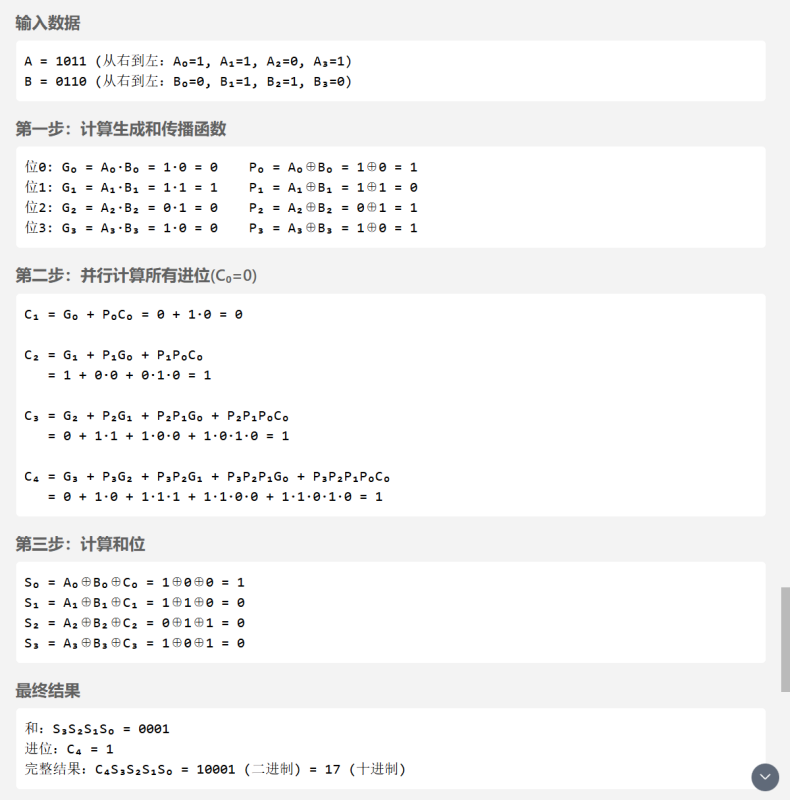
期末知识点总结梳理（chapter3）

- 知识点提炼
  - 1.存储部件的构建
  - 2.运算部件的构建
  - 3.如何用更少的位宽实现更大的数据表示范围?
  - Operations on integers
    - Addition and subtraction
    - Multiplication and division
    - Dealing with overflow
  - Floating-point real numbers
    - Representation and operations
- 整数加法器的设计?
  - 超前进位方法



输入数据

A = 1011 (从右到左: A_0=1, A_1=1, A_2=0, A_3=1)
B = 0110 (从右到左: B_0=0, B_1=1, B_2=1, B_3=0)

第一步：计算生成和传播函数

位0: $G_0 = A_0 \cdot B_0 = 1 \cdot 0 = 0$　　$P_0 = A_0 \oplus B_0 = 1 \oplus 0 = 1$
位1: $G_1 = A_1 \cdot B_1 = 1 \cdot 1 = 1$　　$P_1 = A_1 \oplus B_1 = 1 \oplus 1 = 0$
位2: $G_2 = A_2 \cdot B_2 = 0 \cdot 1 = 0$　　$P_2 = A_2 \oplus B_2 = 0 \oplus 1 = 1$
位3: $G_3 = A_3 \cdot B_3 = 1 \cdot 0 = 0$　　$P_3 = A_3 \oplus B_3 = 1 \oplus 0 = 1$

第二步：并行计算所有进位(C_0=0)

$C_1 = G_0 + P_0C_0 = 0 + 1 \cdot 0 = 0$

$C_2 = G_1 + P_1G_0 + P_1P_0C_0$
$= 1 + 0 \cdot 0 + 0 \cdot 1 \cdot 0 = 1$

$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
$= 0 + 1 \cdot 1 + 1 \cdot 0 \cdot 0 + 1 \cdot 0 \cdot 1 \cdot 0 = 1$

$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$
$= 0 + 1 \cdot 0 + 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 0 \cdot 0 + 1 \cdot 1 \cdot 0 \cdot 1 \cdot 0 = 1$

第三步：计算和位

$S_0 = A_0 \oplus B_0 \oplus C_0 = 1 \oplus 0 \oplus 0 = 1$
$S_1 = A_1 \oplus B_1 \oplus C_1 = 1 \oplus 1 \oplus 0 = 0$
$S_2 = A_2 \oplus B_2 \oplus C_2 = 0 \oplus 1 \oplus 1 = 0$
$S_3 = A_3 \oplus B_3 \oplus C_3 = 1 \oplus 0 \oplus 1 = 0$

最终结果

和: $S_3S_2S_1S_0 = 0001$
进位: $C_4 = 1$
完整结果: $C_4S_3S_2S_1S_0 = 10001$ (二进制) = 17 (十进制)

    - 超前进位追求极致速度但硬件复杂；分组进位在速度和复杂度间取得平衡，更适合实际应用。
  - 分组进位方法

分组进位的层次结构

第1层：组内处理（4位CLA）

```
输入：A = 1011, B = 0110

组0内部使用CLA计算：
├─ 和位：S₃S₂S₁S₀ = 0001
├─ 组内最高进位：C₄ = 1
└─ 生成组级信号给上层使用
```

组级信号的计算

组级生成信号 G_0^{group}：

$$G_0^{group} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$
$$= 0 + 1 \cdot 0 + 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 0 \cdot 0$$
$$= 0 + 0 + 1 + 0 = 1$$

含义：这个4位组能够"生成"一个进位给下一组

组级传播信号 P_0^{group}：

$$P_0^{group} = P_3P_2P_1P_0 = 1 \cdot 1 \cdot 0 \cdot 1 = 0$$

含义：这个4位组不能完全"传播"来自前一组的进位
(因为$P_1=0$，中断了传播链)

第2层：组间进位计算

如果是16位加法器(4组)，组间进位公式为：

$$C_4 = G_0^{group} + P_0^{group} \cdot C_0^{group}$$
$$C_8 = G_1^{group} + P_1^{group} \cdot C_4$$
$$C_{12} = G_2^{group} + P_2^{group} \cdot C_8$$
$$C_{16} = G_3^{group} + P_3^{group} \cdot C_{12}$$

- 整数乘法器的优化
  - 空间角度的优化
    - 压缩寄存器

      优化前：被乘数寄存器 + 积寄存器（独立的64位寄存器）
      优化后：被乘数和积共享同一个64位寄存器
    - 复用寄存器

      原理：一个寄存器在不同阶段存储不同数据

      ```
      示例：64位寄存器在乘法过程中的复用
      ├─ 初始阶段：存储被乘数（Multiplicand）
      ├─ 计算阶段：逐步存储部分积
      └─ 结束阶段：存储最终乘积（Product）
      ```
    - 减少位宽

      原理：根据实际需求优化寄存器位宽

      ```
      优化策略：
      ├─ 乘数寄存器：从64位减少到32位
      ├─ 原因：乘数在右移过程中逐渐减少有效位
      └─ 节省：32位硬件资源
      ```
  - 时间角度的优化
    - 左移位
      - 减少乘数寄存器的移位操作
      - 提高控制逻辑效率
    - 多个乘法并行执行
      - 并行策略：
      - ├── 流水线乘法：多个乘法在不同阶段同时进行
      - ├── 多核乘法：多个乘法器同时工作
      - └── 向量乘法：一次处理多个数据

- 整数除法器的优化
  - 压缩寄存器
  - 复用寄存器
  - 减少位宽
  - 负数的符号位处理复杂，简单右移会改变符号
  - 除法：必须逐步试商，无法并行A ÷ B = 需要逐位确定商，后一位依赖前一位结果
- 浮点数
  - 实际值 = (-1)^符号位 × 1.尾数 × 2^(指数-偏移量)

    具体格式

    32位单精度 (Single Precision)

    ```
    |符号|  指数  |     尾数      |
    | 1  |   8   |      23      |
    |31  |30-23  |    22-0      |
    ```

    64位双精度 (Double Precision)

    ```
    |符号|     指数     |           尾数           |
    | 1  |    11     |           52           |
    |63  |  62-52    |          51-0          |
    ```

    以32位为例:

    十进制：`-2.34 × 10^56`

    步骤：
    1. 转换为二进制科学计数法
    2. 符号位：1（负数）
    3. 指数：调整并加偏移量127
    4. 尾数：保存小数部分（去掉隐含的1）
  - 单精度+127，双精度+1023
  - 浮点数的加法和乘法操作
  - 浮点数加法不满足结合律
    - 浮点数精度有限
    - 大数加小数时，小数可能被"吞没"
  - 位模式本身没有固有含义
  -