期末速通

第一章 二进制编码

1. 计算机系统概述

- *冯·诺伊曼计算机*
  - 采用"存储程序"工作模式
  - 5个基本部件：运算器，控制器，存储器，输入设备，输出设备
  - 二进制形式表示指令和数据
- 计算机系统的层次结构

2. 二进制编码

- **进制转换**
  - 十进制 -> 二进制
  - 二进制 -> 八进制 十六进制
  - etc.
- **数字编码**
  - 原码
    - 1符号位+绝对值表示
  - **补码**
    - 正数补码同原码
    - 负数补码：符号位取1，数值各位取反，末位+1
  - 反码
- 整数表示
  - 无符号整数
  - 带符号整数
    - 补码表示
- 浮点数表示
  - IEEE 754：
    - 单精度：1符号位，8阶码位（移码表示），23尾数位（原码表示，第1位1省略）
    - 双精度：1符号位，11阶码位，52尾数位
    - 移码：数值+偏置常数（2^{n-1}-1）
- *BCD码*
  - 8421码：4位二进制数对应1位十进制数

第二章 数字逻辑基础

1. 布尔代数

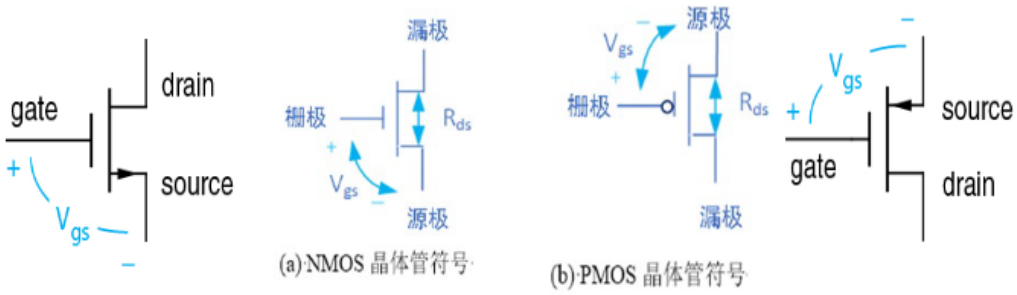
- 各种定理和定律
- **代数化简**

2. 逻辑门

- 门电路：与或非
- 数字抽象
- **CMOS晶体管**

CMOS晶体管

栅极和源极之间电压V_{gs}控制源极和漏极间电阻R_{ds}的大小



NMOS：增大V_{gs}，则R_{ds}下降

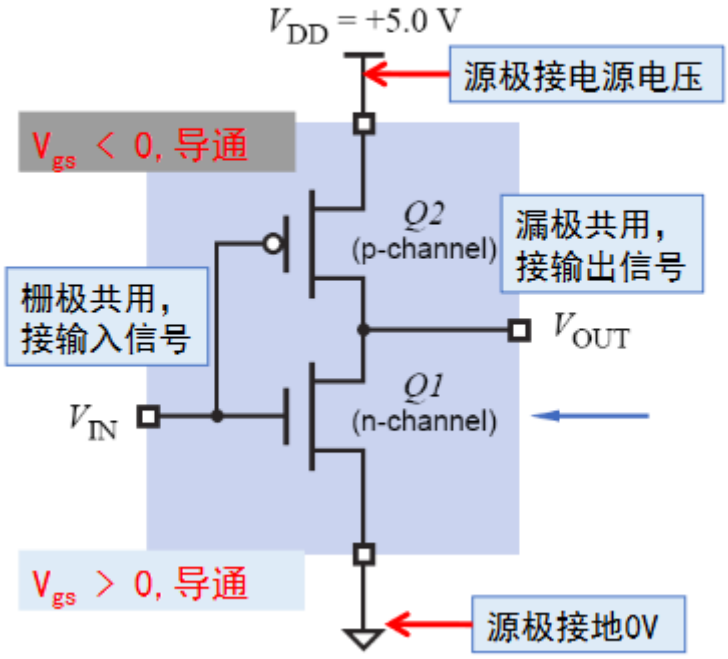
通常$V_{gs} \geqslant 0$

PMOS：减低V_{gs}，则R_{ds}下降

通常 $V_{gs} \leqslant 0$

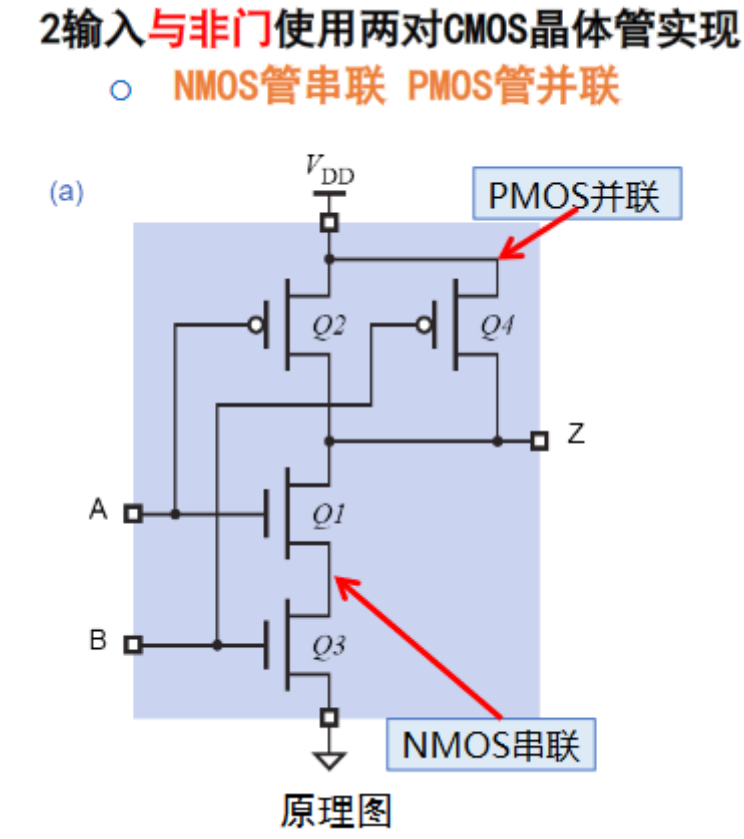
-

非门使用一对CMOS晶体管实现

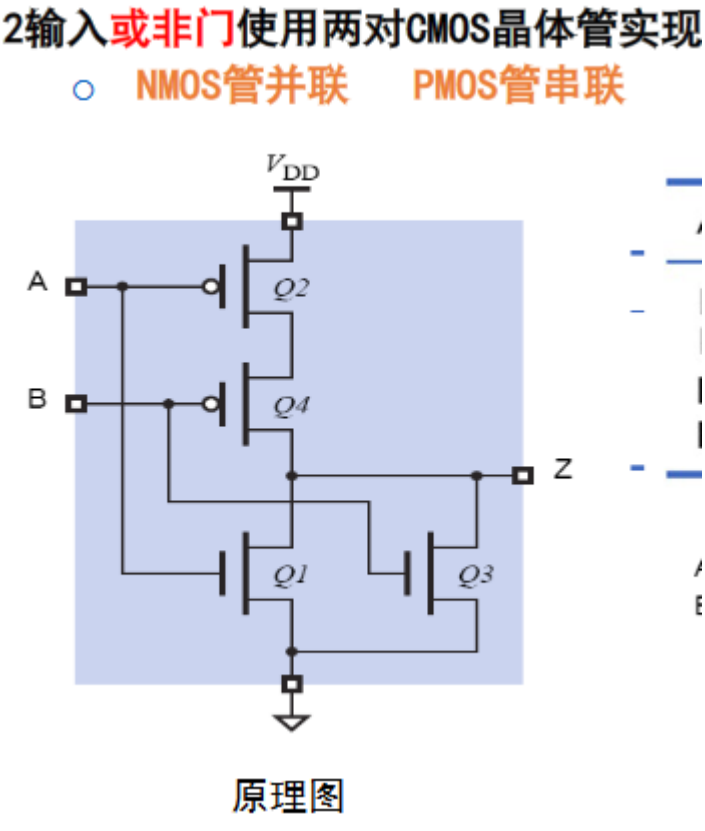


原理图

- 非门的实现



- 与非门的实现



- 或非门的实现
- *电气性质*
  - 转换时间，传输延迟，功率损耗

逻辑函数

- *表示*
  - 真值表，波形图
- 代数法化简

- **卡诺图化简**
  - 实质：将逻辑函数的最小项之和以图形的方式表示出来
  - 操作步骤
    1. 把逻辑函数表示为最小项之和，对应位置填1
    2. 具有相邻性的最小项可以合并
       - 2，4，8矩形合并
    3. 寻找最小覆盖
  - 一些概念
    - 质蕴涵项，实质蕴含项，覆盖，最小覆盖...
- *逻辑函数变换*

3. 组合逻辑电路

概述与概念

- 组合逻辑电路构成规则
  - 每个元件本身是组合逻辑电路
  - 输出连线不能互连
  - 输出连线不能反馈到元件输入端
- 输出和输入(受电气特性约束)
  - 扇入系数：一个逻辑门所允许的输入端的最大数目
  - 扇出系数：一个逻辑门输出端信号所能驱动的下一级输入端的最大数目
- 门延迟
  - 从逻辑门的输入信号改变开始，到输出信号发生改变所用的时间
- 无关项，非法值和高阻态
  - 无关项：任意项或约束项，用d表示
  - 非法值：同时被高电平和低电平驱动
  - 高阻态：三态门的第三种状态，类似电路断开
    - 用途：可用于连接总线，多个三态输出连在一起等

典型部件

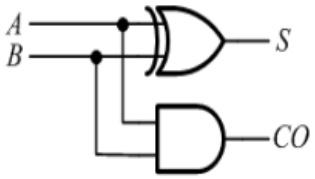
- 译码器
  - 输入n位二进制码，输出通常采用2^n中取1码（$n \sim 2^n$）
  - 用途：地址译码，指令操作码译码等
- 编码器
  - 典例：二进制编码器，与上述译码器相反
  - *优先权编码器*
- 多路选择器（多路复用器，数据选择器，MUX，multiplexer）
  - n位输入端+$\log_2 n$位控制端(向上取整)+1输出端
- 多路分配器(DEMUX)
  - n位输出端+$\log_2 n$位控制端(向上取整)+1输入端
- 半加器

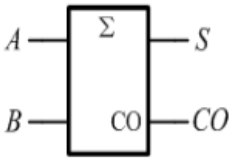
- **半加器（Half Adder，简称HA）：仅考虑加数和被加数，不考虑低位来的进位**

输入		输出	
被加	加数	和数	进位
A	*B*	*S*	*CO*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S=\overline{A}B+A\overline{B}=A\oplus B$$

$$CO=A\cdot B$$





电路图　　逻辑符号

- 全加器

全加器（Full Adder，简称FA）：输入为加数、被加数和低位进位Cin，输出为和F、进位Cout

A	*B*	Cin	*F*	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

真值表

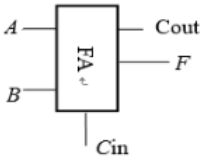
$$F=\overline{A}\cdot\overline{B}\cdot Cin+\overline{A}\cdot B\cdot\overline{Cin}+A\cdot\overline{B}\cdot\overline{Cin}+A\cdot B\cdot Cin$$

$$Cout=\overline{A}\cdot B\cdot Cin+A\cdot\overline{B}\cdot Cin+A\cdot B\cdot\overline{Cin}+A\cdot B\cdot Cin$$

化简后：

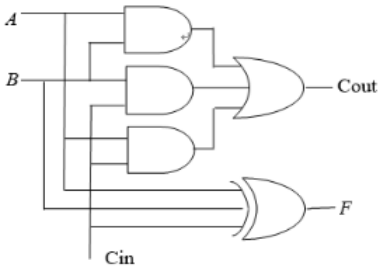
$$F=A\oplus B\oplus Cin$$

$$Cout=A\cdot B+A\cdot Cin+B\cdot Cin$$



逻辑符号

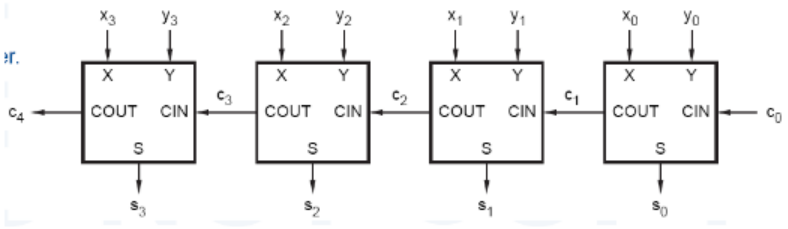
全加器逻辑电路图



- 串行进位加法器/行波进位加法器

串行进位加法器：行波进位加法器 ripper adder

- 规格：
  - 两个二进制字，每个n位，相加。
- 方法：
  - n个全加器的级联，属于迭代电路。
- 延迟：
- 特点：简单、速度慢



组合逻辑设计实践

时序分析

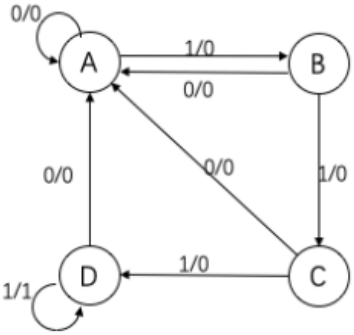
- 时序特征：传输延迟和最小延迟
  - 关键路径：一个组合逻辑电路在输入和输出之间经过的**最长**路径
  - 传输延迟：关键路径上所有元件的传输延迟之和
  - 最小延迟：最短路径上所有元件的最小延迟之和
- *竞争冒险*
  - 竞争(race)：如果存在某个输入信号经过两条或两条以上的路径作用到输出端，由于各路径延迟不同，因而该输入信号对输出端会发生先后不同的影响
  - 毛刺(glitch)：输入信号变化瞬间产生不正确的尖峰信号
    - 消除：低通滤波或冗余项
  - 冒险(hazard)：出现毛刺的电路

4. 时序逻辑电路

概念

- 有限状态机(Finite State Machine，FSM)：一种刻画状态及状态转换的理论工具
  - 通常使用状态图来描述

例：检测输入序列是否为连续4个“1”
A-初始态：若输入1，则转B
B-连续1个“1”：若输入1，则转C
C-连续2个“1”：若输入1，则转D
D-连续3个“1”：若输入1，则状态不变
并输出为1（表示检测到连续4个1）
任何状态下，输入0都会转到初态A



状态表

现态 Q	输入RS 00	01	10	11
0	0	1	0	0*
1	1	1	0	0*

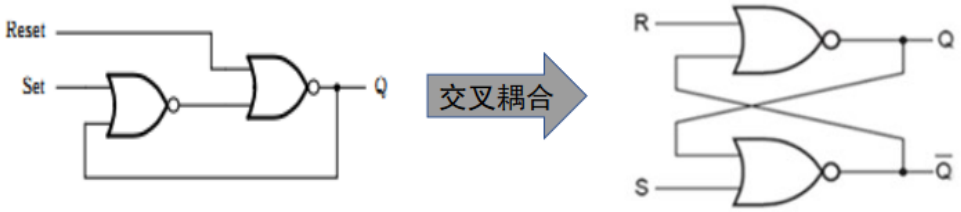
  - or状态表
- 基本结构
  - 三组块结构：状态记忆模块，次态激励逻辑模块F(对应激励函数)，输出逻辑模块G(对应输出函数)
  - Mealy型：输出依赖于当前状态和当前输入信号
  - Moore型：输出仅依赖于当前状态，和当前输入信号无关
- *电路定时*
  - 时钟周期，时钟频率
  - 一般采用边沿触发方式
    - 上升沿触发，下降沿触发

双稳态元件

- 概念
  - 两种类型：
    - 锁存器(latch)：电平触发
    - 触发器(flipflop)：边沿触发
  - 置位(高电平，1)，复位(低电平，0)
- 典型元件
  - SR锁存器

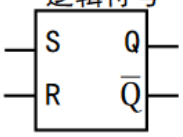
SR锁存器

SR锁存器：使用一对交叉耦合的或非门构成双稳态电路，也称为置位-重置（复位）锁存器。S是置位输入端，R是重置输入端



S	R	Q	$\overline{Q}$
0	0	状态不变	
0	1	0	1
1	0	1	0
1	1	禁	止

逻辑符号



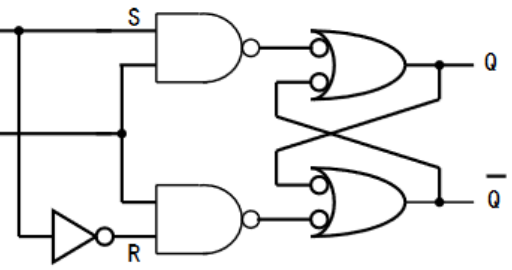
R=S=1时，Q、 $\overline{Q}$状态不相反，无效

  - D锁存器

D锁存器

S=1, R=0时， Q=1
S=0, R=1时， Q=0
S、R输入互补

数据输入端 D
控制端 C



D锁存器功能表

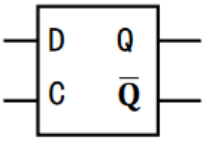
C	D	Q	$\overline{Q}$
0	X	保	持
1	0	0	1
1	1	1	0

C=0时， 输出状态保持不变

C=1时， D = 1时， Q = 1
D = 0时， Q = 0
} Q = D

输出随输入状态而改变

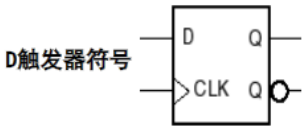
逻辑符号



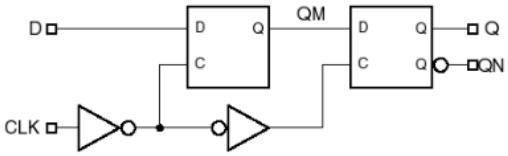
只有一个数据输入端D，称为D锁存器，也称为透明锁存器

  - D触发器

D触发器



由一对主、从D锁存器可构成一个D触发器

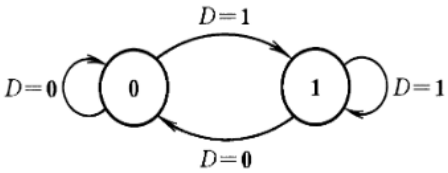


CLK	主锁存器	从锁存器
L	写入	不变
上升沿	锁存	开始写入
H	不变	写入

从锁存器只在时钟CLK的上升沿到来时采样主锁存器的输出QM的值，并确定Q和QN的输出

D	CLK	Q	QN
0	↑	0	1
1	↑	1	0
x	0	last Q	last QN
x	1	last Q	last QN

◆状态转移图



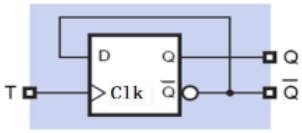
◆D触发器特征方程：Q* = D

- T触发器

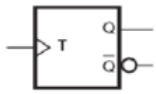
T触发器

T触发器：在每个时钟脉冲的触发边沿都会改变状态

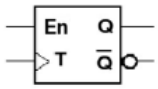
基于D触发器实现；可用于实现计数器、分频器等功能



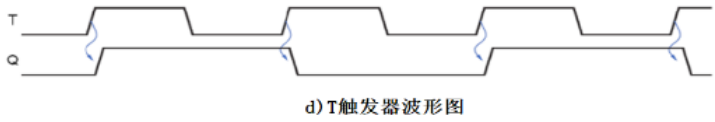
a) T触发器原理图



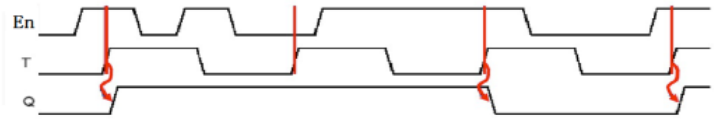
b) T触发器电路符号



C) 带使能端T触发器电路符号



d) T触发器波形图



e) 带使能端T触发器波形图

同步时序逻辑设计

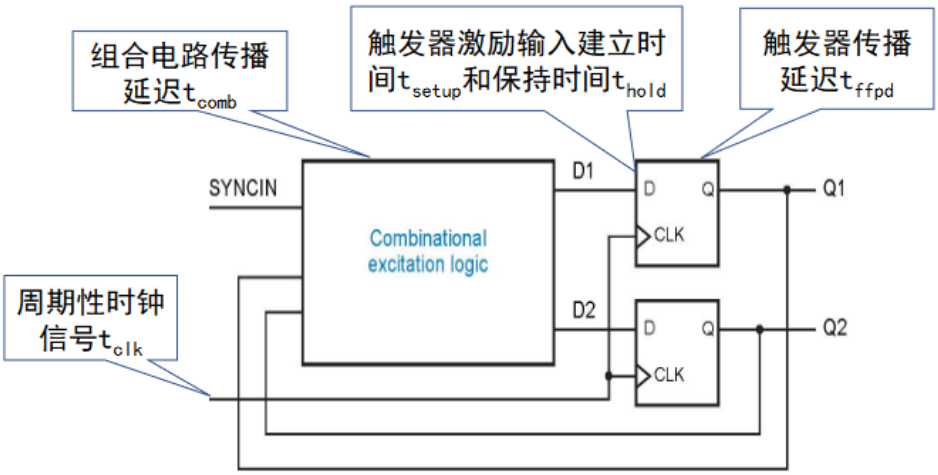
- 电路分析
  - 设计步骤
    1. 激励方程
       - 就是输入端的方程
    2. 转移方程
       - 状态Q^{n+1}关于Q^n的方程
    3. 输出方程
       - 输出端的关于状态的方程
    4. 构建转移表
    5. 画状态转移图
  - 状态设计，化简与编码

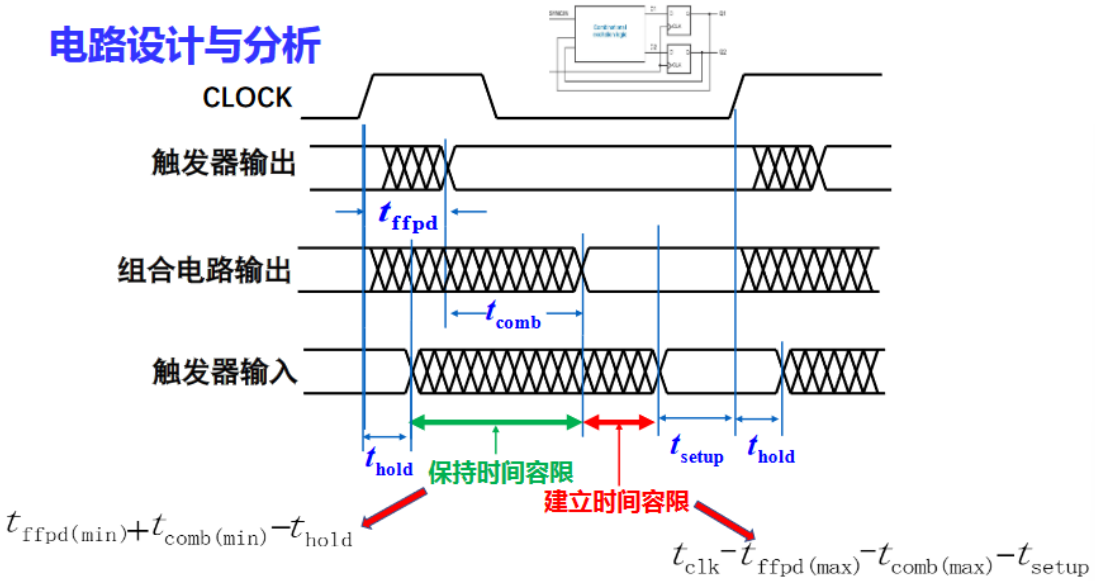
    - 互斥性，完备性
    - 化简：等价状态(在所有输入组合下，它们的输出相同且次态相同或次态等价)合并
    - 编码：次优编码
- 未用状态分析：编码空间>状态集合
    - 挂起：电路加电后进入未用状态，且在未用状态之间形成循环转换而无法进入工作状态
    - 自启动：判定电路进入未用状态时能否在有限个时钟周期后进入到工作状态
- 定时分析

电路设计与分析

电路定时分析

- 时序逻辑电路的工作频率和组合逻辑电路传输延迟、触发器建立和保持时间、触发器传输延迟等时间密切相关。





时间容限指为保证电路正常工作某信号定时所允许的最大时间范围

电路设计与分析

- 建立时间容限=$t_{clk}-t_{ffpd(max)}-t_{comb(max)}-t_{setup}$，>0
- 保持时间容限=$t_{ffpd(min)}+t_{comb(min)}-t_{hold}$，>0

因此，得到时序约束关系：

(1) $t_{clk} > t_{ffpd(max)} + t_{comb(max)} + t_{setup}$

(2) $t_{hold} < t_{ffpd(min)} + t_{comb(min)}$

（1）为使触发器正常工作，必须保证时钟周期t_{clk}不能小于触发器锁存延迟t_{ffpd} + 次态信号经过激励逻辑延迟t_{comb} + 触发器的建立时间t_{setup} 。

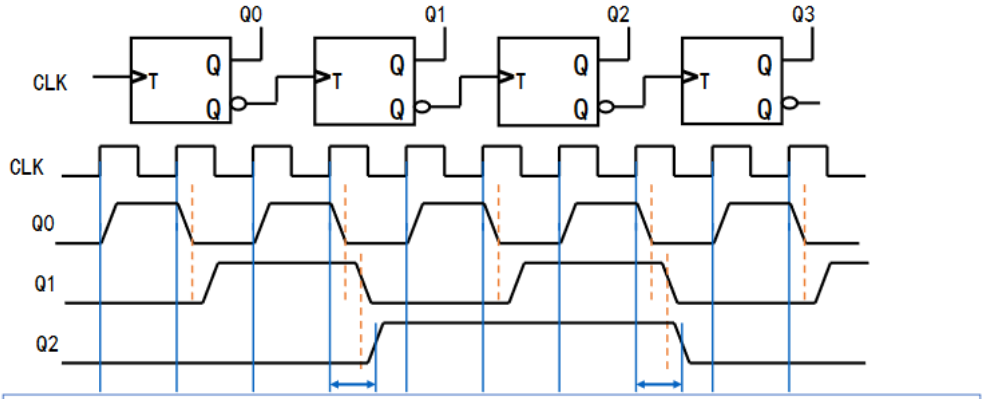
（2）为使触发器正常工作，必须保证外部激励信号在时钟有效边沿到来后的保持时间t_{hold}内能保持稳定不变。这就要求次态信号不能反馈太快，即触发器锁存延迟t_{ffpd} + 次态信号经过激励逻辑延迟t_{comb}不能小于触发器的保持时间t_{hold} 。

典型部件设计

- 计数器
  - 异步行波加法计数器

异步行波加法计数器

利用 T 触发器实现，激励输入像波浪一样由低位向高位传递，每个时钟周期传送一次。



Q_{i+1}总是在Q_i由1变为0时开始改变状态
第n位状态变换最长要经过 $n\times t_{TQ}$ 的延迟时间

  - 同步并行加法计数器

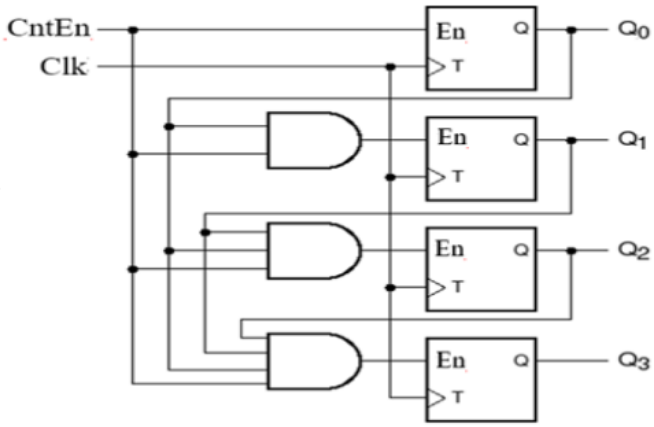
计数器

当编码为1111时，只要经过一个与门+t_{TQ} 延时，就可回到编码0000，比行波（串行）加法计数器快得多！

同步并行加法计数器

- 计数器的状态编码$Q_3Q_2Q_1Q_0$从0000开始，转换过程为
  0000→0001→0010→0011→0100→0101→0110→0111→1000→…→1111

CntEN有效时，每个时钟Q_0都会发生状态改变；对于Q_1、Q_2和Q_3，只有在其所有低位状态都是1的情况下，下个时钟边沿到来后才会发生状态反转。

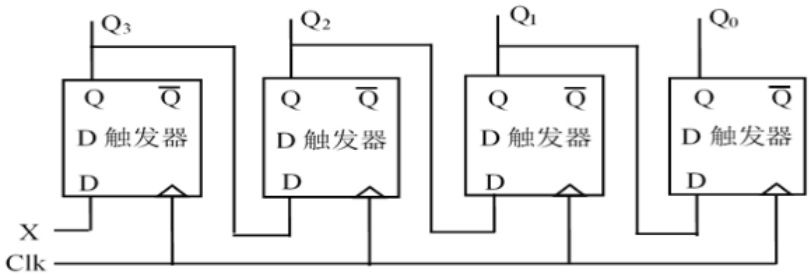


- **寄存器**
  - 寄存器是用来暂存信息的逻辑部件，可直接由若干个触发器组成
  - 移位寄存器

移位寄存器

移位寄存器

能够实现暂存信息的左移或右移功能，通常由时钟信号控制
例如：4个D触发器可构成一个右移寄存器



假设初始状态编码$Q_3Q_2Q_1Q_0$=0000，X输入为序列10011011
则$Q_3Q_2Q_1Q_0$Q3的输出编码依次为：
0000、1000、0100、0010、1001、1100、0110、1011、1101

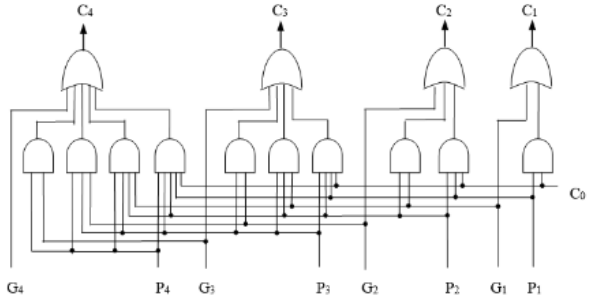
  - 通用移位寄存器
  - 桶形移位器

6. 运算方法和运算部件

运算部件

- 加法器
  - 串行进位加法器/行波进位加法器：同上
    - 缺点：进位按串行方式传递，速度慢
  - 并行进位加法器（CLA加法器, Carry Lookahead Adder）

并行进位加法器（CLA加法器）



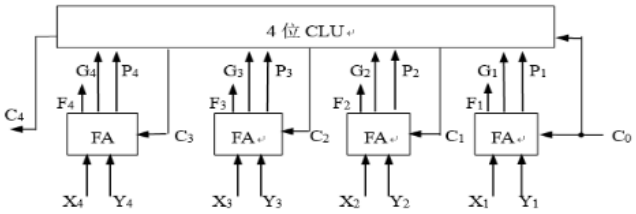
$C_1=G_1+P_1C_0$

$C_2=G_2+P_2C_1=G_2+P_2G_1+P_2P_1C_0$

$C_3=G_3+P_3C_2=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1C_0$

$C_4=G_4+P_4C_3=G_4+P_4G_3+P_4P_3G_2+P_4P_3P_2G_1+P_4P_3P_2P_1C_0$

4位CLU部件



$G_i=A_iB_i$

$P_i=A_i+B_i$（或 $P_i=A_i \oplus B_i$）

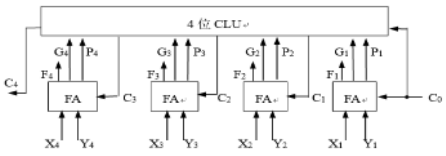
$F_i=A_i \oplus B_i \oplus C_{i-1}$

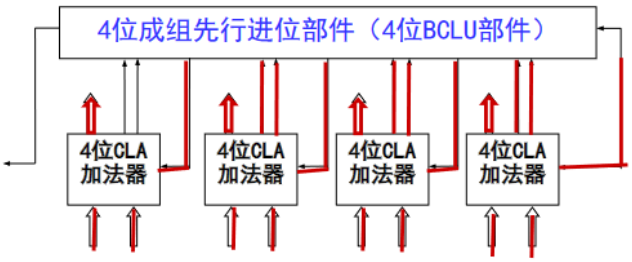
4位CLA加法器

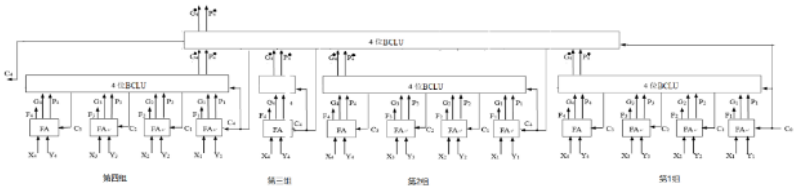
-

重要内容一：加法器

16位两级先行进位加法器



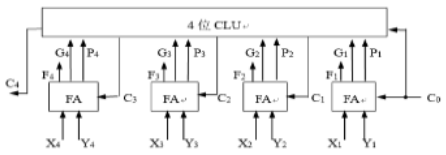


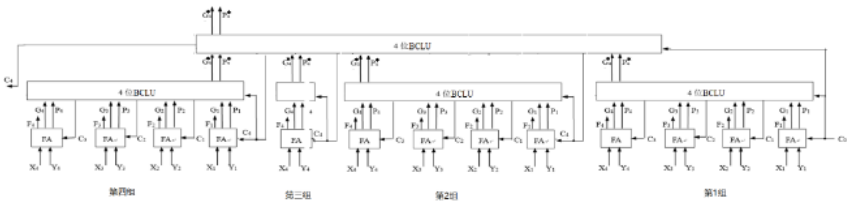


-

重要内容一：加法器

16位两级先行进位加法器





BCLU向下传递进位信号后，下一级CLU计算超前进位需要的延迟

1(gp)+2(clu)+2(clu)+2(clu)+3(xor)=10ty

-

- 多级先行进位加法器：组内并行，组间串行
  - 延迟计算

- 单级(局部)先行进位加法器的进位生成方式：
  “组内并行、组间串行”
- 所以，单级先行进位加法器虽然比行波加法器延迟时间短，但高位组进位依赖低位组进位，故仍有较长的时间延迟
- 通过引入组进位生成/传递函数实现“组内并行、组间并行”进位方式

设n=4, 则： $C_1=G_1+P_1C_0$

$C_2=G_2+P_2C_1=G_2+P_2G_1+P_2P_1C_0$

$C_3=G_3+P_3C_2=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1C_0$

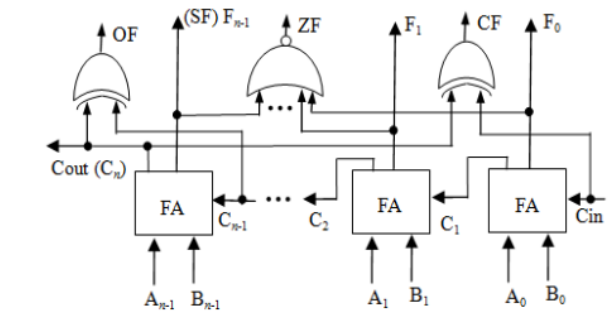
$C_4=G_4+P_4C_3=G_4+P_4G_3+P_4P_3G_2+P_4P_3P_2G_1+P_4P_3P_2P_1C_0$

$G_4^*=G_4+P_4C_3=G_4+P_4G_3+P_4P_3G_2+P_4P_3P_2G_1$ } 额外的输出信号(CLU 的进位生成/传输信号)

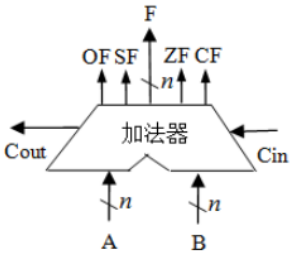
$P_4^*=P_4P_3P_2P_1$

所以$C_4=G_4^*+P_4^*C_0$。把实现上述逻辑的电路称为4位BCLU（Block CLU）部件。

- n位带标志加法器



带标志加法器的逻辑电路



带标志加法器符号

溢出标志OF： 四位元带号数：
$OF=C_n\oplus C_{n-1}$
符号标志SF： 负：7-(≥8)
$SF=F_{n-1}$ 补机：7+(≤8) 且元位
零标志ZF=1：
当且仅当F=0； 负：7-(≤7)
进位/借位标志CF： 补机：7+(≥9)
$CF=Cout\oplus Cin$ 有进位

- 算术逻辑部件ALU

运算方法

- 加减运算
  - 补码加减运算
    - 判断溢出
      - 最高位和次高位进位不同
      - 和的符号位和加数的符号位不同
  - 原码加减运算
  - 移码加减运算
    - 移码和/差 = 和/差的补码
  - 乘法运算
    - 原码一位乘法

无符号整数乘法运算举例

举例说明：若需计算z=x*y; x、y和z都是unsigned类型。

设x=1110　y=1101　应用递推公式：$P_i=2^{-1}(x*y_i+P_{i-1})$

C	乘积P	乘数R
0	0000	1101
+	1110	
0	1110	1101
→ 0	0111	0110
→ 0	0011	1011
+	1110	
1	0001	1011
→ 0	1000	1101
+	1110	
1	0110	1101
→ 0	1011	0110

可用一个双倍字长的乘积寄存器；也可用两个单倍字长的寄存器。

部分积初始为0。

保留进位位。

右移时进位、部分积和剩余乘数一起进行逻辑右移。

当z取4位时，结果发生溢出，因为高4位不为全0！

原码乘法运算

用于浮点数尾数乘运算

符号与数值分开处理：积符异或得到，数值用无符号乘法运算

例：设$[x]_{原}=0.1110$，$[y]_{原}=1.1101$，计算$[x\times y]_{原}$

解：数值部分用无符号数乘法算法计算：1110×1101= 1011 0110

符号位：0⊕1=1，所以：$[x\times y]_{原}=1.10110110$

一位乘法：每次只取乘数的一位判断，需n次循环，速度慢。

两位乘法：每次取乘数两位判断，只需n/2次循环，快一倍。

◆两位乘法递推公式：　T触发器用来记录下次是否要执行“+X”

“–X”运算用“+[-X]补”实现！

00：$P_{i+1}=2^{-2}P_i$

01：$P_{i+1}=2^{-2}(P_i+X)$

10：$P_{i+1}=2^{-2}(P_i+2X)$

11：$P_{i+1}=2^{-2}(P_i+3X)=2^{-2}(P_i+4X-X)$

$=2^{-2}(Pi-X)+X$

3X时，本次-X，下次+X！

部分积右移两位，相当于4X

y_{i-1}	y_i	T	操　作	迭代公式
0	0	0	0→T	$2^{-2}(P_i)$
0	0	1	+X　0→T	$2^{-2}(P_i+X)$
0	1	0	+X　0→T	$2^{-2}(P_i+X)$
0	1	1	+2X　0→T	$2^{-2}(P_i+2X)$
1	0	0	+2X　0→T	$2^{-2}(P_i+2X)$
1	0	1	–X　1→T	$2^{-2}(P_i-X)$
1	1	0	–X　1→T	$2^{-2}(P_i-X)$
1	1	1	1→T	$2^{-2}(P_i)$

- 原码两位乘法

原码两位乘法举例

已知　$[X]_{原}=0.111001$，　$[Y]_{原}=0.100111$，用原码两位乘法计算$[X\times Y]_{原}$

解：　先用无符号数乘法计算111001×100111，原码两位乘法过程如下：

$[[X]_{补}=000\ 111001$，$[-[X]]_{补}=111\ 000111$

P		Y	T	说明
000	000000	100111	0	开始，$P_0=0$，T=0
+111	000111			y_5y_6T=110，–X，T=1
111	000111			P 和 Y 同时右移 2 位
111	110001	11 1001	1	得 P_1
+001	110010			y_3y_4T=011，+2X，T=0
001	100011			P 和 Y 同时右移 2 位
000	011000	1111 10	0	得 P_2
+001	110010			y_1y_2T=100，+2X，T=0
010	001010			P 和 Y 同时右移 2 位
000	100010	101111	0	得 P_3

加上符号位，得 $[X\times Y]_{原}=0.100010101111$

采用补码算术右移，与一位乘法不同，Why？

为模8补码形式(三位符号位)，Why？

若用模4补码，则P和Y同时右移2位时，得到的P3是负数，这显然是错误的！需要再增加一位符号。

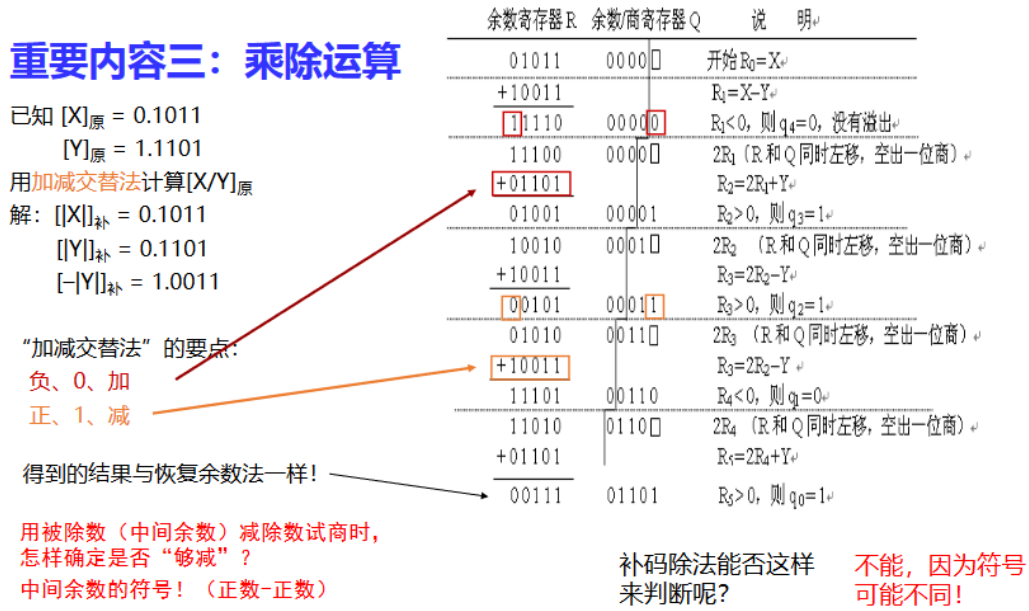
速度快，但代价也大

- 补码一位乘法(Booth算法)

- 补码两位乘法(MBA)

- **原码除法运算**
  - 恢复余数除法

  - 不恢复余数除法(加减交替法)

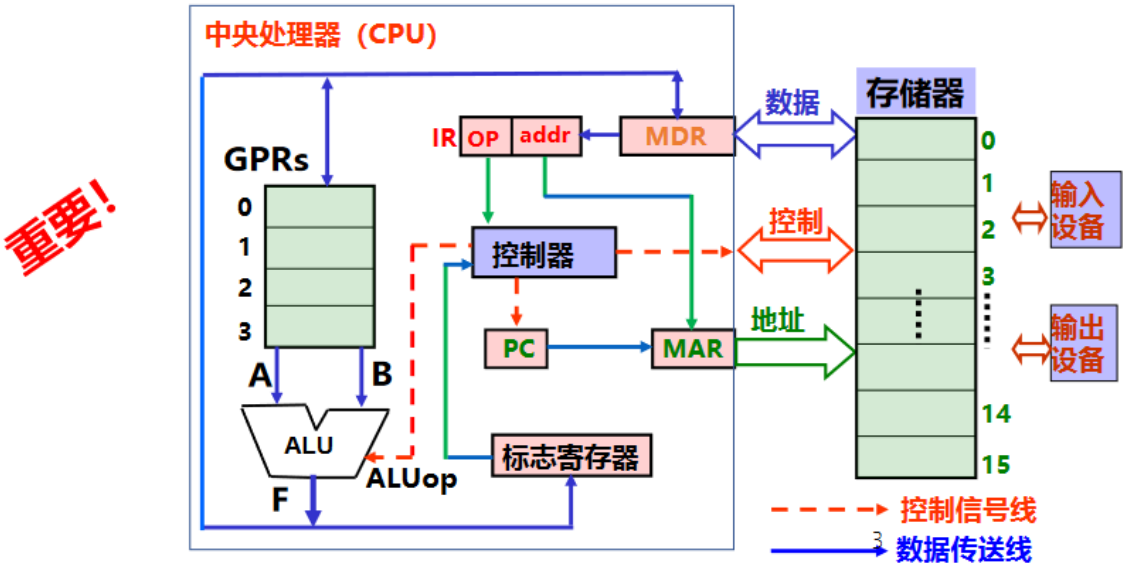


- *补码除法运算*

7. 指令系统

- 现代计算机结构模型

CPU：中央处理器； ALU：算术逻辑部件；控制器；存储器；
MAR：存储器地址寄存器；MDR：存储器数据寄存器；
PC：程序计数器； IR：指令寄存器；
GPRs：通用寄存器组（由若干通用寄存器组成，早期就是累加器）



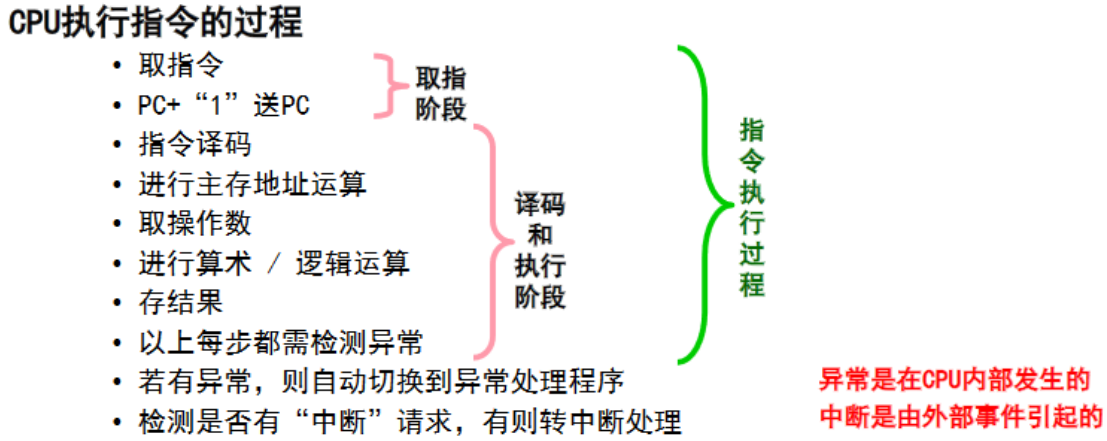
- 寻址方式
  - 立即寻址
  - 直接寻址
  - 间接寻址
  - 寄存器寻址
  - 寄存器间接寻址
  - 偏移寻址
    - 变址寻址
    - 相对寻址
    - 基址寻址
- 指令分类
  - 算术和逻辑运算指令
  - 移位指令
  - 数据传送指令
  - 串指令
  - 顺序控制指令
  - 系统控制指令
  - 输入/输出指令
- RISC设计风格
  - 简化的指令系统
  - 以RR方式工作：除Load/Store指令可访问存储器外，其余指令都只访问寄存器
  - 指令周期短
  - 采用大量通用寄存器，以减少访存次数
  - 采用组合逻辑电路控制，不用或少用微程序控制
  - 采用优化的编译系统，力求有效地支持高级语言程序
- 异常和中断

- 异常：在CPU内部发生的意外事件或特殊事件
  - 故障
  - 自陷
  - 终止
- 中断：在CPU外部发生的特殊事件，通过"中断请求"信号向CPU请求处理。如实时钟、控制台、打印机缺纸、外设准备好、采样计时到、DMA传输结束等

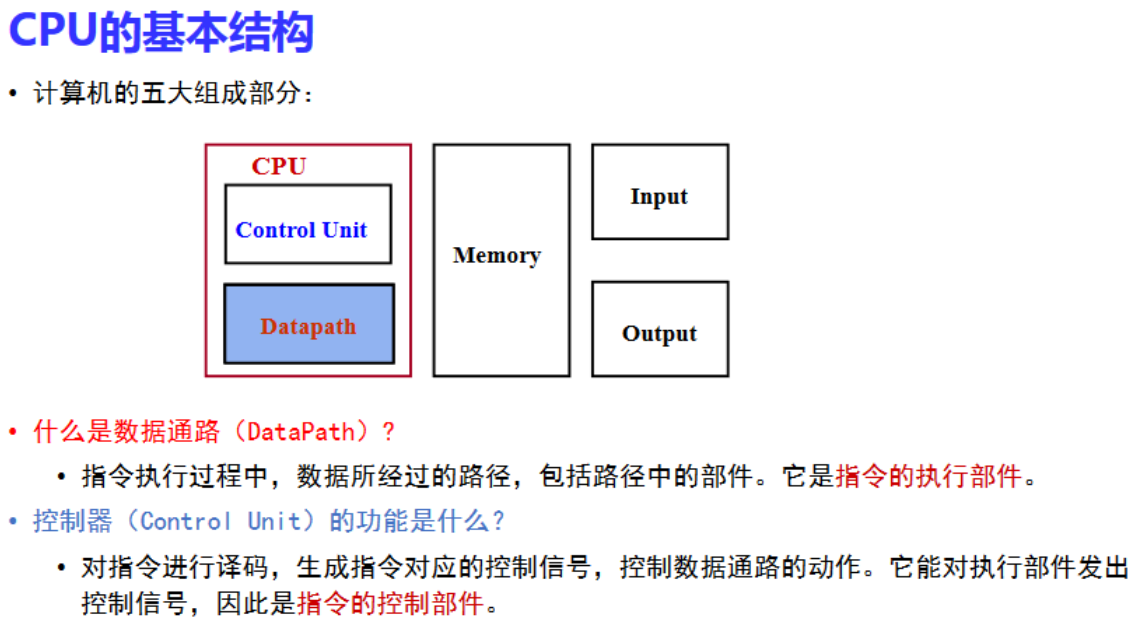
8. 中央处理器

概述

- 指令流程

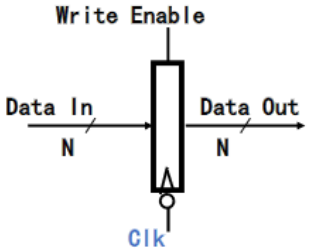


- 基本组成：同上

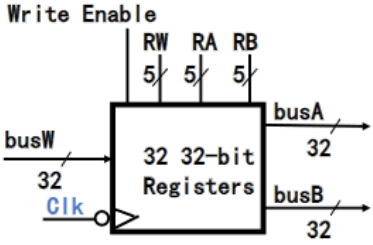


- 数据通路：由操作元件和存储元件通过总线方式或分散方式连接而成的进行数据存储、处理、传送的路径
  - 操作元件：组合逻辑电路
    - Adder，MUX，ALU，Decoder，etc.
  - 状态元件：时序逻辑电路
  - 存储元件：寄存器和寄存器组

存储元件：寄存器和寄存器组

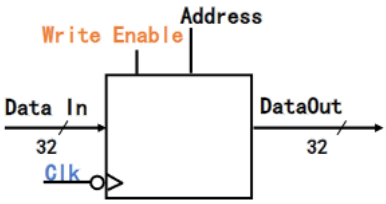


- 寄存器（Register）
  - 有一个写使能（Write Enable-WE）信号
    0：时钟边沿到来时，输出不变
    1：时钟边沿到来时，输出开始变为输入
  - 若每个时钟边沿都写入，则不需WE信号
- 寄存器组（Register File）
  - 两个读口（组合逻辑操作）：busA和busB分别由RA和RB给出地址。地址RA或RB有效后，经一个“取数时间(AccessTime)”，busA和busB有效。
  - 一个写口（时序逻辑操作）：写使能为1的情况下，时钟边沿到来时，busW传来的值开始被写入RW指定的寄存器中。



- 存储元件：理想存储器

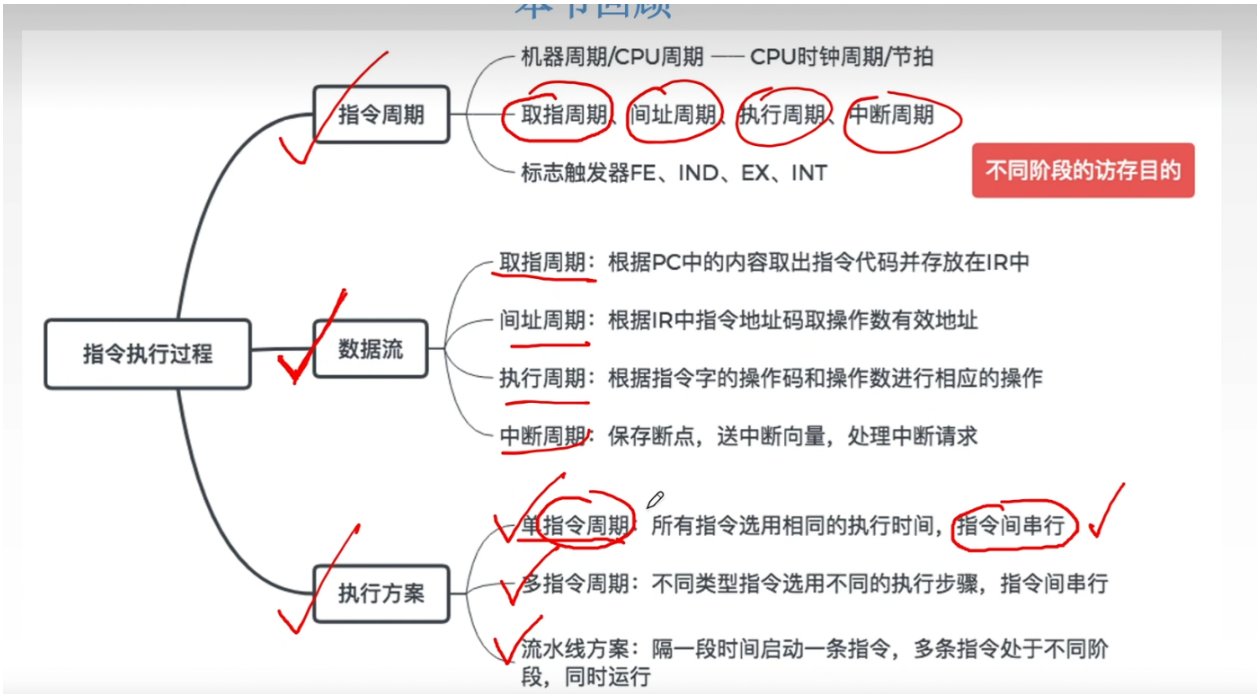
存储元件：理想存储器



- 理想存储器（ idealized memory ）
  - Data Out：32位读出数据
  - Data In： 32位写入数据
  - Address： 读写公用一个32位地址
  - 读操作（组合逻辑操作）：地址Address有效后，经一个“取数时间AccessTime”，Data Out上数据有效。
  - 写操作（时序逻辑操作）：写使能为1的情况下，时钟Clk边沿到来时，Data In传来的值开始被写入Address指定的存储单元中。

为简化数据通路操作说明，把存储器简化为带时钟信号Clk的理想模型

- 指令周期
  - 时钟周期<机器周期<指令周期



- 单周期，多周期，流水线...(自己看吧)