数字逻辑设计

6. 卡诺图

卡诺图

卡诺图：开关函数最常用的表现形式，寻找MSOP和MPOS的最基本的启发式方法形式。卡诺图的每一个单元用响应的真值表行号标记，相邻单元的真值表输入只有一位不同。

平面卡诺图最大可以表示6个变量（3x3）的开关函数。

CD\AB	00	01	11	10
00	m0	m4	m12	m8
01	m1	m5	m13	m9
11	m3	m7	m15	m11
10	m2	m6	m14	m10

函数范式到卡诺图的映射：设f是n个变量的开关函数($n \leq 6$)，卡诺图的单元编码为0~2^n-1，每个数字对应f真值表的一行。若m_i是f的最小项，则将1放在卡诺图的单元i中；若M_i是f的最大项，则将0放在单元i中；若d_i是f的无关项，则将d放在单元i中。

最小项范式形式时，只画1和d即可；最大项范式形式时，只画0和d即可。

逻辑相邻：物理相邻的单元，或者边缘单元和另一边缘的单元都是逻辑相邻的。

卡诺图化简卡关函数：卡诺图中单元若是逻辑相邻的，并且两个单元同时包含逻辑1，则两个单元可以被合并成消除单元标号中的一个单元中是1而另一个单元中是0的变量。

单元合并后的组的大小可以为$2, 4, 8, \cdots, 2^k$个单元。

合并尽可能多的单元，会使得组内对应的扇入尽量少。

尽可能使用最少的组覆盖所有最小项，使得结果中包含尽量少的积项。

SOP的卡诺图化简

蕴含项：一个积项，它可以覆盖一个函数的一些最小项。

质蕴涵项：一个不被该函数的其他蕴含项覆盖的积项。

实质蕴含项：一个至少包含一个最小项不能被其他质蕴涵项集合覆盖的质蕴涵项。

覆盖：函数的每个最小项都被该组内的蕴含项覆盖，则函数被一组蕴含项覆盖。

最小覆盖：一个包含最少的质蕴涵项和最少的符号的覆盖。

算法1

1．计算出卡诺图中每个最小项的相邻单元个数
2．选择没有被包含的具有最小相邻数的最小项；若有多个选择，则随机选取一个
3．生成这个最小项的一个质蕴涵项并将它放入覆盖；若有多个选择，则选择覆盖了最多的还未被覆盖的最小项的质蕴涵项
4．重复2-3，直到覆盖了所有的最小项

算法2：使用卡诺图获得最小的积之和。

1．任意选取一个单元
2．找到包含这个单元的由1和X组成的最大的组，形成质蕴涵项
重复1-2，知道找到所有的质蕴涵项
3．重新考察卡诺图中所有的1；若只被一个质蕴涵项覆盖，则该质蕴涵项时实质蕴含项，必然出现在覆盖中；被实质蕴含项覆盖的1，不需要再进行考察
4．如果还存在一些1没有被任一实质蕴含项覆盖，则选择最小的质蕴涵项集合覆盖剩余的所有1

POS的卡诺图化简

蕴含项：一个和项，可以覆盖一个函数的一些最大项。

质蕴涵项：一个不被函数的任何其他蕴含项覆盖的和项。

实质蕴含项：一个至少包含一个不能被其他质蕴涵项集合覆盖的最大项的质蕴涵项。

覆盖：一组蕴含项，使得函数中的每个最大项都被该组内的某个蕴含项覆盖。

最小覆盖：包含最少的质蕴涵项和最少的符号的覆盖。

算法3

1．计算出卡诺图中每个最大项的相邻单元个数
2．选择没有被包含的具有最小相邻数的最大项；若有多个选择，则随机选取一个
3．生成这个最大项的一个质蕴涵项并将它放入覆盖；若有多个选择，则选择覆盖了最多的还未被覆盖的最大项的质蕴涵项
4．重复2-3，直到覆盖了所有的最大项

非确定函数的化简和静态冒险解决

非确定函数：无关最小项d_i，忽略一些最小项；无关最大项D_i，忽略一些最大项。无关项值为X，化简时可以根据无关项对于函数化简是否有帮助（减少符号数，减少质蕴涵项个数）来确定无关项的值为0还是1。

包含无关项的POS和SOP化简后不一定在形式上互补。

要求非确定函数卡诺图化简时，**质蕴涵项和实质蕴含项中至少包含一个确定性的最大/小项。**

冒险：组合电路中不相同的传输延时，导致不正确的输出。

静态冒险：输出相对正确值的瞬间变化。例如静态1冒险为输出从1到0到1。

动态冒险：输出在状态变化时变化多次。例如动态0到1冒险为输出从0到1到0到1的变化。

静态1冒险：SOP或与或电路，会产生静态1冒险。冒险产生的原因是卡诺图中存在质蕴涵项之间完全独立，没有关联。**解决方案**是添加相邻的最小项的公共积项。

静态0冒险：POS或或与电路，会产生静态0冒险。冒险产生的原因是卡诺图中存在质蕴涵项之间完全独立，没有关联。**解决方案**是添加相邻的最大项的公共积项。

7. Q-M方法

Quine-McCluskey方法

优点：直接的，系统的，可计算的方法；处理变量的数目可以多于6个，可以处理多输出函数。

第一步：枚举所有蕴含项。首先列举出所有的最小项，并按重量（含有1的数目）分组；之后重量相邻的组中元素进行合并，得到新的蕴含项的组；不断迭代，直至无法合并，得到所有的蕴含项。

第二步：获取所有的质蕴涵项。最后一次合并的组中元素，必然全部都是质蕴涵项，用PI标记；之后从右向左从上向下寻找没有被覆盖的蕴含项，其也必然是质蕴涵项，用PI标记；标记完成后从右向左从上到下对PI标号。

第三步：构造质蕴涵项表（PI表）。PI表的每一行对应一个PI，每一列对应一个最小项；表中对应位置用？标出各个PI覆盖了的最小项。

第四步：简化PI表，获得最小覆盖。

> 1．标识出所有只被某个PI覆盖的最小项，讲这些PI加入覆盖中。
> 2．在PI表中去掉第1步标出的PI行，和这些PI覆盖的最小项的列。若为空表则过程结束，否则按如下规则简化，若某行或列被其他行或列覆盖，则删除该行或列。
> 3．如果产生循环表，即没有实质蕴含项，也没有行列覆盖，则进入第4步；否则再次执行第1和2步。
> 4．任意选择一个质蕴含项，优先选择包含最小项数目多的PI。重复执行直到产生空表或不再是循环表，若表非空则进入第1步。

非确定函数的Q-M方法：生成PI时，同时使用最小项和无关项。生成最小覆盖时，只使用最小项。

多输出函数

卡诺图：对每个输出用一次卡诺图化简，得到结果。化简时彼此独立，没有利用公共积项。

Q-M方法：对每一个最小项，标记函数标号，只有当两个最小项的函数标号交集非空时才能进行合并，并且合并后的函数标号为二者共有部分；最小项表中的项，所有符号都应该被处理，一个被处理当且仅当它所有的函数符号出现在一个合并项之中。

Q-M方法中的最小覆盖问题是NP完全问题！

8. 锁存器和触发器

时序电路

组合电路：输入$x=(x_1,\cdots,x_n)$，输出$z=(z_1,\cdots,z_m)$，组合电路完成如下映射。

$$z_i=f_i(x_1,\cdots,x_n),\ i=1,\cdots,m$$

时序电路：输入和出和组合电路相同，引入状态，包括现态$y=(y_1,\cdots,y_r)$和次态$Y=(Y_1,\cdots,Y_r)$，时序电路完成如下映射。

$$z_i = g_i(x_1, \cdots, x_n, y_1, \cdots, y_r),\ i = 1, \cdots, m$$
$$Y_i = h_i(x_1, \cdots, x_n, y_1, \cdots, y_r),\ i = 1, \cdots, m$$

状态图：圆表示状态，有向箭头表示状态转换，线上的标注x/z表示产生状态变换的输入x和对应输出z。

状态表：每一行对应一个状态，每一列对应一个输入，表中单元Y/z对应在该状态下接受x输入，会转移至Y状态，并产生输出z。

存储元件

二进制存储：使用双稳态电子线路，状态0表示存储逻辑0，状态1表示存储逻辑1。输出Q指示了存储单元的现态。

激励输入：每个存储元件有多个输入；能激励或者驱动存储元件进入确定的状态的输入称作激励输入。

锁存器：锁存器的激励输入控制元件的状态。**锁存器立即响应激励输入。**

置位锁存器：激励输入强制元件输出为1。

复位锁存器：激励输入强制元件输出为0。

置位复位锁存器：同时具有置位和复位激励信号的存储元件。

触发器：需要时钟控制信号；时钟信号向触发器发命令，触发器根据激励信号改变状态；在多触发器电路中，时钟信号使得所有触发器同步地改变状态。**触发器依赖时钟响应激励输入。**

锁存器

R-S锁存器：使用两个NOR或两个NAND实现。输入的R为Reset，S为待锁存数据，输出为Q和$\bar{Q}$。R和S不能同时为1，否则会使得Q不稳定。

$$Q(t+\Delta) = S + \bar{R}\,Q(t)$$

恢复时间t_{rec}：复位和置位有效信号间的最小时间。

R-S锁存器的激励输入限制：R和S不能同时无效，否则产生信号追逐，输出产生震荡，但最终必定有一个门获胜，使得锁存器达到稳态，但是不能确定输出的结果。S和R的有效脉冲不能太短，脉冲宽度必须大于t_{rec}。

门控R-S锁存器：在R-S锁存器之前，加入门控信号，用于控制真正进入R-S锁存器的R和S信号是否有效。

$$Q(t+\Delta) = SC + \bar{C}\,Q(t) + \bar{R}\,Q(t)$$

时钟：可以用来保证R-S触发器的输入信号有足够长的稳定时间。

时钟周期：两个相同跳变的时间间隔。

占空比：高电平所占时钟周期的比例。

时钟控制的R-S锁存器：将CLK设置为门控信号EN，只有一半时钟周期用来传播信号，另一半周期用来保持输出状态不变。

触发器

主从触发器：将锁存器级联，一个锁存器的输出连接另一个锁存器的输入；用时钟信号的两个电平分别控制两个锁存器。激励信号仅仅在时钟信号边沿有效，输出在一个时钟周期内保持稳定。

1捕获问题：（或0捕获问题）第一级RS锁存器在时钟电平高时存在0静态冒险，必须限制输入逻辑来消除冒险。

主从D触发器：通过设置R为$\bar{D}$，S为D，使得S和R互补，从而消除1捕获问题。必须保证D在时钟边沿之前是要储存的值。

$$Q(t+\Delta)=D$$

建立时间t_{setup}，在EN变化之前，激励信号必须保持的时间。

保持时间t_{hold}，在EN变化之后，激励信号必须保持的时间。

最小脉冲宽度t_w，为了保证状态稳定，不出现震荡或不确定的状态，时钟信号需要的最小脉冲宽度。

J-K触发器：S-R锁存器中，S和R不能同时为1。J-K触发器中J相当于S，K相当于R，但在J和K同时为1时，J-K触发器发生状态翻转。

$$Q(t+\Delta)=\bar{K}\,Q(t)+J\,\bar{Q}(t)$$

边沿触发器：只需要适中的上升沿或者下降沿。

上升沿触发器：时钟信号0->1跳变。

下降沿触发器：时钟信号1->0跳变。

异步信号：时序电路中不需要时钟参与就可以改变电路状态的激励信号，如CLR和PRE。

同步信号：时序电路中必须在时钟参与的情况下才能改变电路状态的激励信号，如D。

T触发器：J-K触发器的基础之上，加入PRE和CLR异步控制信号，并将J和K直接置为1（接Vcc）。T触发器每个时钟周期翻转一次状态。

$$Q(t+T)=\bar{Q}(t)$$

钟控T触发器：同T触发器类似，将J和K共同接T。T决定触发器保持（T为0）还是翻转（T为1）。

$$Q(t+T)=\bar{T}\,Q(t)+T\,\bar{Q}(t)$$

定时和同步

定时的目的：确保系统在最严苛的情况下，也可以正常运行。

正确定时的基本规则：触发器的正确输入的正确时间；触发器不可能在同一个时钟沿变化两次。

时序参数：建立时间t_{setup}，保持时间t_{hold}，最小时钟宽度t_w，传播时间$t_{PropDelay}$。传播时间为从CLK跳变时刻起到Q变化时刻经历的时间。

时钟扭斜：两个级联元件接受到的时钟存在偏移，不能完全对齐，偏移量即为时钟扭斜t_{skew}。

时序电路定时要求

$$T_{PropDelay}+T_p>T_{hold}+T_{skew}$$
$$T_{clock}+T_{skew}>T_{PropDelay}+T_p+T_{setup}$$

亚稳态：锁存器和触发器都存在两个稳态，即逻辑0和逻辑1；亚稳态时两个稳态之外的第三个平衡状态，亚稳态会由于噪声的介入，在长时间内变为稳态，但并不能确定稳态类型。

亚稳态的危害：不同的门对相同的亚稳态信号的解释不一致。

异步输入的危害：在同步电路中使用异步输入，输入可能会在任何时刻发生变化，很容易引起建立/保持时序违例（setup/hold），进入亚稳态。但异步输入不可能避免，即同步失败的概率不可能减少到0，但可以尽量减小。

同步失败的处理方案：减慢系统时钟，使之满足时序要求（妥协方法）；使用更好的逻辑工艺；**级联两个触发器作为同步器**（非常有效的方案）。

$$\text{异步输入} \Rightarrow \text{触发器}1 \Rightarrow \text{触发器}2 \Rightarrow \text{同步输入}$$

一个异步信号输入一定不可以驱动多于一个触发器，若要驱动则先加入一个同步器（触发器）进行同步。

9. 寄存器和计数器

寄存器

寄存器：共用相同控制和逻辑的触发器集合，被储存的值具有一定的关联性，共用CLK，RESET和SET线。

移位寄存器：保存输入的采样值。例如4bit移位寄存器保存输入序列的最后4个输入值。

通用的移位寄存器：允许串行或并行的输入或输出，允许选择左移或右移，确定移入新值的方向是从左还是从右。

4bit移位寄存器：4bit输入（input），4bit输出（output），时钟（CLK），清零（clear），移位功能选择（s0和s1），左移输入输出（left_in和left_out），右移输入输出（right_in和right_out）

clear	s0	s1	动作
1	-	-	清零
0	0	0	保持原值
0	0	1	右移
0	1	0	左移
0	1	1	载入新值

串并转换：一次性输入或输出为并行通信，移位逐位输入或输出为串行通信。

串行加法器：一个全加器用于按位加，两个N位移位寄存器用于存储加数串，一个进位触发器用于存储进位，一个N为移位寄存器用于存储计算结果。

特定模式串识别器：在移位寄存器的输出连接组合电路用于识别，移位寄存器逐位读入序列进行识别。

特殊序列计数器：产生固定模式序列的串行输出。

Mobius计数器：产生如下序列1000,1100,1110,1111,0111,0011,0001,0000,1000,...。使用移位寄存器进行右移，右移输出通过非门连接右移输入即可。

计数器

计数器：用于记录输入脉冲的个数。控制信号有同步/异步的清零信号clear，使能信号enable，同步/异步装载load。

同步二进制计数器：由n个同步二进制计数器组成。包括n个钟控J-K触发器，其初始状态全部是0，共有2^n个状态，状态序列为$0,1,2,\cdots,2^{n-1},0,1,\cdots$，并且在$2^{n-1}\rightarrow 0$时会有溢出信号产生。

4位同步计数器：上升沿触发同步装载LOAD和清除CLR；并行装载数据DCBA；是能输入EN；RCO为行波进位输出，可以用于级联计数器，当输出计数为1111时，RCO为1，其余时刻为0。

模N计数器：需要计数器和从N-1到0的翻转，因此需要设计判断N-1的逻辑电路，判断翻转成立时，置CLR有效，使得计数翻转到0。典型例子是BCD计数器。

起始偏移计数器：固定DCBA为偏移值，RCO接到LOAD，产生的计数序列为0110,0111,1000,1001,1010,1011,1100,1101,1111,0110,...。起始偏移计数器确定了计数器的起始值。

截止偏移计数器：在输出进行值判断（利用非门和与门）判断结果接到CLR，产生的计数序列为0000,0001,0010,...,1001,0000,...。截止偏移计数器确定了计数器的终止值。

二者结合即可完成任意计数。

分频计数器：使用膜N计数器，截止判断的结果不仅接到CLR，同时作为信号输出，完成分频。

Verilog实现

敏感向量表：always @()，括号中即为敏感向量，敏感向量变化将引起always模块中的动作。

always @(posedge clk)时钟上升沿触发。always @(negedge clk)时钟下降沿触发。

同步和异步：同步线程中，使用一个线程等待时钟事件；异步线程中存在很多并行线程，但只有其中的一个等待时钟事件。

阻塞赋值：X=A，再继续下一条语句之前完成本句赋值。

非阻塞赋值：又称作寄存器传输级赋值RTL，X<=A，在always模块完成之前不完成赋值，always模块完成时统一进行赋值。

10. FSM

时序电路和FSM

时序电路：由基本时序单元（锁存器、触发器、寄存器）组成的电路。

有限状态机FSM：描述时序电路的模型。

同步时序电路：时序电路中的状态的改变由系统的统一时钟进行控制。

异步时序电路：时序电路中的状态的改变不受统一时钟的控制，而是由输入变化直接引起其改变。

FSM组成：状态state，由时序存储元件的可能取值确定。转换transition，描述状态的改变。时钟clock，控制状态元件改变状态的控制信号。

任何时序电路都可以表示为FSM和状态图。

时序电路->状态图：时序电路中所有可能的存储组合即为状态，在状态A接受输入x转移到状态B并产生输出z唯一确定，因此可以确定状态图的边。

状态图->时序电路：使用适量触发器保存状态，用逻辑电路计算次态，时钟信号控制何时触发器更新其存储值。

FSM设计流程

状态图到状态转换表：指定了当前状态和输入与下一状态（次态）的对应关系；需要对状态进行唯一编码。电路实现上，每个状态位对应一个触发器，使用编码设计组合电路即可。

无关状态：由于会有无用的或无意义的状态码，因此会存在无关的状态项。

自启动：在上电时，电路可能处于无用或无效的状态之中，必须使其进入到正确的状态。自启动的解决方案是使得每个无效状态都可以进入到有效状态之中，而有效状态不会再返回无效状态，通常采用全局的复位信号。

一般的状态机模型：存在寄存器中的值表示电路的状态，通过组合电路计算次态和输出。次态为现态和输入的函数；输出为现态和输入的函数（Mealy机）或现态的函数（Moore机）。

状态State：$S_1, S_2, \cdots, S_k$

输入Input：$I_1, I_2, \cdots, I_m$

输出Output：$O_1, O_2, \cdots, O_n$

转换函数Transition function：$F_S(S_i, I_j)$

输出函数Output function：$F_o(S_i)$或$F_o(S_i, I_j)$

Moore机：输出函数为$F_o(S_i)$，Moore机更加安全。

$$
\begin{aligned}
&Inputs + Current\ state \xrightarrow{\text{组合电路 计算次态}} Next\ state \xrightarrow{\text{存入}} \text{寄存器} \\
&\qquad\qquad \Uparrow \\
&\text{寄存器} \xrightarrow{\text{读出}} Current\ state \xrightarrow{\text{组合电路 计算输出}} Outputs
\end{aligned}
$$

Mealy机：输出函数为$F_o(S_i, I_j)$，Mealy机状态数目少，对输入的反应更快。

$$
\begin{aligned}
&Inputs + Current\ state \xrightarrow{\text{组合电路 计算次态}} Next\ state \xrightarrow{\text{存入}} \text{寄存器} \\
&Inputs + Current\ state \xrightarrow{\text{组合电路 计算输出}} Outputs \\
&\text{寄存器} \xrightarrow{\text{读出}} Current\ state
\end{aligned}
$$

同步Mealy机：输出函数为Mealy机输出函数，但是输出至寄存器，同步后在下一个周期输出。

$$
\begin{aligned}
&Inputs + Current\ state \xrightarrow{\text{组合电路 计算次态}} Next\ state \xrightarrow{\text{存入}} \text{寄存器} \\
&Inputs + Current\ state \xrightarrow{\text{组合电路 计算输出}} Outputs \xrightarrow{\text{存入}} \text{寄存器} \\
&\text{寄存器} \xrightarrow{\text{读出}} Current\ state \\
&\text{寄存器} \xrightarrow{\text{读出}} Output \xrightarrow{\text{输出}} Output
\end{aligned}
$$

RESET：引导电路在上电之后进入确定的初始状态。同步时序电路中，在上电后产生一定时间宽度的RESET信号，采用异步复位和置位的方式，将电路设置为初始状态；当RESET信号无效之后，系统进入正常工作的状态。

11. FSM设计

FSM设计流程

设计流程

```
1. 理解问题，理解输入/输出行为，并对问题进行提炼
2. 使用Moore机或Mealy机画状态图，首先确定状态数目，之后添加跳转的边，要求每个状态对每个输入都能跳转到正确的次态
3. 将状态图转换为状态转换表，根据状态表确定各个信号的输出
4. 实现组合逻辑电路
```

Moore机状态图：边上标号表示在对应输入进行该跳转，状态的圈中的[]内标识该状态的对应输出。

Mealy机状态图：边上标号x/y表示在接收x输入时进行跳转，并且产生输出y。

Moore机的Verilog实现：input包括CLK，RESET和输入信号IN，output只有输出信号OUT，内部具有STATE和NEXT_STATE变量（reg）。具有三个alway模块。

```
1. 状态转换模块
    always @(posedge CLK)
        if (RESET) STATE = __INIT_STATE__;  // 初始化
        else STATE = NEXT_STATE;    // 状态转换
2. 产生次态模块
    always @(IN or STATE)
        case (STATE) ... endcase // 根据现态和输入产生不同的次态
3. 产生输出模块
    always @(STATE)
        case (STATE) ... endcase // 根据现态产生不同的输出
```

Mealy机的Verilog实现：input和output及状态定义均同Moore机。具有两个always模块。

```
1. 状态转换模块，同Moore机
2. 产生次态和输出模块
    always @(IN or STATE)
        case (STATE) ... endcase // 根据现态和输入产生不同的次态和输出
```

同步Mealy机的Verilog实现：input和output及状态定义同上。只有一个always模块。

```
always @(posedge CLK)
    if (RESET) STATE = __INIT__STATE__;
    else
        case (STATE) ... endcase // 根据现态和输入产生不同的次态和输出
```

Moore机的电路实现：寄存器存储状态，寄存器输出Q为现态，输入D为次态，时钟信号连接C，RESET信号连接R。使用SOP或POS组合电路接收Q和输入并产生D，用于状态转换；使用SOP或POS组合电路接收Q并产生输出。

Mealy机的电路实现：寄存器存储状态。使用组合电路接收Q和输入并产生D，用于状态转换；使用组合电路接收Q和输入，产生输出，该输出与REST信号作与确保初始化时的输出，最终用于输出。

Moore机到同步Mealy机的实现：Moore机电路，将输出的组合电路移至接收寄存器的输入信号D，之后组合电路的输出进入寄存器，同步后进行输出。

Mealy机到同步Mealy机的实现：Mealy机电路，原输出直接进入寄存器，同步后进行输出。

FSM通讯

FSM通讯：一个状态机的输出，作为另一个状态机的输入。

数字硬件系统：数据通路+控制电路。

数据通路：寄存器，计数器，组合运算逻辑（ALU），通信总线（BUS）等。

控制电路：FSM产生的控制信号序列，用于控制数据通路的行为。

ASM

算法形状态机ASM：使用算法框图，表示的FSM。

状态框：用于表示状态名称，和Moore机的输出。用矩形表示，多/单入边，单出边。

决策框：用于表示状态的变换条件，决策框只处理一bit输入的判断。用菱形表示，单入边，两出边（0/1），输入多于1bit时需要级联决策框。

条件输出框：用于表示Mealy机的输出。用圆角矩形表示，单入边，单出边。

状态分配：二进制编码，和One-hot编码。二进制编码的ASM需要使用决策框；One-hot编码的ASM不使用决策框，其具有一条链的主干。

12. FSM分析

冗余状态

状态等价性

定义1：完全确定的时序电路中的状态$S_1,\cdots,S_j$等价，当且仅当对于任意的输入序列，以$S_1,\cdots,S_j$中的任意状态作为初始状态，电路的输出序列完全相同。

定义2：设S_i和S_j是完全确定的时序电路的两个状态，S_k和S_l是在输入为I_p时的S_i和S_j的下一个状态，S_i和S_j等价，当且仅当对于每一个可能的I_p，S_i和S_j输出相同且S_k和S_l等价。

等价关系：设R是集合S上定义的关系，R是等 价的当且仅当R是自反的，对称的，传递的。一个集合的等价关系可以将集合划分为不相交的等价类。

等价和等价关系：时序电路的状态等价时状态集合的一个等价关系。

定理：时序电路状态等价定义出的等价类可以用来表示等价电路中的状态。

完全确定时序电路的化简

观察法：两个状态等价，则他们在相同输入输出的前提下，下一个状态相同或下一个状态是这两个状态之一。

观察法很难发现一条链环的等价关系。

划分法：一组连续的过程，每一步形成的划分P_k，由一些块组成，每个块中的状态都是K-等价的（两个状态接收K个输入序列，输出序列相同）。

```
1. 将输出相同的状态放在一个块中，形成1-等价划分
2. 对于P(k-1)中每个块，将其进行划分，使得每个状态都落在P(k-1)中的相同的块内，合并这些划分得到P(k)
3. 重复直到P(k)=P(k-1)，得到等价类
```

蕴含表法：蕴含表是一个平面表，其行列都对应状态，且行缺少第一个状态，列缺少最后一个状态，元素只能写在对角线及其下。

1．画空的蕴含表，表中单元表示所有可能的状态对
2．检查蕴含表每个单元，当对应的两个状态的输出不同时，在方块内画x
3．将输出相同的次态对填入表格单元。若蕴含项等于相应的两状态，或者是同一状态时，则画√，当单元内包含的所有蕴含对都变为√时，则对应的单元化画√
4．处理蕴含表，确定每个状态对的等价性。若蕴含单元包含一个为x的蕴含项，则该单元也画x
5．蕴含表得到等价状态，到处等价划分，没有x的单元对应的状态是等价对

非完全确定自动机的化简

指定取值后再进行化简