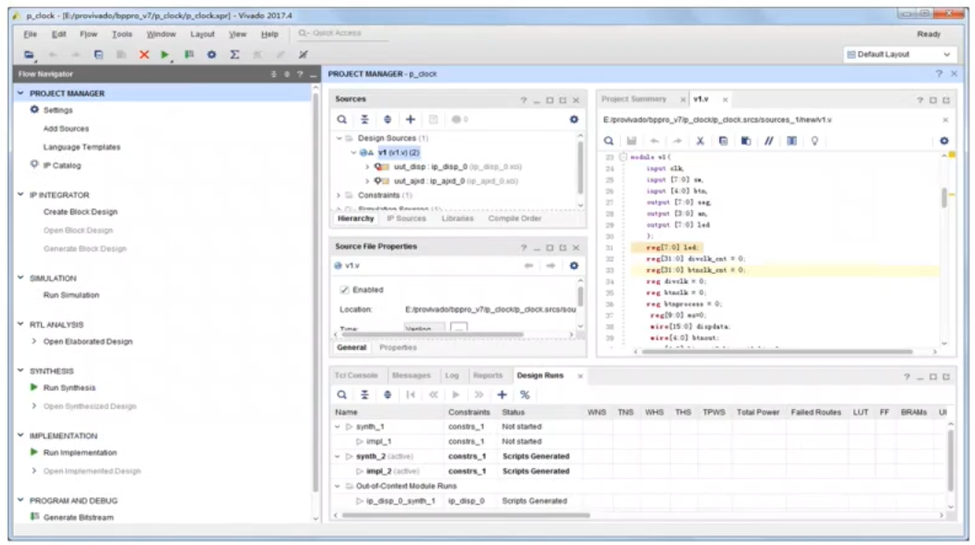


图3 Vivado工程界面

如图3所示为Vivado的项目工程界面，最左侧为流程导航板块，可以分为设置、仿真、RTL分析、综合、实现、编程和调试；在右侧代码区中，上方左侧为工程管理板块，主要是工程源文件；右侧为主页框，用于显示代码以及仿真波形；下方为控制台、日志和消息区，用于监视项目过程中的信息。

# 四、Verilog HDL的学习和使用（基于资料学习）

## ①基本结构

Verilog HDL程序是由模块构成的。每个模块嵌套在module和endmodule声明语句中;每个Verilog HDL源文件中只有一个顶层模块，其他为子模块。可以每个模块写一个文件;每个模块要进行端口定义，并说明输入输出端口，然后对模块的功能进行行为逻辑描述;模块中的时序逻辑部分在always块的内部，在always块中只能对寄存器变量赋值；模块中对端口或其他wire型变量的赋值，必须在always块的外部使用assign语句，通常是将寄存器的值送出；程序书写格式自由，一行可以写几个语句，一个语句也可以分多行写；除了endmodule语句、begin end语句和fork join语句外，每个语句和数据定义的最后必须有分号；可用/\*...\*/和//...对程序的任何部分作注释。加上必要的注释，可以增强程序的可读性和可维护性。如图4所示为一个module示意图。



图4 module示意图

## ②数据类型及变量、常量

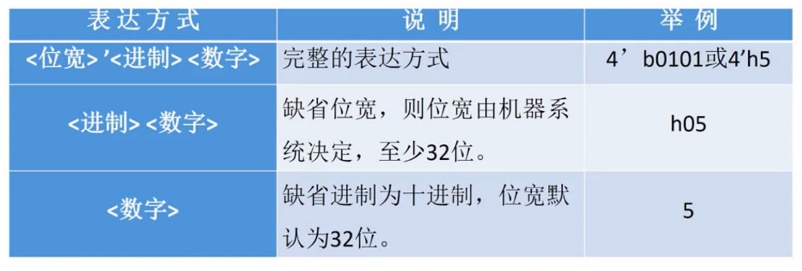
Verilog HDL有两种常用的数据类型，线网(Net)类型及变量类型。常量的值是不能够被改变的，变量的值是可以被改变的，例如寄存器型的变量reg；从逻辑值和常量式开始，对于线网类型重点是常用的wire型变量，对于变量类型重点是reg型。图5所示分别为Verilog HDL中的逻辑值以及数据表达方式。

图5 Verilog中数据的逻辑值以及表达方式

## ③运算符

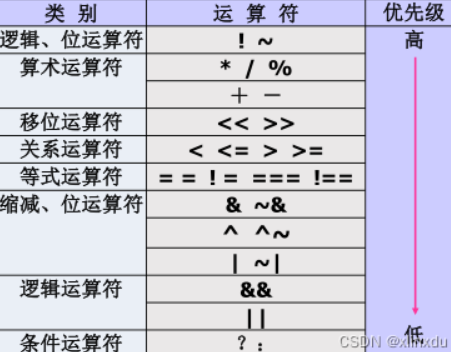


图6 Verilog中运算符

如图6所示为Verilog HDL语言中的运算符，运算符按功能分为9类：算术运算符、逻辑运算符、关系运算符、等式运算符、缩减运算符、条件运算符、位运算符、移位运算符、位拼接运算符；运算符按操作数的个数分为3类：单目运算符–带一个操作数：逻辑非！、按位取反~、缩减运算符、移位运算符；双目运算符–带两个操作数：算术、关系、等式运算符；逻辑、位运算符的大部分；三目运算符–带三个操作数：条件运算符。

对于算术运算符，进行整数除法运算时，结果值略去小数部分，只取整数部分；%称为求模（或求余）运算符，要求%两侧均为整型数据；求模运算结果值的符号位取第一个操作数的符号位！，例-11%3，结果为-2；进行算术运算时，若某操作数为不定值x，则整个结果也为x。

对于逻辑运算符，非零的操作数被认为是真(1‘b1)；零被认为是假(1‘b0)；不确定的操作数如4’bxx00,被认为是不确定的(可能为零，也可能为非零）（记为1’bx)；但4’bxx11被认为是真（记为1’b1，因为它肯定是非零的）；注意进行逻辑运算后的结果为布尔值（为1或0或x）；“&&”和“||”的优先级除高于条件运算符外，低于关系运算符、等式运算符等几乎所有运算符；逻辑非“！”优先级最高。

对于位运算符，位运算其结果与操作数位数相同。位运算符中的双目运算符要求对两个操作数的相应位逐位进行运算；两个不同长度的操作数进行位运算时，将自动按右端对齐，位数少的操作数会在高位用0补齐。

对于关系运算符（双目），运算结果为1位的逻辑值1或0或x。关系运算时，若关系为真，则返回值为1；若声明的关系为假，则返回值为0；若某操作数为不定值x，则返回值为x；所有的关系运算符优先级别相同；关系运算符的优先级低于算术运算符。

对于等式运算符（双目），运算结果为1位的逻辑值1或0或x；等于运算符(= =)和全等运算符(= = =)的区别；使用等于运算符时，两个操作数必须逐位相等,结果才为1；若某些位为x或z，则结果为x；使用全等运算符时，若两个操作数的相应位完全一致（如同是1，或同是0，或同是x，或同是z）,则结果为1；否则为0；所有的等式运算符优先级别相同。

对于缩减运算符，运算法则与位运算符类似，但运算过程不同；对单个操作数进行递推运算,即先将操作数的最低位与第二位进行与、或、非运算，再将运算结果与第三位进行相同的运算，依次类推，直至最高位；运算结果缩减为1位二进制数。

对于移位运算符，其用法是：A>>n 或 A<<n，将操作数右移或左移n位，同时用n个0填补移出的空位。

条件运算符（三目）的格式为：信号 = 条件？表达式1：表达式2。当条件为真，信号取表达式1的值；为假，则取表达式2的值。

位拼接运算符位拼接运算符为{ }，用于将两个或多个信号的某些位拼接起来，表示一个整体信号。

## ④基本语句

Verilog HDL语句包括过程语句、块语句、赋值语句、条件语句、循环语句、编译导向语句等。

Verilog 中的过程语句包括 `initial` 和 `always`。`initial` 语句在仿真中初始化，语句块中的内容仅执行一次；而 `always` 语句则会不断重复执行，通常用于描述连续行为。每个 `initial` 和 `always` 语句块都是独立并行执行的，执行顺序与书写顺序无关，且在仿真时间 0 时刻同时开始。注意，`initial` 和 `always` 语句不能嵌套使用。

块语句有顺序块（`begin-end`）和并行块（`fork-join`）。顺序块用于顺序执行一系列语句，格式为 `begin ... end`。并行块用于并行执行多条语句，格式为 `fork ... join`。并行块中的语句会同时开始执行，如果它们同时对同一变量产生影响，可能会导致竞争问题。

赋值语句在 Verilog 中主要用于连续赋值和阻塞赋值。连续赋值使用 `assign` 关键字，通常用于线网类型变量，输出值随输入值变化而变化。阻塞赋值使用 `=`，用于过程块中的赋值。连续赋值可以在声明线网变量时直接赋值，也可以在赋值语句中隐含声明线网变量。

条件语句包括 `if-else` 和 `case` 语句。`if-else` 用于根据逻辑表达式的值执行不同的操作，逻辑表达式值为 0、x、z 时为假，值为 1 时为真。`if-else` 语句可以嵌套多个操作语句，有优先级顺序。`case` 语句用于多分支选择，没有优先级顺序，各分支并行，且表达式的位宽必须相等。

Verilog 提供四种循环语句：`forever`、`repeat`、`while` 和 `for`。`forever` 用于生成周期性波形，通常在 `initial` 中使用，表示永久循环。`repeat` 按固定次数执行循环。`while` 根据条件表达式执行循环。`for` 循环使用初始化表达式、条件表达式和迭代表达式控制循环执行。

编译导向语句用于控制 Verilog 编译器的行为，例如 `define` 用于宏定义，`include` 用于包含其他文件，`ifdef` 和 `ifndef` 用于条件编译。这些语句帮助在编译过程中控制代码的编译和包含。

# 五、数字系统设计学习（基于视频学习）

## ①绪论

本课的绪论部分主要介绍了数字系统的基本概念，即数字信号与数字系统概述与实现。

数字信号是相对于模拟信号的概念。模拟信号可以取任意的连续变化的值，而数字信号只会取有限的可能的一个集合中的值。世界上的绝大部分是模拟信号，但数字信号正在成为趋势，如图7所示为音乐光盘波形转化成数字。

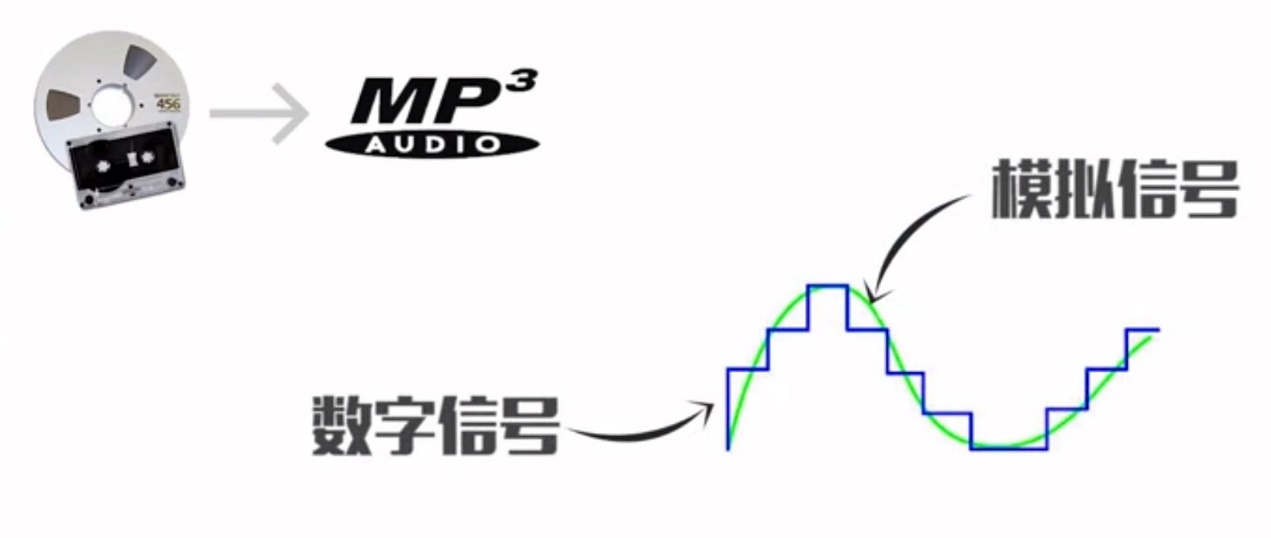


图7 音乐光盘波形转化成数字

实际上，经过采样和量化，大部分的模拟信号都可以被转化成数字信号。一般地，模拟信号更加真实的反映了原始的信号，数字信号则会丢失一些信息。但是，模拟信号在处理连续信号时器件会产生噪音，如热噪声；而对于数字信号，当误差在一定范围内是，噪声可以被忽略，因此数字信号正在成为趋势。

数字信号最基本的逻辑值为0和1，当使用二进制表示模拟信号时，数字信号就可以被模拟信号表征。在理想情况下，0电平和1电平之间可以实现瞬间跳变，但实际上从0跳变到1或从1跳变到0需要有延时，污染延时则被定义为信号变化后，经过多长时间输出才会发生变化，对于污染延时，寄存器设计时需要考虑，组合电路设计时不需要考虑。

目前数字系统已经得到很大的普及，对于所有数字系统，都可以看作是一个处理器外部交互的接口，而数字系统的核心则包括——专用微处理器，只能做一种处理任务；通用微处理器，可以完成不同的任务，数字系统则是微处理器与外部接口的协同。

微处理器是一种电子数字的逻辑电路，也可以称为集成电路或者IC，它的最底层物理特性为高电平和低电平，即1和0，也对应于有电和没电，用0和1来表示逻辑，利用晶体管的特性，即可构成不同的与或非门，以实现不同电路功能。

利用门电路可以构成基本的组合电路和时序电路。组合电路是没有记忆功能的电路，它的输出只和当前时刻的输入有关，和以前的输入无关。而时序电路则是电路可以处于不同的状态，然后在不同状态之间发生跳转，它是有记忆功能的电路。而微处理器则是这两者的综合利用。

FPGA是数字系统中很重要的一个部分，即现场可编程逻辑阵列，它具有可配置的逻辑块和可配置的连接，使得其能够实现可编程的大规模数字逻辑功能。微处理器做ASIC专用集成电路流片之前，都会把这些设计放到FPGA上面做验证。

数字系统中其他三个重要概念是DSP、MCU、ASIC。DSP即Digital signal processor数字信号处理器，通过硬连线一些基本的常用的数字信号处理单元来实现功能，目前它的应用范围在萎缩，这是由于他的程序的执行和一般的通用微处理器一样，并且FPGA以及一些通用处理器也挤压了DSP的市场空间；MCU即微控制器，利用低成本的微处理器加上一些常用的外设以及存储，组成微控制器，它的好处是编程容易、价格便宜，但功能上比FPGA弱；ASIC即专用集成电路，是专门为了某个功能定制的专用芯片，流片前要放在FPGA上验证。

数字系统的实现还得益于硬件描述语言，它可以实现组合逻辑与时序逻辑，还可以结合使用，使用其对FPGA的逻辑进行配置，则可以实现数字系统。

## ②数字逻辑结构

本章讲述的主要内容为数字逻辑基础即数字逻辑运算的一般法则、数字逻辑化简方法、几种典型的组合逻辑电路、竞争冒险、组合逻辑分析与设计、几种典型的时序逻辑电路以及时序逻辑分析

数字逻辑基础首先讲明了数字逻辑的五个公理，即身份律，即A与0相加等于A，A与1相乘也等于A；零律和单位律，即A与1相加等于1，A与0相乘等于0；幂等律，即A与自身相加或相乘都等于A；互补律，即A与其补A'相加等于1，而A与其补相乘等于0；以及双重否定律，即一个变量的补的补等于该变量自身。此外还介绍了单变量定理、二变量定理和三变量定理，单变量定理包括：身份律，即A与0相加等于A，A与1相乘也等于A；零律和单位律，即A与1相加等于1，A与0相乘等于0；幂等律，即A与自身相加或相乘都等于A；互补律，即A与其补A'相加等于1，而A与其补相乘等于0；以及双重否定律，即一个变量的补的补等于该变量自身。二变量定理包括：交换律，即A与B相加等于B与A相加，A与B相乘等于B与A相乘；结合律，即(A与B)与C相加等于A与(B与C)相加，(A与B)与C相乘等于A与(B与C)相乘；分配律，即A与(B与C)相加等于(A与B)与(A与C)相加，A与(B与C)相乘等于(A与B)与(A与C)相乘。三变量定理包括：结合律在三个变量之间的应用，即A与(B与C)相加等于(A与B)与C相加，A与(B与C)相乘等于(A与B)与C相乘；以及分配律在三个变量之间的应用，即A与(B与(C与D))相加等于(A与B)与(C与D)相加，A与(B与(C与D))相乘等于(A与B)与(C与D)相乘。

数字逻辑的化简方法有公式法化简，包括并项法、吸收法、消项法、消因子法、配项法，此外还有卡诺图法化简，一个n输入逻辑函数的卡诺图是个含有2n个单元的矩阵图，每个单元代表一个可能存在的最小项。其化简的步骤为：首先，将要化简的逻辑表达式转换为最小项或最大项的和，并在卡诺图上标出相应的1（对于最小项）或0（对于最大项）。接着，将相邻的1（或0）组合成最大的矩形区域，每个区域的大小为2的幂次（如1、2、4、8等）。然后，从卡诺图中提取每个矩形区域对应的简化表达式，确保覆盖所有的1（或0）。最后，将这些简化表达式组合成最终的化简结果。

本章讲述的组合逻辑电路主要有编码器电路、译码器电路、比较器电路、加法器电路和数据选择器电路。编码器电路的作用是将某一特定的逻辑信号转换为二进制代码，图8为一个编码器电路。此外，有一类特殊的编码器电路称之为优先编码器，它可以同时输出两个以上信号，并按优先级输出，如图9所示为集成优先编码器举例——74148（8线-3线）

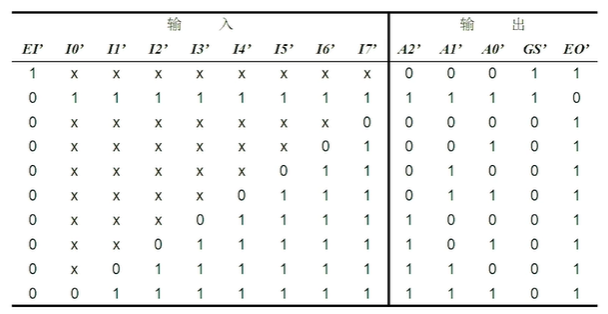


图9 集成优先编码器举例——74148（8线-3线）

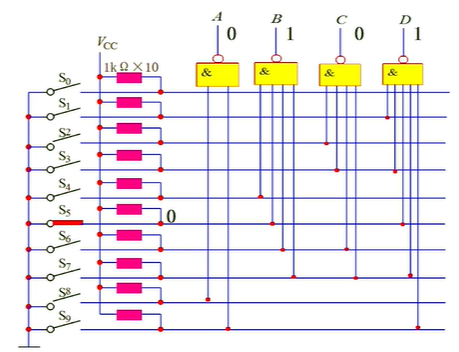


图8 编码器电路

译码器的作用是将输入代码转换成特定的输出信号，如图10所示为74x138 3-8译码器真值表，此外，还有4-16线译码器，还可以在其基础上扩展为5-32线译码器，具体电路如图11所示。在此基础上还有许多类型的数字显示译码器。



图10 74x138 3-8译码器真值表

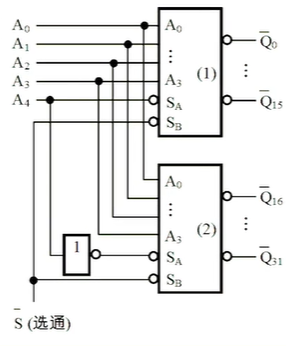


图11 将4-16线扩展为5-32线译码器

数值比较器用来比较两个位数相同的二进制数的大小，如图12所示为2位数值比较器的真值表。

加法器用来实现两个二进制数的加法运算，可以分为半加器与全加器。半加器只能进行本位加数、被加数的加法运算，而不考虑低位进位，而全加器则考虑低位进位。在此基础上还衍生出全减器，即能同时进行本位数和相邻低位的错位信号的减法运算器件，如图13所示为全加器和全减器的逻辑表达式。

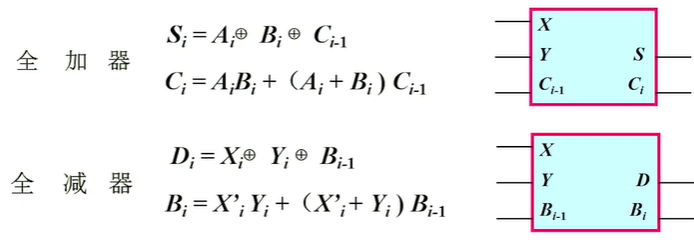


图13 全加器和全减器的逻辑表达式

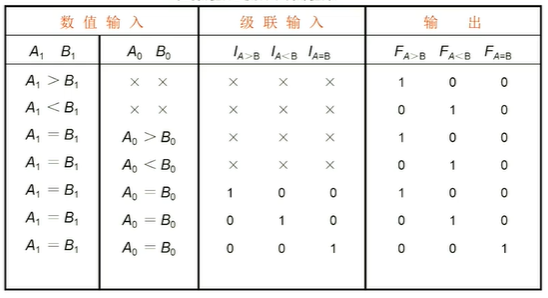


图12 2位数值比较器的真值表

数据选择器是通过地址选择码在n路输入数据中选择一路进行输出，输出数据编码时习惯从高位到低位，从左到右。

组合逻辑电路中竞争冒险的产生主要是由于输入信号同时变化引起的，这些信号通过不同的逻辑门和路径传播到输出端，导致输出信号在短时间内出现不确定状态。存在“0”冒险和“1”冒险。

组合逻辑电路的分析方法有以下几点：一、首先画出逻辑图；二、由逻辑图逐级写出表达式；三、化简与变换；四、由表达式列出真值表；五、分析逻辑功能。

组合逻辑电路设计方法为：一、逻辑抽象；二、写出真值表，给出逻辑函数式；三、化简为适当形式；四、根据化简式画出逻辑电路图；五、工艺设计。

本章对于时序逻辑电路主要讲述了锁存器、边沿触发器和寄存器，另外还讲到了时序逻辑电路分析设计方法。

锁存器是具有记忆功能的基本逻辑单元，能够自行保持状态，有两个稳定的状态，能够表示0、1逻辑或二进制数码0、1，在触发信号时改变输出状态，保持或取反。锁存器和触发器两者都是时序电路的基本构件，区别是锁存器不断监测输入，并独立于时钟信号，且在任何时刻都可以改变输出；触发器不断监测输入，但只在时钟信号所确定的时刻(非时间段)改变输出。，典型的锁存器有SR锁存器。

在锁存器基础上，即开发出主从式SR触发器、边沿触发D触发器，而边沿D触发器主要利用了CMOS传输门，如图14所示。边沿触发的原因是为了提高触发器的工作可靠性，增强抗干扰能力，可分为正边沿触发和负边沿触发。边沿触发器的特点是触发器的次态仅取决于时钟信号的上升沿或下降沿，在这以前或以后，输入信号的变化对触发器输出的状态没有影响。



图14 利用CMOS传输门的边沿D触发器

移位寄存器的用途是寄存数码，在移位脉冲作用下，寄存器中的数码可根据需要，向左或向右移动1位，可分为单向移位寄存器和双向移位寄存器，其中单向移位寄存器中又有右移寄存器和左移寄存器，而双向移位寄存器就是将右移寄存器和左移寄存器组合起来，并引入一控制端S便构成既可左移又可右移的双向移位寄存器，其电路如图15所示。

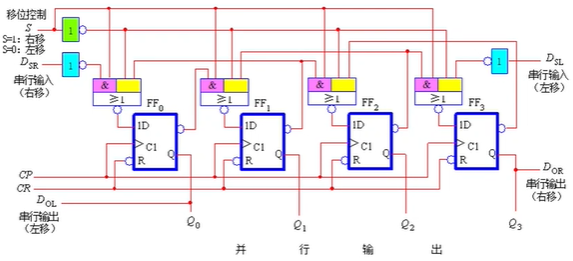


图15 双向移位寄存器

如果把移位寄存器的首尾连接起来，即把串行输比接回到串行输入上，那么在时钟信号的作用下，奇存器里的数据将循环右移，称为环形计数器，其特点是电路简单，N位移位寄存器可以计N个数，实现模N计数器状态为1的输出端的序号等于计数脉冲的个数，通常不需要译码电路，缺点是没有充分利用电路的状态，用n位的奇存器组成环形计数器只用了n个状态。而电路总共有2n个状态。此外还有扭环形计数器，只有一个触发器改变状态，所以译码时不会出现竞争-冒险现象，环形计数器和扭环形计数器电路如图16所示。

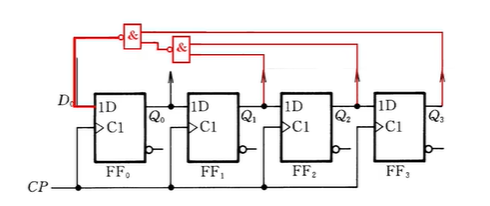
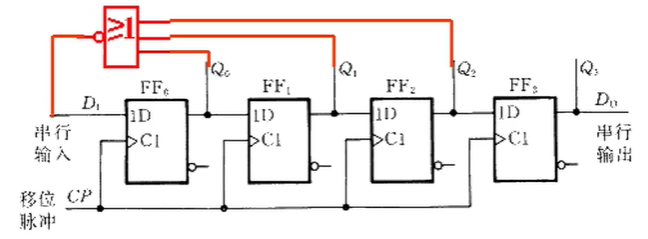


图16 环形计数器和扭环形计数器

在进行时序逻辑分析之前应该首先明确时序电路的特点，即任何一个时刻的输出状态取决于当时的输入信号和电路的原状态。分析时序电路的一般步骤为：一、由逻辑图写出各触发器的特性方程、时序电路的输出方程、各触发器的驱动方程；二、将驱动方程代入相应触发器的特性方程，求得时序逻辑电路的状态方程；三、根据状态方程和输出方程，列出该时序电路的状态表，画出状态图或时序图；四、根据电路状态表或状态图说明给定时序逻辑电路的逻辑功能。

时序逻辑电路还分为同步时序逻辑电路和异步时序逻辑电路。同步时序逻辑电路即所有触发器由同一个公共时钟信号来同步，异步时序逻辑电路即系统中的触发器由不同时钟信号驱动，除系统初始化时刻外，还将根据需要使用预置和清零。

同步时序逻辑电路的设计步骤为：根据设计要求设定状态，导出对应状态图或状态表，再进行状态化简，消去多余的状态，然后状态分配，又称状态编码，即把一组适当的二进制代码分配给简化状态图中的各个状态，选择触发器类型，根据编码状态表及所采用的触发器的逻辑功能导出待设计电路的输出方程和驱动方程，根据输出方程和驱动方程画出逻辑图，最后检查电路能否自启动。

## ③有限状态机

通过状态存储器和组合逻辑电路，可以记住过去输入完整历史的控制器电路，状态机是每个微处理器的核心部件，有限状态机的设计是数字系统设计的关键。有限状态机的特点是不停地经过一系列不同的状态；状态机执行被分配给该状态的操作；有限状态机的状态数量是有限的；有限状态机经过的状态序列可以是无限长的。有限状态机要实现的是控制一个状态到下一个的转换，同时在每个状态产生不同的输出信号进行操作。根据当前状态和当前输入决定下个状态即控制状态机状态序列的走向，根据当前状态和当前输入决定输出信号即分配每个状态在对应输入下的需执行的操作。

有限状态机可以分为摩尔型状态机和米利型状态机。摩尔型状态机的输出信号只依赖于当前状态，与当前输入信号无关，米利型状态机其输出信号同时依赖于当前状态和当前输入信号，两者结构图如图17所示。摩尔型状态机有更大的状态存储器、较大次态逻辑电路、输出逻辑电路相对简单；米利型状态机有较大的输出逻辑电路，因为其输入信号直接实时决定输出信号，属于异步输入。

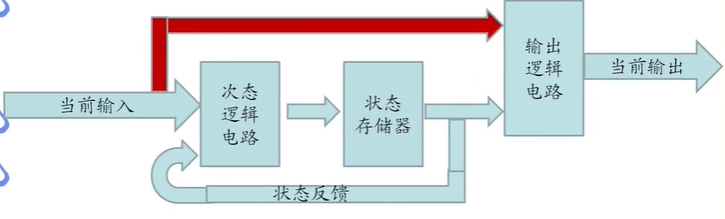


图17 摩尔型状态机和米利型状态机结构图

状态图是通过节点之间的连接来表示有限状态机的操作的，包含节点和有向边，状态图的画法为：一、确定状态机的有效状态；二、确定有向边，即状态间的转换及对应的转换条件；三、细化状态内的活动；四、将等价的状态化简成一个。

有限状态机分析指的是根据给定的有限状态机电路，分析其工作流程并进而得到它的功能，目标是画出所给电路的状态图，得到其工作流程。有限状态机分析的步骤为：一、根据电路中的次态逻辑电路及触发器特征方程导出次态方程；二、由第一步所得的次态方程，利用真值表的形式导出次态表；三、根据电路中的输出逻辑电路导出输出方程；四、由第三步所得的输出方程，利用真值表形式导出输出表；五、由第二步和第四步所得的次态表和输出表画出状态图，并得到其工作流程。

状态机综合指的是给我们一个具体的电路功能描述让我们获得实现这一功能的完整状态机电路，是状态机分析的逆过程，目的是画出具有所给电路功能的实际电路图。状态机综合的步骤为：一、根据电路功能描述画出状态图；二、由第一步所得的状态图，得到次态图；三、再由次态表得到次态方程；四、由状态图得到输出表；五、由输出表得到输出方程；六、由前面第三步和第五步所得的次态方程和输出方程画出对应的状态机电路。

状态机优化是指在保留电路功能条件，使电路尽可能小、速度快、功率低，其具体步骤为：一、优化状态存储器，减少状态数目；二、优化组合逻辑电路，优化状态编码，优化多余状态。

有限状态机的描述方式包括状态图（状态转移图）、次态图（状态转移列表），还可以用HDL语言描述状态机，用HDL语言描述状态机又可分为一段式、二段式和三段式。

微处理器可以分为专用微处理器和通用处理器，专用微处理器是专门用来处理某种特定任务的处理器，运算速度比从内存中调取指令再运算快几倍；通用处理器能够在不同的程序指今下完成不同的功能任务，每种运算所使用的逻辑电路根据系统需求进行编程，给定不同的指令集或程序，可实现不同的功能，如图18所示为微处理器的基本结构。

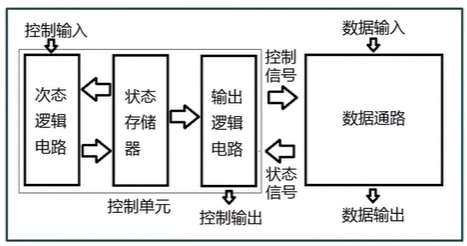


图18 微处理器的基本结构

数据通路设计指的是数据通路负责给定问题所涉及的全部数据操作的执行，从寄存器中读取数据，通过运算单元修改数据，然后将结果写讲寄存器的运算，定义为给定问题涉及的数据操作提供实际执行电路属于寄存器传输级层面的设计，目的是能执行给定问题的所有数据操作。数据通路设计需考虑因素有寄存器的选择、功能单元的选择、数据传输连接方式的选择和产生状态反馈信号。对于寄存器的选择，多数情况下，每个变量，在数据通路设计需要使用一个寄存器，两个以上变量不是同一时钟周期内访问寄存器时，可共用一个寄存器，存在多个信号输入到同一个寄存器，则需要使用多路数据选择器，使用多路寄存器后还需考虑存储器类型；对于功能单元的选择，要根据所有数据操作中包含的运算进行综合考虑；对于数据传输连接方式选择，确定寄存器和功能单元的连接方式，以实现他们间的正确连接；对于产生状态反馈信号，状态反馈信号是由数据通路中对信号进行条件测试获得的。

## ④通用处理器

本章主要讲述了两种微处理器设计分别是EC1电路设计以及EC2电路设计。

在进行微处理器设计时，应首先定义处理器的指令集，指令集定义好之后，微处理器则可以根据指令完成任务，在设计相应的状态机的状态图。如图19所示为EC1的指令集，包括指令、二进制码、操作、跳转指令，其中二进制码中又包括操作码和操作数，而其datapath则如图20所示。

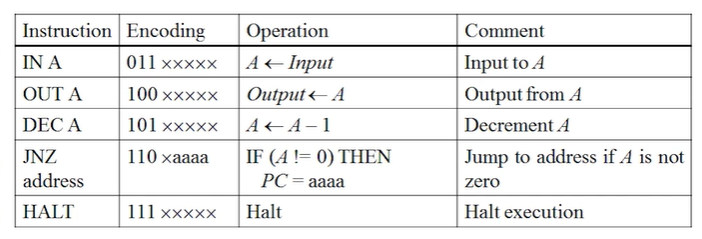


图19 EC1的指令集

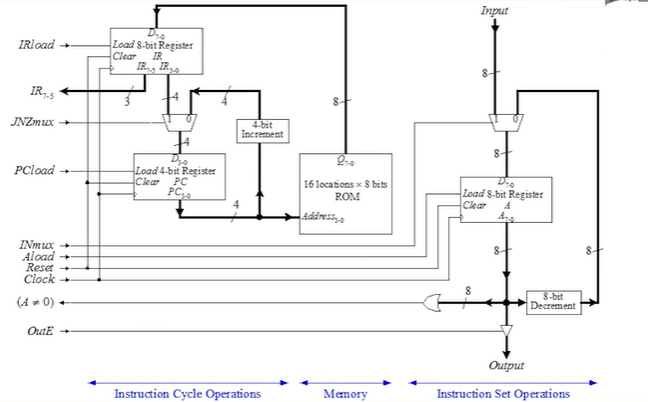


图20 EC1的datapath

指令集定义好之后，应该设计相应状态机的状态图，其状态图如图21所示，则其使用门级电路实现如图22所示，其最终的电路实现如图23所示，包括数据的输入输出环节、可以控制的信号、两个状态反馈的信号以及控制单元。

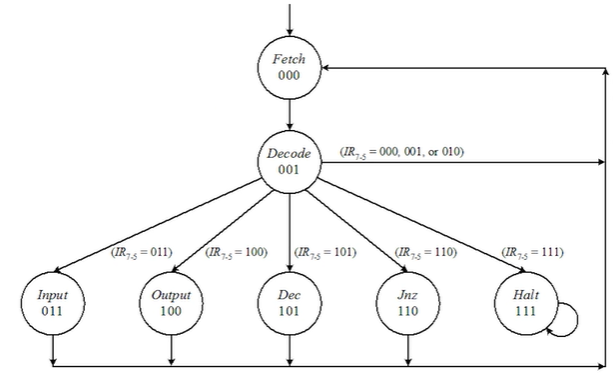


图21 EC1的状态图

如图24所示为EC2的指令集，同样包括指令、二进制码、操作、跳转指令，其中二进制码中又包括操作码和操作数，而其datapath则如图25所示。

指令集定义好之后，EC2状态图如图26所示，其最终电路实现如图27所示。其电路功能为RAM指定地址位置的数据放到累加器A里面；把累加器A里的数据放回到RAM指定的地址里面；把累加器里面的数读出来和RAM指定地址里面的数据读出来做相加并写回到累加器A；把累加器数据和RAM指定位置读出的数据做相减，然后把它写回到累加器A里去。

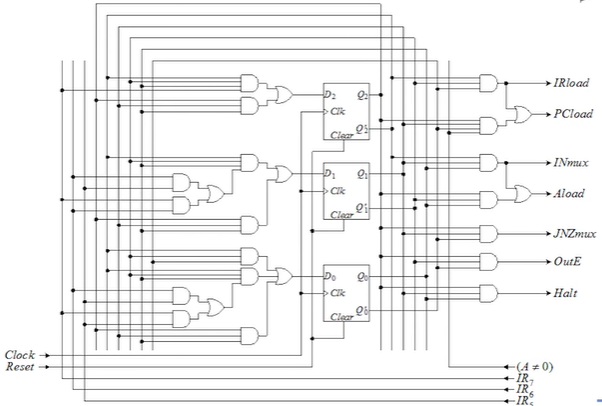


图23 EC1的门级电路实现

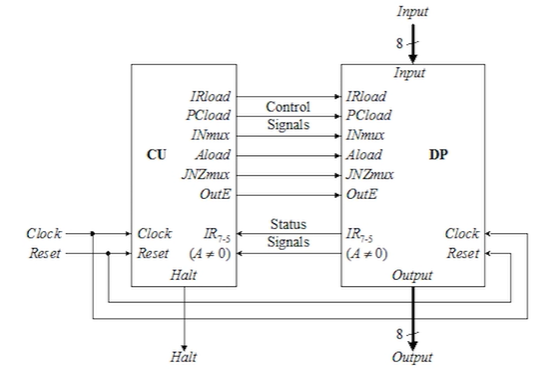


图22 EC1的电路实现

## ⑤数字系统接口

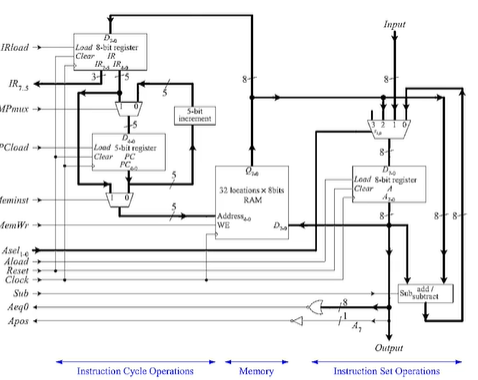


图25 EC2的datapath

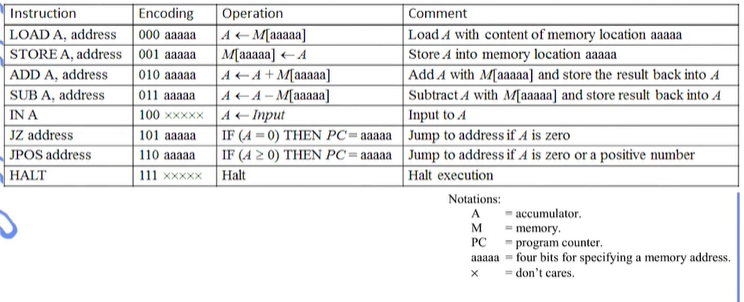


图24 EC2的指令集

本章内容主要涉及几种数字系统接口，如7段数码管和RS232、PS2键盘、PS2鼠标、VGA、A/D控制和I2C。输入/输出接口用于外部设备与CPU数据交换或控制连接电路，几种接口如图28所示。

7段LED数码管为显示数字的半导体显示器件，由七个段状发光二极管按8字形结构排列，其有两种接法，分别是共阴极接法和共阳极接法，共阳极指的是正极为八个发光二极管的共有正极，共阴极则与共阳极的接法相反，单独七段数码管需要七根连线和一个公共地，8个七段数码管需要56根连线，为减少连线数量，采取将每位数码管所有相同段都共同连接在一起，称为段选，再将公共端分别引出，称为位选。7段LED数码管采用动态驱动方式驱动，分时轮流控制各个LED数码管的公共端，只在短时间内一次打开一位数字，一组稳定的品示结果，不会有闪烁感。对于三位七段式数码管，使用3个比特位分别对应三位数字，称为位选，使用7个比特位分别对应7段，称为段选。

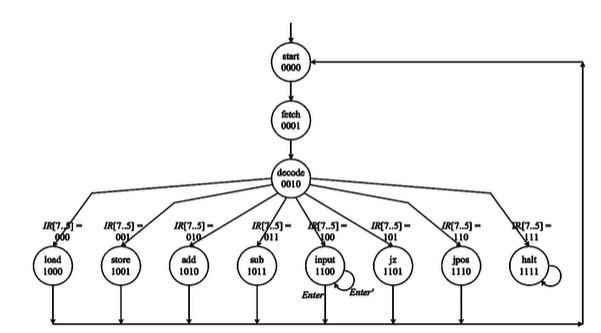


图26 EC2的状态图

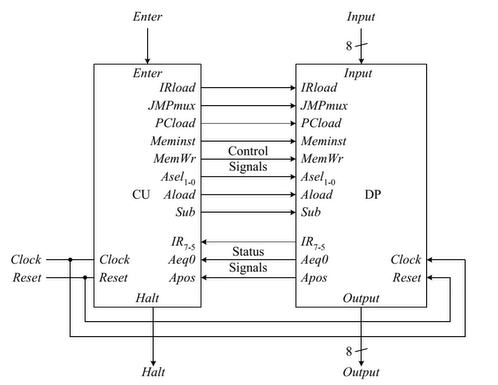


图27 EC2的最终电路实现

对于RS232接口，其特点是通信线路简单，协议比USB更简单，采用三线式通信方式，使用9针DB-9连接器中的3针，2脚用于接收，3脚用于发送，5脚为地，RS232没有时钟线，需要通信双方预先商定好通信的波特率，典型波特率为9600、38400和11520，空闲状态为1，以起始位0开始、8位数据位，可选的奇偶校验位，1位或更多位的停止位。RS232采用N81配置方式，没有校验位，8位数据位和1位停止位，发送端使用并串转换器，接收端使用串并转换器，9600波特率，使用相同的N81配置。

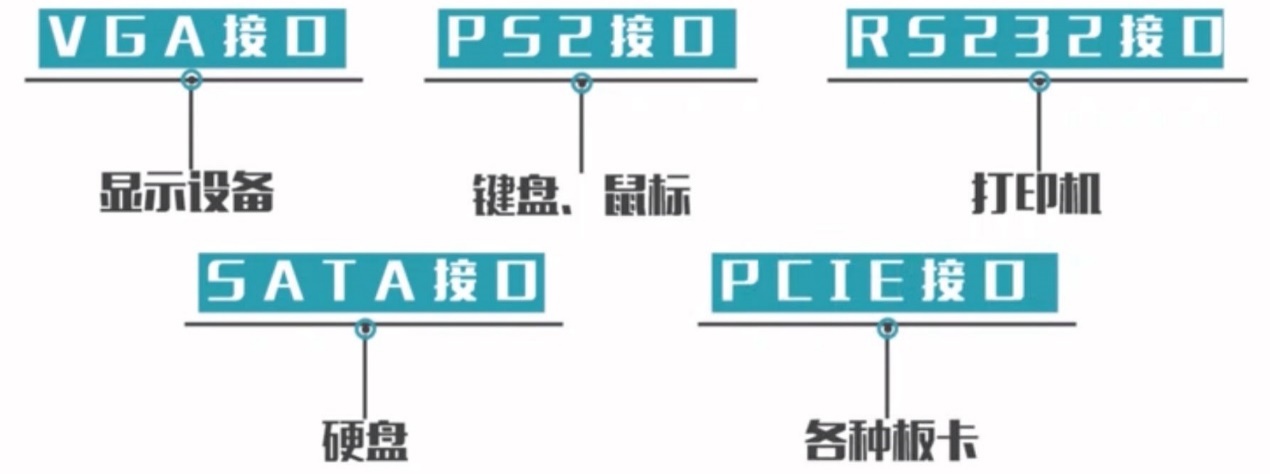


图28 几种接口示意图

PS2接口同时支持鼠标键盘连接，键盘使用简单的单向通道，鼠标使用较难的双向通道。PS2键盘使用两条信号线——时钟和数据，当按下（或释放）某个键时，连续向控制器发送该键的唯一代码，线发送的每个键码的数据由11位组成，起始位为0，8个从最低有效位到最高有效位的数据位，奇偶校验位，停止位为1，PS2接口的键值与键码如图29所示，PS2键盘有限状态机电路图如图30所示。

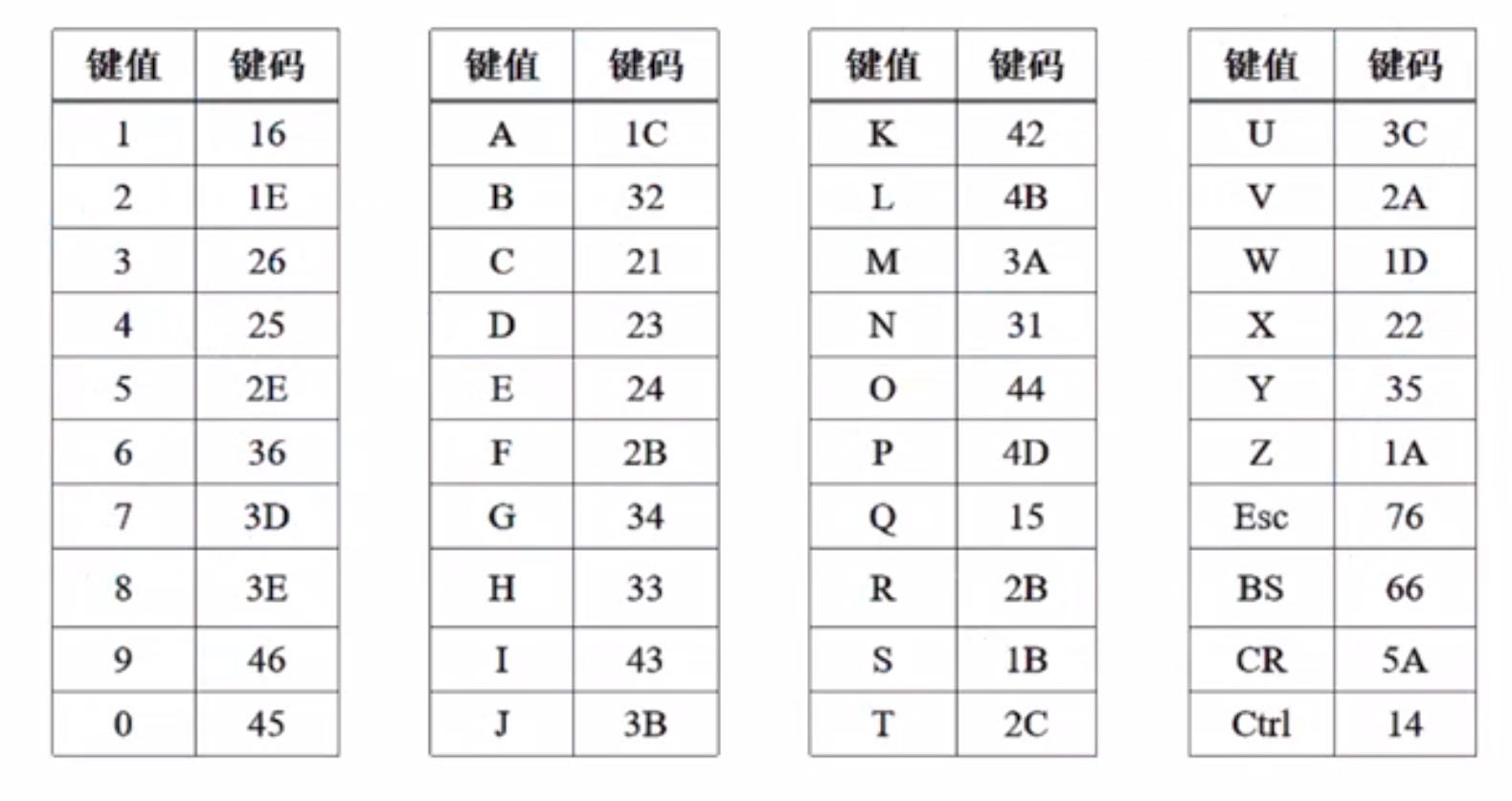


图29 PS2键值与键码

PS2鼠标协议与键盘协议的区别在于需要控制器的初始化命令，从MouseData和MouseClock，控制器需要首先向鼠标发送初始化命令，初始化后，Mouseclock线无法为控制器提供时钟。鼠标发送数据时，必须先下拉MouSeCIock线(至少100us)，设置为高阻状态来释放时钟，等待鼠标将MouseClock线拉至高电平，鼠标将在MouseClock线上生成时钟信号。标准PS2鼠标支持输入左右位移、上下位移、左键、中键和右键，固定的频率读取输入，并更新不同的计数器，来自鼠标的数据数据位被分组为每帧三个数据包，每个数据包由11位组成，起始位为0，奇偶检验位，停止位为1。

VGA接口是一种用于连接计算机和显示设备（如显示器、投影仪）的模拟视频接口，由IBM于1987年推出，最初用于其PS/2系列计算机。VGA接口采用15针D-sub连接器，传输模拟信号，支持分辨率最高可达640x480像素，但随着技术发展，也能支持更高的分辨率，如800x600和1024x768。尽管在长距离传输时可能会出现信号衰减和图像失真，VGA接口在许多老旧设备上仍然常见。

A/D控制器是将模拟信号转换为数字信号的电子元件，ADC芯片和微控制器之间串行数据连接。

I2C总线控制器为双向串行总线，具有简单性、有效性，占用空间小，标准模式下100kHz为最大时钟频率，快速模式下400kHz为最大时钟频率。其应用如DS3232芯片，是高精度数控时钟发生芯片，微控制器是主设备，DS3232是从设备。

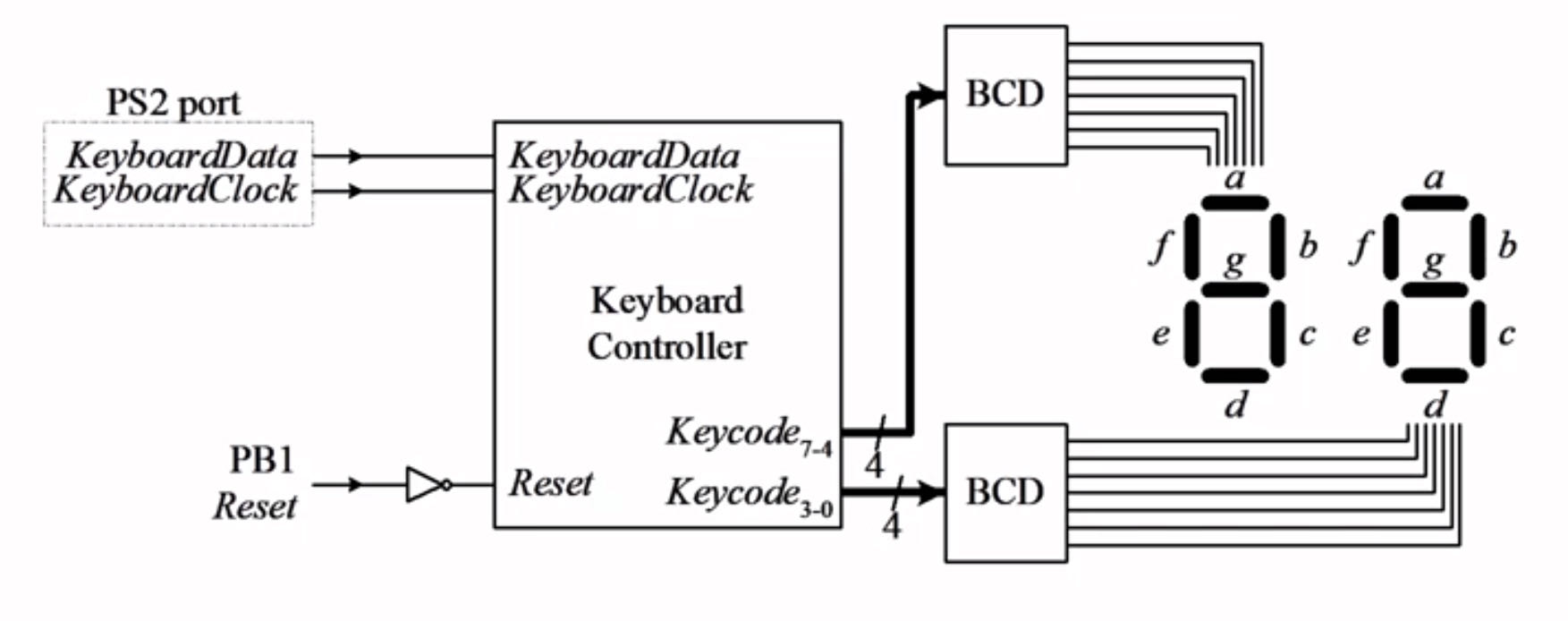
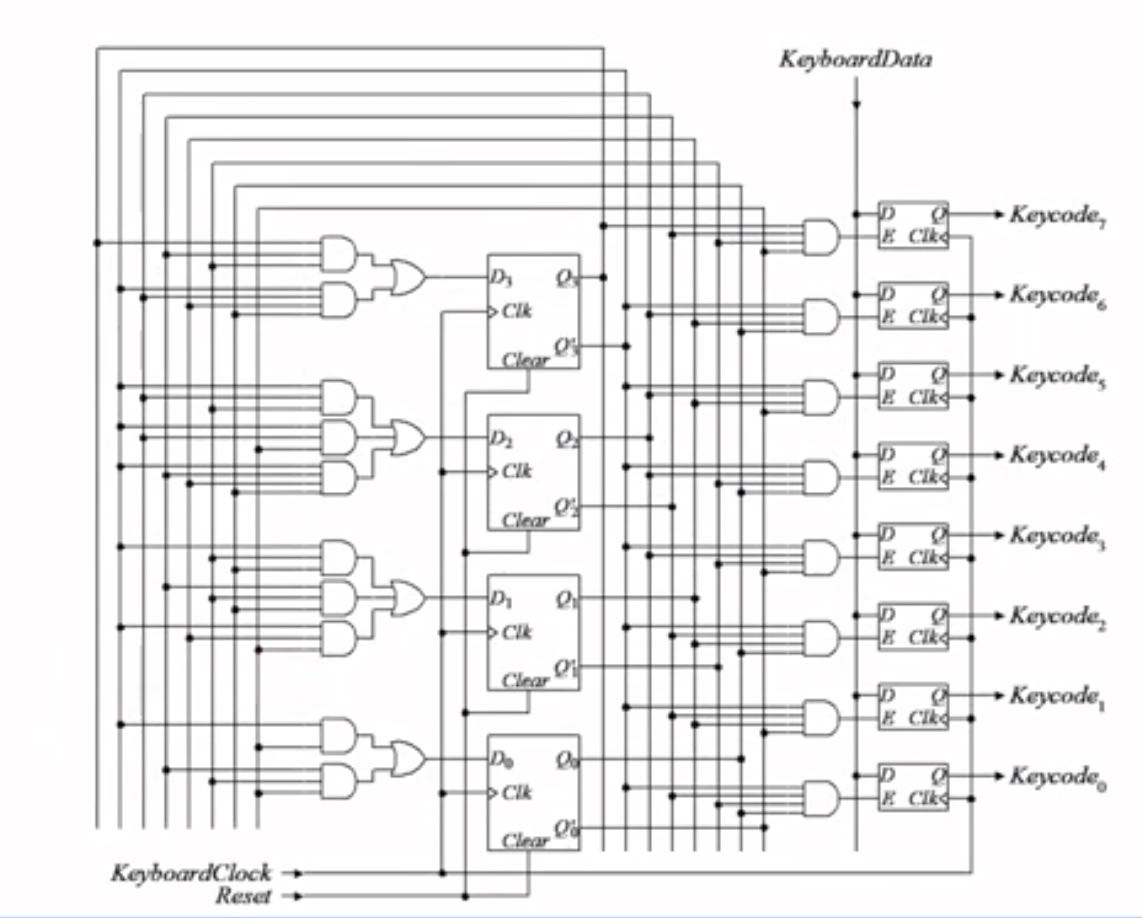


图30 PS2键盘有限状态机电路图

# 六、学习总结与心得

通过本课程的学习，初步掌握了数字系统设计的方法，并且学会了Vivado等工具的使用，并且温习了Verilog语言的使用，将以往学的数电、数集等知识应用到具体实际当中，掌握逻辑代数、逻辑函数以及组合逻辑、时序逻辑的基础知识；掌握组合逻辑的分析与设计方法，能够使用或替换逻辑部件；掌握同步时序逻辑分析及设计方法，熟悉状态机以及相关组件；掌握现代数字系统设计的方法与过程，学习使用硬件描述语言，能够设计控制器、微处理器；掌握可编程逻辑器件的基本原理及开发过程，熟悉EDA设计工具，能够设计开发数字系统。

在学习过程中，本人意识到，在实际设计中，EDA（电子设计自动化）工具如Verilog、VHDL等是必不可少的。这些工具不仅能提高设计效率，还能帮助我们进行仿真和验证，确保设计的正确性。在使用这些工具时，熟练掌握其功能和操作方法，可以大大减少设计过程中的错误和反复修改的时间。其次，数字系统设计不仅仅是对某个模块或电路的设计，更是对整个系统的整体把握。在设计过程中，我们需要从系统的角度出发，考虑各个模块之间的协调与配合，确保系统的稳定性和可靠性。这就需要我们具备全局观，能够从宏观上把控设计的方向和细节。最后，一个成功的数字系统设计往往需要经过大量的仿真和实际测试，才能确保其在各种复杂环境下的正常运行。

总的来说，数字系统设计是一项复杂而系统的工程，它需要我们具备扎实的理论基础、熟练的工具使用能力、系统级的思维以及耐心细致的工作态度。通过不断的学习和实践，相信我们一定能够在这一领域取得更大的突破和进步。