数字逻辑设计

1. 绪论

数字系统

模拟量：时间上连续，取值上连续，难以度量和保存且容易受到噪声干扰，但表示的值精确。

数字量：时间上离散，取值上也离散，灵活性高计算方便，易于存储，易于进行误差监测和修正，容易进行最小化，可以避免噪声累计。

A/D转换：模拟信号转换为数字信号，采样->量化->编码，产生数字序列。

D/A转换：模拟低通滤波器，将数字信号转换为近似的模拟信号。

数字系统：以数字的形式对信息进行存储、处理和传输的系统。

逻辑设计：决定数字逻辑单元的集合，执行一个指定的控制过程、数据加工或通信功能，并决定逻辑元件之间的互连。

设计：给定一个问题的表述，从可用功能部件集合中从优选择，得到一种解决问题的方法，并且该方法可以满足一定的约束条件。

二进制编码

二进制数：数据的一种数字表示方法，基数$r=2$，数字集为$\{0,1\}$。

编码：利用给定符号集合对信息的一种系统的标准化的表示。

二进制的信号表示：电压高低表示逻辑值，信号被置为1称为有效（或真），被置为0称为无效（或假）。

信号的极性：表示信号是高有效还是低有效。

正负逻辑：正逻辑下使用高有效信号，负逻辑下使用低有效信号。

George Boole：将逻辑表述映射为符号，采用数学的方法处理逻辑推理。

Claude Elwood Shannon：将布尔代数与硬件开关联系，开创信息论，提出bit作为度量信息量的单位。

数字电路

逻辑门符号：非，与（与非），或（或非）。

开关：数字系统物理实现的基本单元，使用开关可以构造更为复杂的功能。

组合电路：输出值是输入值的逻辑函数，输入值变化后一段时间后输出新的输出值，电路没有记忆功能，也没有循环反馈和时钟。

时序电路：输出值是输入值和电路状态的函数，输入变化后新的输出出现在下一个时钟事件或其他事件发生，电路有存储元件，有循环反馈。

计算硬件简史

$$Count \rightarrow Calculate \rightarrow Calculator \rightarrow Computer$$

早期计算工具：算盘，对数（1612-1614，John Napier），计算尺（1633，Oughtred）

机械计算机：Pascal加法器，1642-1643，Blaise Pascal；Leibniz四则运算计算器，1673，Leibniz；差分机和分析机，19世纪，Charles Babbage，第一台计算机。

穿孔卡片和程序：穿孔卡片式织布机，1804，Joseph Jacquard；概念性程序提出，1815-1852，Ada Byron；记录式加法器，1886，Burroughs；CTR公司，1911，IBM公司，1924。

继电器计算机：继电器，1835，Joseph Henry；电话开关网络，1876，AT&T公司，Alexander Bell；Z-1，1935，Konrad Zuse；Model-K，1937，George Slibitz，Bell Lab；Mark-1,1944，IBM/Harvard。

第一台数字电子计算机：ABC，1942，使用真空管，二进制。

第一台通用的数字电子计算机：ENIAC，使用真空管，十进制。

第一代数字电子计算机：真空管实现。

第二代数字电子计算机：晶体管实现。

第三代数字电子计算机：集成电路。

第四代数字电子计算机：大规模集成电路，微处理器。

Moore定律：一块芯片上的晶体管数量每18个月翻一番。

2. 数据编码和布尔代数

数制

r进制按位计数法：.为基点，r为基数，n为整数位数，m为小数位数，a_i $(n-1 \geq i \geq 0)$为整数部分第i位，a_{-i} $(-1 \geq i \geq -m)$为小数部分第i位。

$$\begin{aligned} N &= (a_{n-1}a_{n-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-m})_r \\ N &= a_{n-1}\times r^{n-1}+\cdots+a_0\times r^0+a_{-1}\times r^{-1}+\cdots+a_{-m}\times r^{-m} \\ &= \sum_{i=-m}^{n-1} a_i\cdot r^i \end{aligned}$$

二进制的优势：两个稳态的材料容易获取，进制取$e=2.718\cdots$时最节约材料。

基本算术运算：加+，减-，乘×，除÷

r进制加法进位规则：逢r进1，借1当r

进制转换：逐步代入法，整数取余法和乘数取整法。

逐步代入法：将A进制数转为十进制数，将其带入N的计算公式完成转换。

除数取余法：对整数部分采用，通过连除法完成转换。

$$N = B(b_{n-1}B^{n-1}+\cdots+b_1B^0)+b_0B^0$$

乘数取整法：对小数部分采用，通过连乘法完成转换。

$$N=\frac{b_{-1}B^0}{B}+\frac{1}{B}(b_{-2}B^{-1}+\cdots+b_{-m}B^{-(m-1)})$$

A进制向B进制转换：逐步代入法（B=10）+除数取余法和整数取整法；或进行如下的转换，$A\to 10\to B$。

十进制向A进制转换：整数取余法和乘数取整法。

A向A^k进制数的转换：将A进制数从基点起向左右分，每k个数字分一组，不够补0，每一组中用一个A^k进制数代替。

A^k向A进制数转换：将A^k进制数中的每个数字用k个A进制数代替。

数的表示

原码：$N=(sa_{n-1}a_{n-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-m})_{原}$，$s\in\{0,r-1\}$，s取0时表示正数，取r-1时表示负数。原码时最直接的表示，存在+0和-0的问题，实现的电路也更慢更复杂，数字系统中不常用。

补运算：n位r进制数$(N)_r$的补定义为

$$[N]_r=r^n-(N)_r$$

补码算法1：多项式表示$A=a_{n-1}(-2^{n-1})+\sum_{i=0}^{n-2}a_i2^i$

补码算法2：对$(N)_r$中的数字，从低位起寻找第一个非0的数字a_i，将其用$r-a_i$替换，其余每个数字a_j使用$(r-1)-a_j$替换。

补码算法3：每位数字a_i用$(r-1)-a_i$替换，末位加一。

补码：正数$N=+(a_{n-2}\cdots a_0)_2=(0a_{n-2}\cdots a_0)_{补}$；负数$N=-(a_{n-1}\cdots a_0)=[a_{n-1}\cdots a_0]_r$；$(10\cdots0)$表示$-2^{n-1}$以避免出现-0重码。

正数表示范围$[0,2^{n-1}-1]$，负数表示范围$[-2^{n-1},-1]$。

补码运算：补码用来将减法转换成加法，从而减少机器硬件（只需要加法器即可）；然而使用补码运算需要忽略进位，但必须判断溢出。

$$\begin{gathered}\text{对于}B\geq0,C\geq0\\A=B+C,\ (A)_2=(B)_2+(C)_2\\A=B-C,\ (A)_2=(B)_2+[C]_2\\A=-B-C,\ (A)_2=[B]_2+[C]_2\end{gathered}$$

溢出：$N>2^{n-1}-1$时变为负数（上溢出），或$N<-2^{n-1}$时变为正数（下溢出）。最高位进位与次高位进位不同**则发生溢出。

反运算：将$(N)_r$中的每个数字a_i替换为$r-1-a_i$。

反码：数$(N)_r$的反码$[N]_{r-1}$定义为

$$\begin{gathered}[N]_{r-1}=r^n-(N)_r-1\\[N]_{r-1}=[N]_r-1\end{gathered}$$

定点表示：计算机中表示定点数的方法，定点整数小数点默认在最低位右侧，定点小数小数点默认在符号位和最高位之间，可以使用原码、补码和反码表示。

负电表示：计算机中表示指数（科学计数法）的方法，IEEE 754标准。

BCD码：8421码，十进制的二进制编码，用四位二进制表示十进制数字0-9。

BCD加法：首先直接使用二进制加法，若结果小于9则没有溢出，就是BCD码；否则加法大于9，将结果加上6（0110）。

二进制转BCD码：二进制数左移，若移位次数满足，则BCD位于各个列中，结束；否则，若任何列中数制大于等于5，则该列上+3，之后继续左移。

操作	百位	十位	个位	二进制	二进制
十六进制				A	B
初始状态				1010	1011

ASCII编码：7-bit编码符号，第8bit用来校验。

格雷码：相邻数字的编码Hamming距离为1，是一种循环码。

通讯中的编码

重量：$w(I)$为n位信息编码I中的1的个数。

距离：$d(I,J)$为n为信息编码I和J中的不同的位的个数。

最小距离：d_{min}，若信息编码提供t位纠错能力和附加s位的检错能力，则有

$$2t+s+1\leq d_{min}$$

奇偶校验码：blablabla

汉明码：blablabla

布尔代数

公理1（封闭性）：布尔代数是一个封闭的代数系统，它包含一个集合B，B包括两个以上的元素，B上定义两个二元操作$\{\cdot\}$和$\{+\}$，一个一元操作$\{'\}$，分别称作与或非；对$\forall a,b\in B,\ a\cdot b\in B,\ a+b\in B$。

公理2（交换律）：对于任何B中元素a和b

$$a+b=b+a$$
$$a\cdot b=b\cdot a$$

公理3（结合律）：对于任何B中元素a，b和c

$$a+(b+c)=(a+b)+c$$
$$a\cdot(b\cdot c)=(a\cdot b)\cdot c$$

公理4（恒等性）：在集合B中存在唯一的元素0和1，对集合中任意元素a都有下式。0对于+是恒等单元，1对于·是恒等单元。

$$a+0=a$$
$$a\cdot 1=a$$

公理5（分配律）：对于任何B中元素a，b和c

$$a+(b\cdot c)=(a+b)\cdot(a+c)$$
$$a\cdot(b+c)=(a\cdot b)+(a\cdot c)$$

公理6（互补）：对于任何B中元素a，在B中存在唯一的元素a'（也可写作$\bar{a}$），使得

$$a+a'=1$$
$$a\cdot a'=0$$

定理1（幂等性）：$X+X=X,\ X\cdot X=X$

定理2（空单元）：$X+1=1,\ X\cdot 0=0$

定理3（自反律）：$\bar{\bar{X}}=X$

定理4（吸收律）：$X+XY=X,\ X(X+Y)=X$

定理5：$X+\bar{X}Y=X+Y,\ X\cdot(\bar{X}+Y)=XY$

定理6：$XY+X\bar{Y}=X,\ (X+Y)(X+\bar{Y})=X$

定理7（摩根定律）：$(X\bar{+}Y)=\bar{X}\bar{Y},\ (\bar{XY})=\bar{X}+\bar{Y}$，用于求布尔表达式的反。

一般的摩根定律。

$$(X+Y+Z+\cdots)'=X'Y'Z'\cdots$$
$$(XYZ\cdots)'=X'+Y'+Z'+\cdots$$

定理8（对偶原理）：正确的布尔表达式和运算过程，将其中的0换为1,1换为0，与换为或，或换为与，依然成立。

定理9：$(X+Y)(\bar{X}+Z)=XZ+\bar{X}Y,\ XY+\bar{X}Z=(X+Z)(\bar{X}+Y)$

定理10：$XY+YZ+\bar{X}Z=XY+\bar{X}Z,\ (X+Y)(Y+Z)(\bar{X}+Z)=(X+Y)(\bar{X}+Z)$

3. 组合逻辑和CMOS

布尔函数范式

积之和SOP：形如$f(A,B,C)=AB+A'C+AC'$的布尔函数范式。

和之积POS：形如$f(A,B,C,D)=(A'+B+C)(B'+C+D')(A+C'+D)$的布尔函数范式。

最小项范式：对应于积之和，将布尔函数的变量按顺序编码，编码$(a_1a_2\cdots a_k)_2$对应于$f(A_1,A_2,\cdots,A_k)$，将使得f取1的各项对应的编码相加，得到最小项范式。

$$\begin{aligned}f(A,B,C)&=A'B'C'+A'BC'+ABC'+A'BC+ABC\\&=m_0+m_2+m_6+m_3+m_7\\&=\sum m(0,2,3,6,7)\end{aligned}$$

最大项范式：对应于和之积，将使得f取0的各项对应的编码相乘，得到最大项范式。

$$\begin{aligned}f(A,B,C)&=(A+B+C)(A+B+C')(A'+B+C)(A'+B+C')\\&=M_0\cdot M_1\cdot M_4\cdot M_5\\&=\Pi M(0,1,4,5)\end{aligned}$$

非确定函数：一些确定的输入组合不会产生，一些输出为0或1只对特定的输入组合成立。

无关最小项：忽略一些最小项，d_i

无关最大项：忽略一些最大项，D_i

逻辑门的CMOS实现

开关模型：开关在控制信号的控制下，连接两个端点。

正开关：控制信号为0时开（断开不连接），为1时关（合住连接）。

负开关：控制信号为1时开，为0时关。

电源：逻辑1。地：逻辑0。

非门NOT：被动上拉模型，主动上拉模型。

与非门NAND：被动上拉模型，主动上拉模型。

或非门NOR：被动上拉模型，主动上拉模型。

AND，OR，NOT可以只用NAND或NOR来构造！

半导体器件

N型半导体：Si+五价元素（P），自由电子导电。

P型半导体：Si+三价元素（B），空穴导电。

PN结：连接P型和N型半导体，连接处为PN节。平衡状态下PN节连接处电荷平衡，称为“耗尽层”；N区高电压P区低电压时不导电；P区高电压N区低电压时导通。

二极管：P导线连接PN结两端，P区端口为阳极，N区端口为阴极。

三极管：双极性晶体管，三极分别为基极，发射极，集电极，基极电压高过阈值时，两个PN结导通，发射极和集电极在电压作用下产生电流。分NPN型和PNP型。

MOS：金属氧化物半导体场效应管（FET）。

NMOS：P作衬底，挖出N沟道，NPN型三极管。栅极高电平时，源极和漏极导通。

PMOS：N作衬底，挖出P沟道，PNP型三极管。栅极低电平时，源极和漏极导通。

CMOS：NMOS和PMOS形成互补电路，电流小功耗低，集成度高。

CMOS等比伸缩原理：C负载电容，Vdd电压，f频率。

$$Power \propto C \times V_{dd}^2 \times f$$

CMOS门电路：反向器，与非门，或非门，缓冲器，异或门，特殊门...

反向器NOT，缓冲器：单输入，单输出。缓冲器即为两个非门相连。

与非门NAND，或非门NOR：二输入，多输入。

异或门：AB信号不同则输出1，相同则输出0。最简单的做法，使用8个4组耗尽型三极管对AB信号组合进行选择即可。

传输门：NMOS与PMOS开关对接而成，当NMOS和PMOS同时开，传输门两端电阻非常小。传输门可以用于2选1逻辑。

三态门：EN和A输入，OUT输出，只有当EN信号有效时，OUT=A，否则OUT无意义。三态门可以用于驱动总线。

漏开路门：两个NMOS串联，一端输出一端接地，没有PMOS作为上拉电阻。当AB信号中任何一个为L，输出端开路，当AB信号全为1，输出0，相当于开路的NAND。使用开路门可以实现多输入与门，这些开路门输出相连，共用一个上拉电阻。

避免扇出相连！否则有可能导致电源与地直接相连。

组合电路的分析和设计

电路设计：从一个电路功能的描述到一组开关函数进而到门级，PLD或其他逻辑单元实现的转化过程。

电路分析：从一个数字电路的实现除法，得到电路的某种形式的功能描述。

开关表达式，真值表，时序图等行为描述。

代数分析法：用开关代数来获取指定的功能形式。

真值表分析法：逐次分析各个门的真值表，直到输出为止。

时序图分析法：开关网络的输入和输出信号关系在时间维度上的图形表示，显示中间信号和传播延迟。

好的电路：减少输入数目（较少的输入意味着较少的晶体管和较快的门，并且扇入会受到工艺限制），减少门的个数（较少的门意味着较小的电路和更低的成本）。

好的实现需要在增加电路延迟和规模之间做权衡。

两级组合逻辑电路：先与后或AND-OR，先或后与OR-AND

AND-OR逻辑电路：开关表达式采用SOP形式。

OR-AND逻辑电路：开关表达式采用POS形式。

布尔表达式合并定理：$A(B'+B)=A$

布尔立方体：合并定理的图形化技术，n个输入变量对应n维立方体。

合并定理将立方体的两个元素合并成一个元素。

多级组合逻辑电路：由于扇入有限制，因此采用多于两级的电路。

圆泡移动法：加入电路使用NOR实现，则在与或非门电路中加入双圆泡，将所有OR输出取反，将所有AND输入取反，最后使用非门将多于圆泡替换，最终所有门都是NOR。NAND类似。

与或非电路AND-OR-INV：一组与门之后跟一个或非门，用来实现两级SOP。

4. Verilog和FPGA

Verilog硬件描述语言

HDL硬件描述语言：结构级描述——原理图的文本替代和功能模块的模块层次化组合；行为级/功能级描述——描述模块应该和不应该做什么，不对如何做进行描述，可以综合产生模块电路。

事件驱动的模拟技术，模拟数据流并行的硬件电路。

Module：说明输入输出，双向和内部信号。连续赋值下，一个门的输出任何时刻都是输入的函数结果，不需要进行调用；传播延迟下，输入影响输出的时间和延迟的概念。

类型：Input，Output，Wire，Reg。

赋值：阻塞赋值（=），非阻塞赋值（<=）

Assign：可以对output，reg，wire类型赋值，无论输入值是否变化，都会执行。

Always @()：@()中为敏感向量表，只有当其中的值发生变化时才执行。Always中可以有if-else和case语句。

阻塞赋值：交换阻塞赋值的顺序，依然正确，描述的硬件是一样的。

信号连接：$\{a,b\}$，将a和b信号连接起来，由于a和b已经定义了长度，因此连接的长度也是固定的。

FPGA现场可编程门阵列

三态门：用于驱动总线。

地址译码：共享总线，输入信号最为nbit地址+En，经过译码器得到输出信号，输出信号在待选单元中进行选择，被选择的单元使用输出总线进行数据输出。

6-input LUT：由两个共用输入的5-input LUT组成，第6bit输入作为选择信号，具有一个或两个输出，支持任意6变量函数（单输出）或者两个独立的五变量函数（双输出）。

FPGA设计流程：设计输入，综合，设计验证，后端设计

5. 标准模块化组合电路

组合逻辑的一般设计方法：理解问题=>采用适合的设计表达方式进行形式化描述=>选择实现目标=>进行实现过程

自顶向下设计：将设计分解为已经定义的或容易实现的多个简单的电路。

自底向上设计：将已经完成的设计互连而完成设计目标。

多路选择器和多路分配器

多路选择器MUX：多个输入连接到一个输出。

2选1 MUX：两个数据输入X,Y，一个控制输入S，使用传输门实现2选1。

$$Z=\bar{S}X+SY$$

$2^n数据输入（第i个用I_{i}$表示，从0计数），n控制信号，1输出。

多选级联：大多选器可以由多个小多选器级联而成。

MUX作为通用逻辑：一个2^n：1多选器可以实现任意n输入变量的布尔函数。变量作为控制信号输入，数据输入固定为0或1，其本质是一个LUT。

一个2^{n-1}：1多选器可以实现任意n变量的布尔函数。n-1个变量作为控制信号，数据输入连接最后的变量，它的反以及0或1。

译码器/多路分配器DEMUX：一个输入连接到多个输出。

多路DEMUX：一个数据输入（使能端，G），n个控制输入（S），2^n个输出。

DEMUX作为通用逻辑：一个$n:2^n$译码器可以实现n个变量的函数。变量作为控制信号输入，使能信号固定连接1，相当于最小项生成器。

可编程逻辑部件PLA

与/或门的预先制造模块：实际上采用NOR/NAND，在门之间建立或断开连接来进行编程。NAND积之和形式的可编程部件，输入经过AND阵列转换为积项，之后积项经过OR阵列得到输出。

共享积项：输出函数之间共享积项。

编程前：所有连接都有效，所有的AND和OR实际上都是NAND。

编程后：将不需要的连接去除。

熔丝型：通常情况下连接，编程后熔断不需要的连接。

反熔丝：通常情况下不连接，编程后连接。

积项表：每一行为函数中的一个积项，第一部分为积项部分；第二部分为输入，每列一个输入，表中填1表示使用该输入，填0表示使用该输入的反，填-表示该输入不参与该积项；第三部分为输出，每列一个输出，填0表示不连接该积项，填1表示连接该积项。

存储器实现逻辑

地址译码：用于总线共享。

ROM：每行一个字word，行大小即为字长，字的索引为地址，一个字使用一根字线；各行的相同位共用位线。译码器接受地址作为输入，译码后选择相应字线，所选的字由位线输出。

编码器

Encoder编码器：对每个输入信号分配一个唯一的二进制编码，实现译码器的反函数功能。

互斥唯一输入编码器：输入信号在任意时刻，都只能有唯一的一个输入是有效的。不可能出现多于一个输入同时有效，若出现则硬件输出为d。

非互斥编码器：任何一个特定时刻，可能出现多于一个输入同时有效。若出现多个输入同时有效，则输出全零编码。

优先级编码器：任何一个特定时刻，可能出现多余一个输入同时有效，若出现多个输入同时有效，那么输出按照输入的优先级编码。增加两个特殊的输出，GS表示一个或多于一个输入有效，EO表示输入全部无效，GS与EO互斥。

逻辑函数单元

多功能的函数模块：3根控制信号标明进行的操作（0,1，OR，NAND，XOR，NXOR，AND，NOR），两根数据输入作为操作数，1根输出作为结果。

加法器

半加器：$S_i = X_i \hat{} Y_i$, $C_i = X_i Y_i$。

实现：2个门，1XOR，1AND。

全加器：$S_i = X_i \hat{} Y_i \hat{} C_{i-1}$, $C_i = X_i Y_i + X_i C_{i-1} + Y_i C_{i-1}$

标准实现：6个门，2XOR，2AND，2OR。

另一种实现：5个门，两个半加器，2XOR，2AND，1OR。

减法器：C_0输入为0表示加法，为1表示减法；减法时，减数输入取反。

溢出检测：C_{n-1}、C_n = 1则出现溢出。

行波进位加法器：全加器串联。

关键路径：低位到高位传播进位。

$$
\begin{aligned}
&\text{半加器：} t_{add} = 3t_{gate},\ t_{carry} = 2t_{gate} \\
&\text{全加器：} t_{add} = 3t_{gate},\ t_{carry} = 2t_{gate} \\
&\text{行波进位加法器：} \\
&\quad t_{add} = (n-1)t_{carry} + t_{add} \\
&\quad\quad = (2n+1)t_{gate}
\end{aligned}
$$

超前进位加法器：引入进位产生因子G和进位传播因子P。

$$
\begin{aligned}
G_i &= A_i B_i \\
P_i &= A_i \text{、} B_i \\
S_i &= P_i \text{、} C_i \\
C_{i+1} &= G_i + C_i P_i \\
&= G_i + G_{i-1} P_i + C_{i-1} P_{i-1} P_i \\
&= \cdots\cdots
\end{aligned}
$$

先行进位链：对每一个进位都使用P，G和C_0进行表示，越高位进位的公式越为复杂，但所有进位公式都可以使用两级逻辑来实现，所有输入信号都来源于原始输入信号而不是中间进位信号。

级联先行进位链：由于先行进位链不可能很长，因此对小的先行进位加法器再使用先行进位链，得到级联先行进位链的先行进位加法器。

进位选择加法器：使用冗余的硬件来加快进位产生。