

# 《数字系统设计》

立

## Chapter 1 数制与码制

### 1、常用数制与转换

	转换	例
N进制→十进制	$(k_n k_{n-1} \dots k_0)_N = (\sum_0^n k_i N^i)_{10}$	$(101)_2 = (1 * 1 + 0 * 2 + 1 * 4)_{10} = (5)_{10}$
十进制→N进制	整数部分短除法，小数部分乘N向下取整	$(173.8125)_{10} = (10101101.1101)_2$
二进制↔十六进制	4位等分	$(0101.1011)_2 = (5.B)_{16}$ $(8.C)_{16} = (1000.1100)_2$
二进制↔八进制	3位等分，方法同上	/

### 2、二进制算术运算

首位为符号位（正数为0，负数为1）

	定义	6	-6
原码	符号位+二进制数	0110	1110
反码	符号位+正数不变，负数0、1互换	0110	1001
补码	符号位+正数不变，负数为反码末位加1	0110	1010

（1000表示-8的补码）

### 3、几种常用编码

十进制代码

（1）**8421BCD码**：恒权代码，每一个二进制代码的1都代表一个固定数值。

(2) 余3码：对应十进制数加3的二进制码。非恒权代码，两个相加为9的十进制数的代码互为反码。

(3) 2421码：恒权代码，两个相加为9的十进制数的代码互为反码。

格雷码

右数第N位数以  $\underbrace{00\dots00}_{2^{N-1}\uparrow 0} \underbrace{11\dots11}_{2^N\uparrow 1} \underbrace{00\dots00}_{2^{N-1}\uparrow 0}$  的顺序循环变化（第4位只有半个循环）

优点：相邻代码只有一位发生变化，不产生过渡噪声。

## Chapter 2 逻辑代数基础

### 1、基本运算

	算式	门
与	$A \cdot B$	&
或	$A + B$	$\geq 1$
非	$\bar{A}$	1
异或	$A \oplus B$	$\neq 1$
同或	$A \odot B$	=

### 2、常用公式

$$(1) A + BC = (A + B)(A + C) \rightarrow A + \bar{A}B = A + B$$

$$(2) AB + \bar{A}C + BC = AB + \bar{A}C$$

### 3、基本定理

(1) 代入定理：在任何一个包含变量 A 的逻辑等式中，若以另外一个逻辑式代入式中所有 A 的位置，则等式仍然成立。

(2) 反演定理：取反时，与或互换，0 1 互换，原反互换，由外而内。

### 4、逻辑函数标准形式

最小项

$$Y(A, B, C, D) = \sum m(m_1, m_2, \dots, m_n) \quad m \text{ 为乘积项}$$

所有最小项之和为 1，任意两个最小项之积为 0

最大项

$$Y(A, B, C, D) = \prod M(M_1, M_2, \dots, M_n) \quad M \text{ 为和项}$$

所有最大项之积为 0，任意两个最大项之和为 1

最小项与最大项的关系

$$\sum m_i = \prod M_k \quad i \neq k \quad \text{即互为对偶式}$$

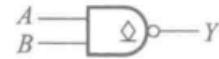
## Chapter 3 门电路

### 1、CMOS 门电路

无论 G 端接多大电阻，电平始终与接入处状态相同，且无法悬空。

漏极开路输出门电路（OD 门）

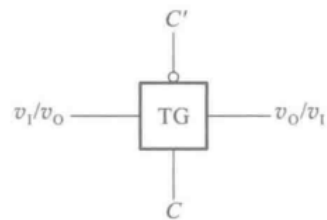
满足输出电平变换、吸收大负载电流与实现线与连接需要



传输门

$C = 1, \bar{C} = 0$  通

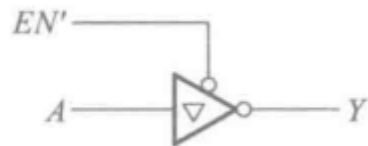
$C = 0, \bar{C} = 1$  截止



三态门

$EN = 1 \quad Y = \bar{A}$

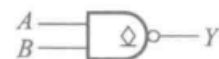
$EN = 0 \quad Y = Z(\text{高阻态})$



### 2、TTL 门电路

漏极开路输出门电路（OC 门）

满足输出电平变换、吸收大负载电流与实现线与连接需要



三态门

同 CMOS 三态门

## Chapter 4 组合逻辑

### 1、编码器

十进制转二进制

普通编码器

任何时刻只允许输入一个编码信号

优先编码器

允许输入多个编码信号，但只对优先级最高的一个进行编码（ $7 > 6 > \dots > 0$ ）

S'（选通信号）	Y <sub>S</sub> '（附加输出信号）	Y <sub>EX</sub> '（附加输出信号）	状态	芯片图
1	1	1	不工作	
0	1	0	工作，有输入	
	0	1	工作，无输入	

2、译码器

十进制转十进制，即输出最小项  $\bar{Y}_i = \bar{m}_i$

二进制译码器

片选输入端			状态	芯片图
S <sub>1</sub>	S <sub>2</sub> '	S <sub>3</sub> '		
1	0	0	正常工作	
反之			输出Y <sub>i</sub> '均为高电平	

二—十进制译码器

拒绝了伪码 1010 – 1111

3、数据选择器

双四选一数据选择器

$$Y_i = (D_{i0} \bar{A}_1 \bar{A}_0 + D_{i1} \bar{A}_1 A_0 + D_{i2} A_1 \bar{A}_0 + D_{i3} A_1 A_0) \cdot S_i$$

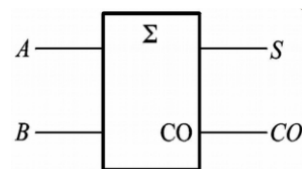
S <sub>i</sub> '	状态	芯片图
0	正常工作	
1	输出Y <sub>i</sub> =0	

4、一位加法器

S 为输出，CO 为向高位的进位，CI 为来自低位的进位

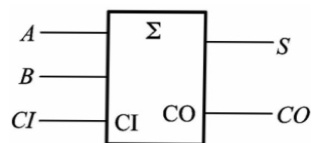
半加器

不考虑  $CI$  
$$\begin{cases} S = A \oplus B \\ CO = A \cdot B \end{cases}$$



全加器

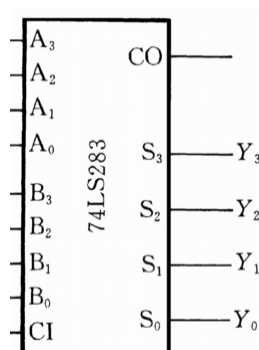
考虑  $CI$



## 5、多位加法器

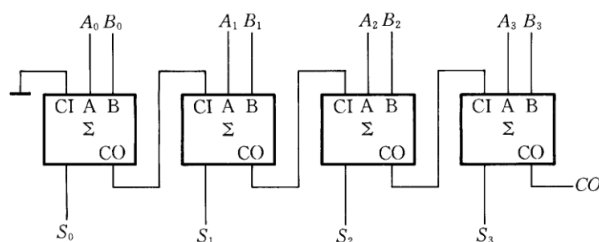
超前进位加法器

快但复杂。



串行进位加法器

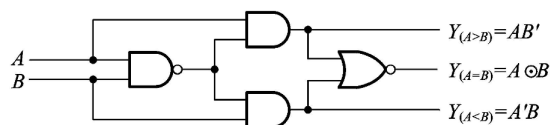
慢但简单。



## 6、数值比较器

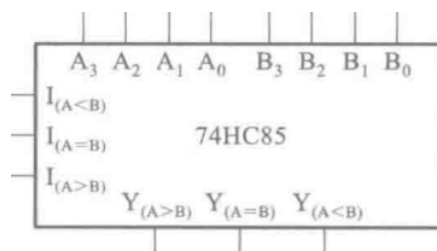
一位数值比较器

用来比较两个一位二进制数的大小。



多位数值比较器

从高位比起，只有高位相等，才比较下一位。



$I$  为来自低位的比较结果。

## 7、组合逻辑中的竞争-冒险现象

成因

门电路的两个输入同时向相反的逻辑电平变化，称为“竞争”。

因“竞争”而可能在输出端产生尖峰脉冲的现象，称为“竞争-冒险”

判断

输出为  $Y = A + \bar{A}$  或  $Y = A \cdot \bar{A}$  型

消除方法

(1) 接入滤波电容

(2) 引入选通脉冲

(3) 添加冗余项：如  $Y = AB + \bar{A}C$ ，当  $B = C = 1$  时， $Y = A + \bar{A}$ 。稳态下  $Y = 1$  且存在竞争-冒险现象。

则添加冗余项  $BC$ （即在  $B = C = 1$  时使得  $Y$  恒为 1），此时  $Y = AB + \bar{A}C + BC$  即可消除。

## 8、Verilog HDL

基本结构

```
module<模块名>(<端口列表>);  
    参数定义（可选）；  
    端口说明（input,output,inout）；  
    数据类型定义  
    持续赋值语句  
    过程块（always/initial）--行为描述语句  
    底层模块实例--调用底层模块  
    任务和函数  
    延时说明块  
  
endmodule
```

定义

```
input,output,inout  定义端口      // input[6:0]A → input A[0] A[1]...a[6]  
assign              端口赋值      // assign x=A?B:C  
wire,reg            定义连接点    // wire:实际连接点，用于定义assign类型或门原语  
                                // reg:虚拟连接点，用于always过程块  
always@(<端口名>)    // 当<端口名>改变时，会发生begin之后的动作  
begin  
    //等式左端为reg类型变量  
end
```

门原语

```
<门原语> <门名称> (<端口名>)    // 端口名为输出在前，输入在后  
and,nand,or,nor,xor,xnor        // 与，与非，或，或非，异或，同或
```

位运算符

取反	按位与	按位或	按位异或	按位同或
~	&		^	^~

## 位拼接运算符

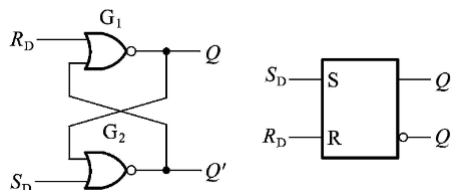
即把某些信号的某些位详细地列出来，中间用逗号分开，最后用大括号来表示一个整体信号。

如：  $\{a, b[3:0], w, 3'b101\} = \{a, b[3], b[2], b[1], b[0], w, 1'b1, 1'b0, 1'b1\}$   
  $\{b, \{3\{a, b\}\}\} = \{b, a, b, a, b, a, b\}$

# Chapter 5 半导体储存电路

## 1、SR锁存器

$S_D$ ：置1输入端；  $R_D$ ：置0输入端。



特点：在任何时刻，输入都能直接改变输出的状态，即使输入信号消失，输出也保持。但需满足  $S_D \cdot R_D = 0$  。

## 2、触发器

		特点	次态表达式	电路图
电平触发	SR触发器	CLK=1时，输入端的任何变化都会引起输出状态的变化	$Q^* = S + R'Q$	
	D触发器		$Q^* = D$	
脉冲触发	主从SR触发器	CLK=1时，“主”翻转，“从”不变 CLK↓时，“从”翻转，“主”不变	$Q^* = S + R'Q$	
	主从JK触发器	CLK=1期间，“主”只允许翻转一次	$Q^* = JQ' + K'Q$	
边沿触发	/	触发器的次态仅取决于CLK上升或下降沿到达时刻输入信号的状态，而与之之前之后状态无关	/	/

触发器按逻辑功能分类

触发器	特性方程
$SR$ 触发器	$Q^* = S + \bar{R}Q$
$JK$ 触发器	$Q^* = J\bar{Q} + \bar{K}Q$
$T$ 触发器	$Q^* = T\bar{Q} + \bar{T}Q$
$D$ 触发器	$Q^* = D$

## 2、随机存储器

### 静态随机存储器SRAM

由存储矩阵、地址译码器、读写控制电路组成。容量=“（字数）×（每个字的位数）”。

### 动态随机存储器DRAM

利用了MOS电容可以存储电荷的原理

## 3、只读存储器ROM

由存储矩阵、地址译码器、输出缓冲器组成。字线与位线每一个交点都是一个存储单元，接二极管：存1；没接：存0。

### 掩模ROM

出厂时数据已固化，适合大量生产

### 可编程PROM

只能写入一次（写入时将0的存储单元熔丝烧断），无法修改

### 可擦除的可编程PROM

雪崩写入，紫外线擦除

## 4、存储器容量的扩展

扩展方式	接法	电路图
位扩展	将各片的地址线、读写线、片选线并联	
字扩展	将 $A_9$ 、 $A_8$ 译成 $Y'_0 - Y'_3$ 分别接4片的 $CS'$	

地址数=字数= $2^{\text{地址线数}}$

位数=读写线/数据线数

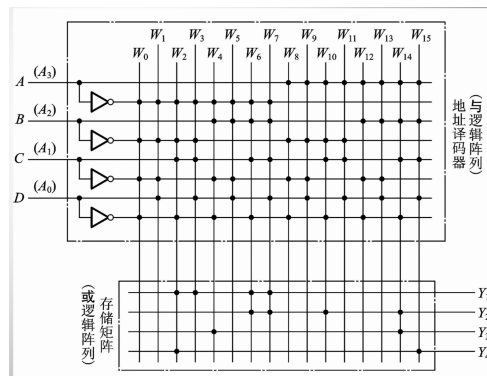


$$Y_1 = \sum m(2, 3, 6, 7)$$

$$Y_2 = \sum m(6, 7, 10, 14)$$

$$Y_3 = \sum m(4, 14)$$

$$Y_4 = \sum m(2, 15)$$



## Chapter 6 时序电路

### 1、移位寄存器

4位双向移位寄存器

$D_3 - D_0$ : 数据输入端  $D_{IR}$ : 右移串行输入  $D_{IL}$ : 左移串行输入  $Q_3 - Q_0$ : 输出端

$R_D'$	$S_1$	$S_0$	工作状态	芯片图
0	/	/	置零	
1	0	0	保持	
	1	0	左移	
	0	1	右移	
	1	1	并行输入	

### 2、同步计数器

同步二进制加法计数器

$D_3 - D_0$ : 数据输入端（预置数端）  $Q_3 - Q_0$ : 输出端

0000  $\rightarrow$  0001  $\rightarrow \dots \rightarrow$  1111  $\rightarrow$  0000(同时  $C = 1$ )  $\rightarrow \dots$

CLK	$R_D'$	$LD'$	EP	ET	工作状态	芯片图
/	0	/	/	/	置零	
$\uparrow$	1	0	/	/	预置数 $D_3 \sim D_0$ (同步)	
/		1	0	1	保持 (包括 C)	
/		1	/	0	保持 ( $C=0$ )	
$\uparrow$		1	1	1	计数	

### 3、异步计数器

	特性	电路图
二进制加法计数器	在 $n$ 位 1 到 0 跳变时, $n+1$ 位翻转。	
二进制减法计数器	在 $n$ 位 0 到 1 跳变时, $n+1$ 位翻转。	
十进制加法计数器	跳过 1010~1111 四个状态。	

### 4、任意进制计数器的构成方法 ( $N$ 进制 $\rightarrow M$ 进制)

$N > M$

设法跳过  $N - M$  个状态

(1) 置零: 进入  $S_M$  状态后利用置零端  $R'_D$  立即进入  $S_0$  状态

(2) 置数: 任意状态均可利用置数端  $L'_D$

$M > N$

若  $M$  能拆成两个数  $N_1$ 、 $N_2$  的乘积, 则可接成  $N_1$  进制与  $N_2$  进制计数器。

(1) 并行进位: 同一时间信号  $CLK$ , 低位输出作为高位的计数控制信号。

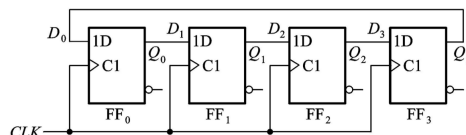
(2) 串行进位: 低位输出作为高位的时间信号  $CLK$ , 同时处于计数状态。

若  $M$  为质数, 则寻找一  $M' = N'_1 \times N'_2$  ( $M' > M$ ), 接成  $N'_1$  进制与  $N'_2$  进制计数器, 进行置零或置位。

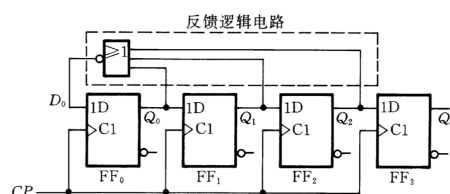
### 5、移位寄存器计数器

环形计数器

实现数据循环右移:  $1000 \rightarrow 0100 \rightarrow 0010 \rightarrow 0001 \rightarrow 1000 \rightarrow \dots$

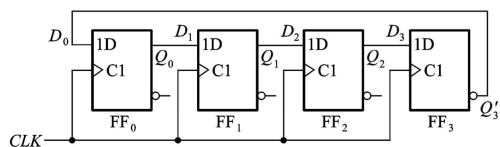


能自启动的:

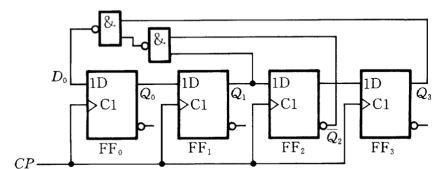


## 扭环形计数器

0000  $\rightarrow$  1000  $\rightarrow$  1100  $\rightarrow$  1110  $\rightarrow$  1111  $\rightarrow$  0111  $\rightarrow$  0011  $\rightarrow$  0001  $\rightarrow$  0000  $\rightarrow$  ...



能自启动的:



## Chapter 7 控制器设计

### 1、状态机

状态机的类型

米里状态机：输出与当前状态和输入信号有关。

莫尔状态机：输出仅与当前状态有关。

算法状态机 ASM

	符号
状态框	
判断框	
条件框	

### 2、控制器

(1) 设置状态变量

(2) 画出 ASM 图

(3) 列出状态编码的现态 ( $PS$ ) 与次态 ( $NS$ ) 真值表

(4) 写出控制信号关于状态编码的表达式

(5) 画出电路图

#### 计数器型控制器

按给定的  $ASM$  流程图构造一个状态发生电路, 使它具有  $ASM$  中所需的全部状态, 并能依照控制算法条件进行状态转移和条件输出。

$n$  个触发器最多可代表  $2^n$  个状态。将所要求的控制状态按一定原则进行编码分配, 就可设计出一种状态计数型的控制器, 通常称之为计数器型控制器。

通过状态转移表求得次态激励方程, 利用  $D$  触发器/ $JK$  触发器输出控制信号, 画出电路图。

#### 多路选择器型控制器

按控制算法要求, 为其对应的触发器生成次态激励函数。

状态转移数据表比通常的状态转移表多了一栏: 转换条件。转换条件栏的数据或表达式, 也就是对应的多路选择器的数据输入值。

通过状态转移表求得  $MUX$  的数据输入端, 用状态编码现态  $PS$  和转换条件  $C$  作为  $MUX$  的地址输入, 最后利用  $D$  触发器输出控制信号, 画出电路图。

#### 定序型控制器

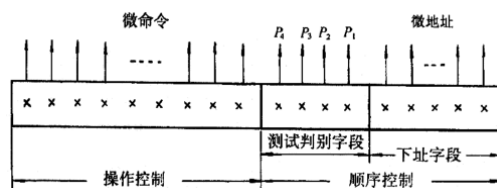
一对一: 触发器的数目代表了状态数, 并依赖一组最新的代码实现状态转换。

通过状态转移表, 可按照  $NS = \sum PS \cdot C$  求得次态激励方程, 利用  $D$  触发器输出控制信号, 画出电路图。

### 3、微码控制器

#### 微指令

微指令除给出微命令信息外, 还应给出测试判别信息。一旦出现此信息, 执行这条微指令时要对系统的有关“状态标志”进行测试, 从而实现控制算法流程图的条件分支。微指令中还包含一个下址字段, 该字段将指明  $ROM$  中下一条微指令的地址。



长条框内的符号  $\times$  表示一个二进制位 ( $bit$ )。其中微命令字段用于操作控制;  $\times$  编码为 1 时表示有微命令,  $\times$  编码为 0 时表示无微命令。测试判别字段和下址字段一起实现顺序控制: 当测试判别字段无效时 ( $\times$  编码为 0), 下址字段信息即是下条微指令的地址; 当测试判别字段有效时 ( $\times$  编码为 1, 可以有多个测试), 根据反馈线来的“状态”信息对下址字段信息进行修改, 修改后的地址即为下条微指令的地址。

#### 微程序控制器

主要由控制存储器、微地址寄存器、微码命令寄存器和地址转移逻辑等部分组成。微地址寄存器和微命令寄存器两者的总长度即为一条微指令的长度, 所以两者合在一起称为微指令寄存器。为便于说明形成下一条微指令的地址, 特意将微指令的下址字段独立出来, 称为微地址寄存器。

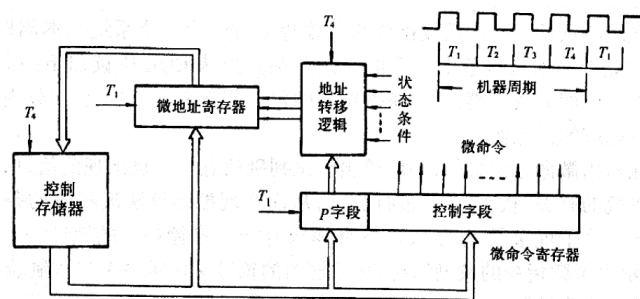


图 8.32 微程序控制器结构框图

$n$  位微地址寄存器长度对应  $2^n$  条微指令， $n$  位微指令对应  $n$  位控制存储器  $ROM$  字长。

## 4、算法与流水结构

设数据流中元素个数为  $n$  运算段个数为  $L$  每段运算时间为  $\Delta t$

顺序（串行）算法结构

在执行算法的整个过程中，同一时间只进行一种或一组相关的子运算。

$$T_s = n \cdot L \cdot \Delta t$$

并行算法结构

指在同一时间段中，有多条路径在同时进行运算，在这些同时执行的子运算操作之间是相互独立的。

$$T_p = n \cdot L' \cdot \Delta t \quad (L' \ll L, \text{ 因此加快了速度})$$

流水线算法结构

把整个运算过程分解为若干个段，系统在同一时间可对先后输入的数据流元素进行不同段的同时运算。

$$T = L \cdot \Delta t + (n - 1)\Delta t$$

# Chapter 8 验证与测试

## 1、测试过程

首先把预先确定的测试矢量装入能够向被测器件（*device under test*,  $DUT$ ）提供激励并采集相应响应的测试设备。测试矢量由测试程序来定义，它描述了所应用的波形、电平、时钟频率以及所期望得到的响应。需要用一个探针卡或  $DUT$  板把测试仪的输入和输出连到芯片或封装相应的引线上。

生产测试可按测试目的分为以下几类：

测试类型	测试目的
诊断测试	用在芯片和板级调试期间，其目的是对于一个给定的失效部件识别和指出失效的部位。
功能测试	确定一个制造出的元件是否能工作。每一个制造出来的芯片都要经过这一测试，直接影响芯片成本，应当尽可能简单快速。
参数测试	在各种工作条件（如温度和电源电压）下检查许多非离散参数，一般分为静态和动态测试。

## 2、可测性设计

组合电路属于易观察和可控制的电路。时序电路的可测试设计方法划分成 3 类：

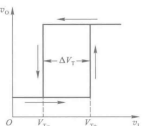
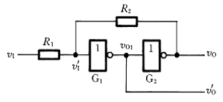
测试类型	特点
专门测试	集合了一些可用来提高一个设计的可观察性和可控性的技术，同应用类型相关。
扫描测试 (边界扫描)	把一个板上个部件的输入-输出引线连接成一条串联的扫描链，并且此方法已经被标准化以确保在不同厂商之间的兼容性。可以利用各种控制模块来测试各个部件以及板上的互连线。
自测试	测试规则结构（如存储器）时极为有用

## Chapter 9 脉冲电路

稳态时电阻两端初始电压相等。电容两端电压不突变，当电阻两端出现电压差时，高电压端放电，低电压端充电。

$$x(t) = x(\infty) + [x(0) - x(\infty)]e^{-t/\tau} \quad t = RC \ln \frac{V_R(\infty) - V_R(0)}{V_R(\infty) - V_{TH}}$$

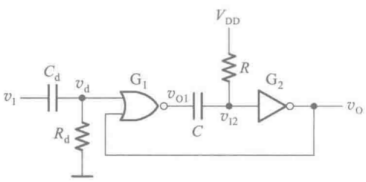
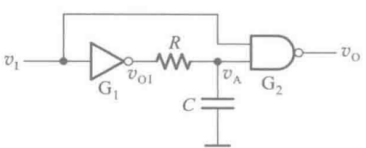
### 1、施密特触发电路

$V_{TH}$	$V_{T+}$	$V_{T-}$	$\Delta V$	传输特性	门电路组成
$\frac{1}{2}V_{DD}$	$(1 + \frac{R_1}{R_2})V_{TH}$	$(1 - \frac{R_1}{R_2})V_{TH}$	$2 \cdot \frac{R_1}{R_2}V_{TH}$		

特点：内部正反馈使得波形边沿很陡。

即输入电压上升到  $V_{T+}$  时，输出电压才会↑；下降到  $V_{T-}$  时，输出电压才会↓。

### 2、单稳态电路

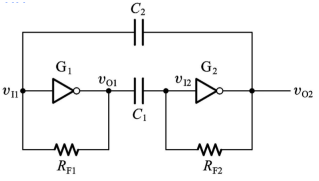
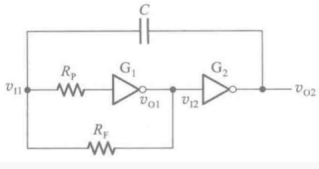
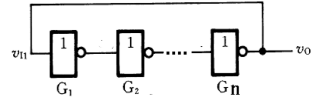
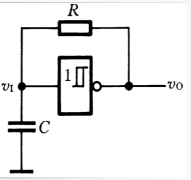
	$V_{TH}$	脉冲宽度 $t_w$	门电路组成
微分型	$\frac{1}{2}V_{DD}$	$RC \ln 2$	
积分型	$\frac{1}{2}V_{DD}$	$(R_0 + R)C \ln \frac{V_{OL} - V_{OH}}{V_{OL} - V_{TH}}$ $R_0$ : $G_1$ 内部电阻	

特点：（1）两个工作状态；（2）暂稳态维持时间长短取决于电路本身参数

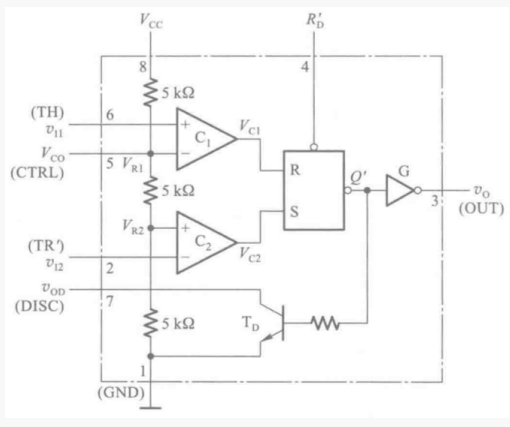
集成单稳态电路

触发方式	脉冲宽度	芯片图
B=1: A <sub>1</sub> A <sub>2</sub> 下降沿触发	$R_{ext}C_{ext}\ln 2$	
A <sub>1</sub> =0 或 A <sub>2</sub> =0: B上升沿触发		

### 3、多谐振荡电路

	振荡周期	门电路组成
对称式	$2R_E C \ln \frac{V_E - V_{IK}}{V_E - V_{TH}}$	
非对称式	$2R_F C \ln 3$	
环形	$2nt_{pd} \quad n \geq 3 \text{ 且为奇数}$	
施密特触发器构成	$RC \ln \frac{V_{DD} - V_{T-}}{V_{DD} - V_{T+}} \cdot \frac{V_{T+}}{V_{T-}}$	

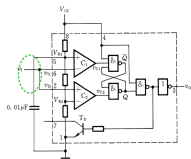
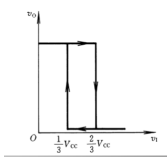
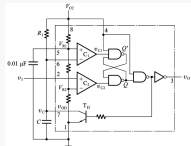
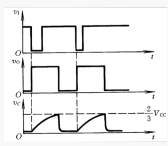
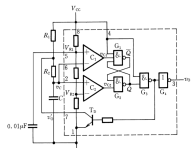
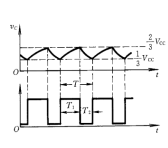
### 4、555定时器

输入			输出		芯片图
R <sub>D</sub> '	v <sub>I1</sub> (TH)	v <sub>I2</sub> (TR')	v <sub>O</sub>	T <sub>D</sub> 状态	
0	/	/	低	导通	
1	>2V <sub>CC</sub> /3	>1V <sub>CC</sub> /3	低	导通	
	<2V <sub>CC</sub> /3	>1V <sub>CC</sub> /3	不变	不变	
	<2V <sub>CC</sub> /3	<1V <sub>CC</sub> /3	高	截止	
	>2V <sub>CC</sub> /3	<1V <sub>CC</sub> /3	高	截止	

V<sub>CO</sub> 为控制端：

悬空时  $V_{R1} = \frac{2}{3}V_{CC} \quad V_{R2} = \frac{1}{3}V_{CC}$

接电压时  $V_{R1} = V_{CO} \quad V_{R2} = \frac{1}{2}V_{CO}$

	特点	芯片图	波形图
施密特 触发器	$V_{I1}$ 与 $V_{I2}$ 接在一起作为 $V_I$ $V_{T+} = \frac{2}{3}V_{CC}$ $V_{T-} = \frac{1}{3}V_{CC}$ $\Delta V_T = \frac{1}{3}V_{CC}$		
单稳态 电路	$t_w = RC\ln 3$ 不可重复触发		
多谐振 荡电路	充电: $T_1 = (R_1 + R_2)C\ln 2$ 放电: $T_2 = R_2C\ln 2$ 周期: $T = (R_1 + 2R_2)C\ln 2$ 占空比: $q = \frac{R_1 + R_2}{R_1 + 2R_2}$		

## Chapter 10 微处理器设计

### 1、微处理器结构

	特点
冯诺依曼结构	将程序存储和数据存储放在同一物理存储空间，具有更好的硬件效率。
哈佛结构	将程序存储和数据存储分别放在不同的物理存储空间，具有更好的灵活性和稳定性。

冯诺依曼结构主要包括：输入、输出、存储器、微处理器。

微处理器又分为控制单元与数据通路。

控制单元的指令周期：取指、译指、执行指令。

数据通路主要包括运算单元、存储单元（寄存器）。

### 2、微处理器指令集

**MIPS** 架构的四个原则：（1）简洁规整；（2）加快经常性事件的速度；（3）较小的速度更快；（4）好的设计要求良好的妥协。

汇编语言



	指令	
加法	<b>High-Level Code</b> a = b + c;	<b>MIPS Assembly Code</b> add a, b, c
减法	<b>High-Level Code</b> a = b - c;	<b>MIPS Assembly Code</b> sub a, b, c
复杂运算	<b>High-Level Code</b> a = b + c - d;   // single-line comment /* multiple-line comment */	<b>MIPS Assembly Code</b> sub t, c, d           # t = c - d add a, b, t           # a = b + t
寄存器操作	<b>High-Level Code</b> a = b + c;	<b>MIPS Assembly Code</b> # \$s0 = a, \$s1 = b, \$s2 = c add \$s0, \$s1, \$s2     # a = b + c
<i>load word</i>	<b>Assembly Code</b> # This assembly code (unlike MIPS) assumes word-addressable memory lw \$s3, 1(\$0)       # read memory word 1 into \$s3	
<i>store word</i>	<b>Assembly Code</b> # This assembly code (unlike MIPS) assumes word-addressable memory sw \$s7, 5(\$0)       # write \$s7 to memory word 5	
立即数操作	<b>High-Level Code</b> a = a + 4; b = a - 12;	<b>MIPS Assembly Code</b> # \$s0 = a, \$s1 = b addi \$s0, \$s0, 4       # a = a + 4 addi \$s1, \$s0, -12     # b = a - 12

机器语言

*MIPS* 采用 32 位指令，定义了三种指令格式：*R* 型、*I* 型、*J* 型。

指令格式	特点
<i>R</i> 型 (寄存器类型)	使用三个寄存器作为操作数：两个作为源，一个作为目标。如 <i>sub</i> 、 <i>add</i> 。
<i>I</i> 型 (立即数类型)	使用两个寄存器操作数和一个立即数。如 <i>lw</i> 、 <i>sw</i> 。
<i>J</i> 型 (跳转型)	只用跳转指令使用。有一个 26 位的立即数，无寄存器