

本节内容

计算机实现 乘法运算的 三种方式

(以无符号整数乘法为例)

知识总览

计算机实现乘法运算的三种方式

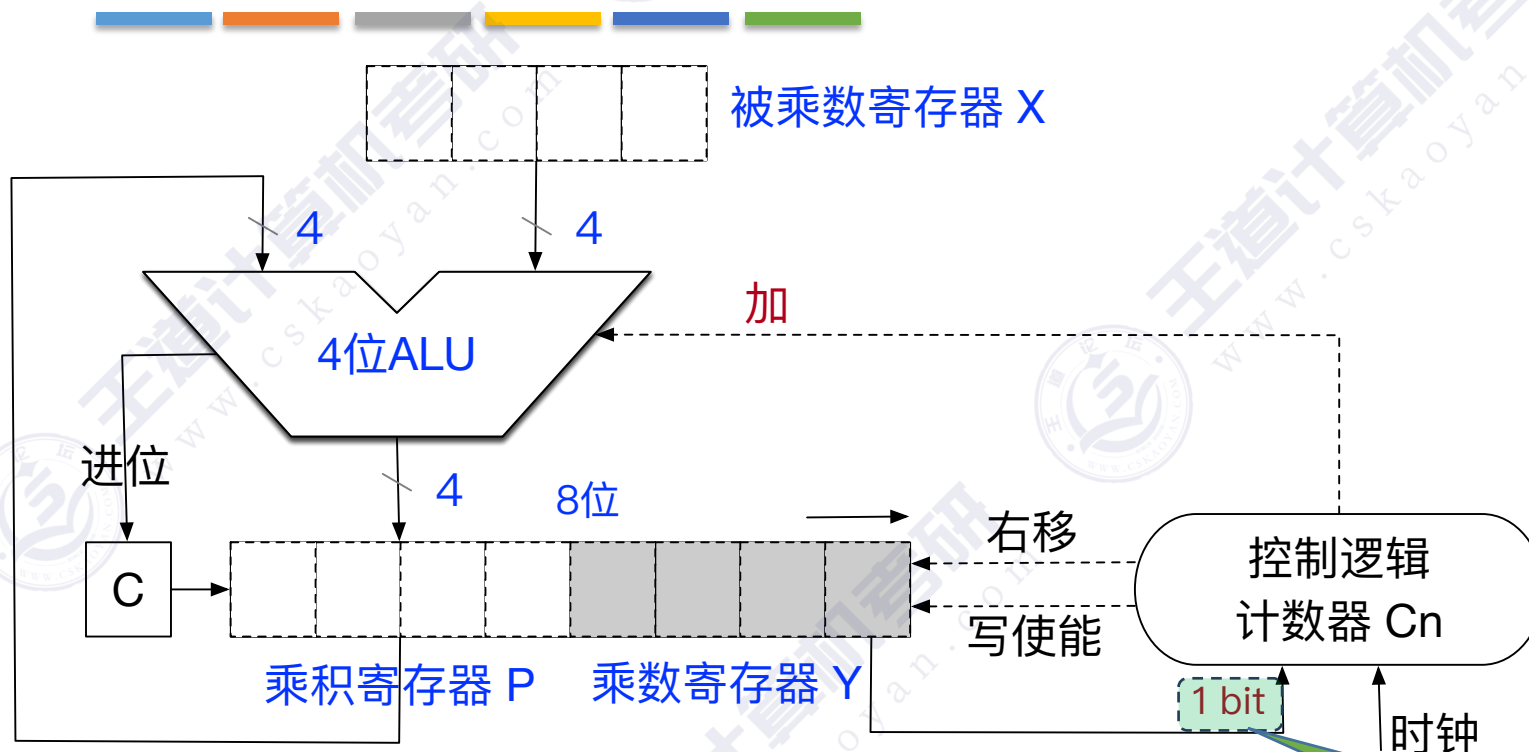
由ALU、移位器、寄存器、控制逻辑组成的乘法电路

阵列乘法器（快速乘法器中的一种）

用移位运算、加/减运算等效实现乘法

由ALU、移位器、寄存器、控制逻辑组成的乘法电路

$$\begin{array}{r} 1101 \\ \times 1011 \\ \hline 0000 \quad \text{部分积 } P_0 \\ + 1101 \\ \hline 1101 \quad \text{部分积 } P_1 \\ + 1101 \\ \hline 100111 \quad \text{部分积 } P_2 \\ + 0000 \\ \hline 100111 \quad \text{部分积 } P_3 \\ + 1101 \\ \hline 10001111 \quad \text{最终乘积 } P_4 \end{array}$$

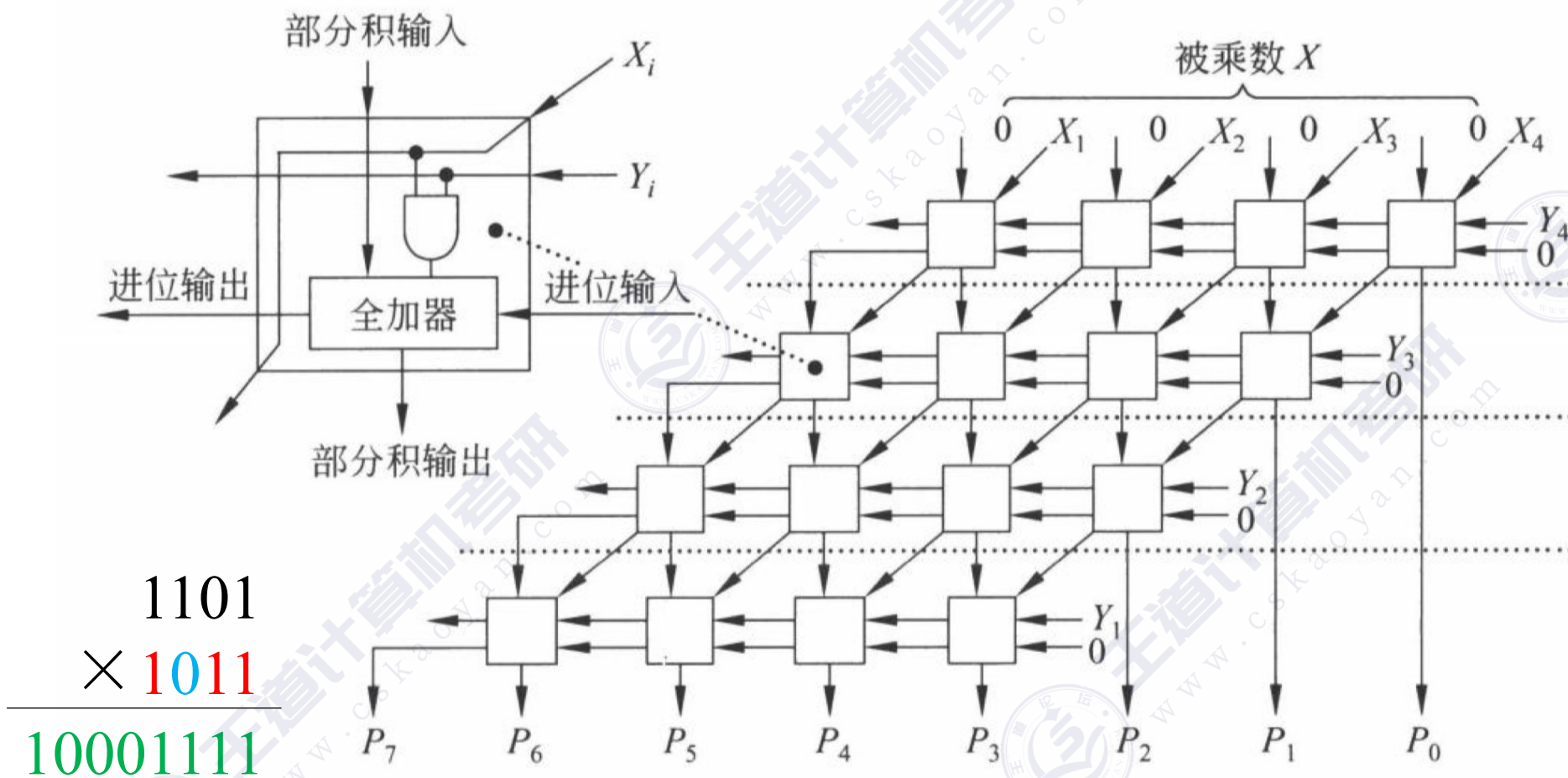


该电路实现 n bit 无符号数相乘，至少需要 n 个时钟。

改进方案：可以实现“两位乘法”，每轮处理乘数寄存器 Y 的末尾 2bit，实现，此时仅需 $n/2$ 个时钟即可完成运算

改为两位乘法：
每次处理 2bit

阵列乘法器



特点：可以在1个时钟内完成乘法运算

注：阵列乘法器是快速乘法器中的一种。很多“快速乘法器”都可以在1个时钟内完成乘法运算

上图为 4×4 位阵列乘法器（ 32×32 同理）

用逻辑运算、加/减运算等效实现乘法

思考：在一个没有乘法运算电路的计算机中，能否用其他运算等效实现乘法？

```
/*用移位运算、加法运算等效实现32bit无符号数乘法*/
unsigned int multiply_unsigned(unsigned int x, unsigned int y) {
    unsigned int result = 0;
    for (int i = 0; i < 32; i++) {
        // 提取乘数 y 的最低位
        unsigned int bit = y & 1;
        // 如果当前位为 1, 加上被乘数 x
        if (bit) {
            result += x;
        }
        // 被乘数 x 左移一位, 乘数 y 右移一位
        x <<= 1;
        y >>= 1;
    }
    return result;
}
```

仅用到逻辑运算（位运算、移位运算），加/减运算

优点：在没有乘法运算电路、不支持乘法指令的计算机中，也可以等效实现乘法效果

缺点：运算速度很慢（在非流水线计算机中，每条指令的执行都至少需要1个时钟）

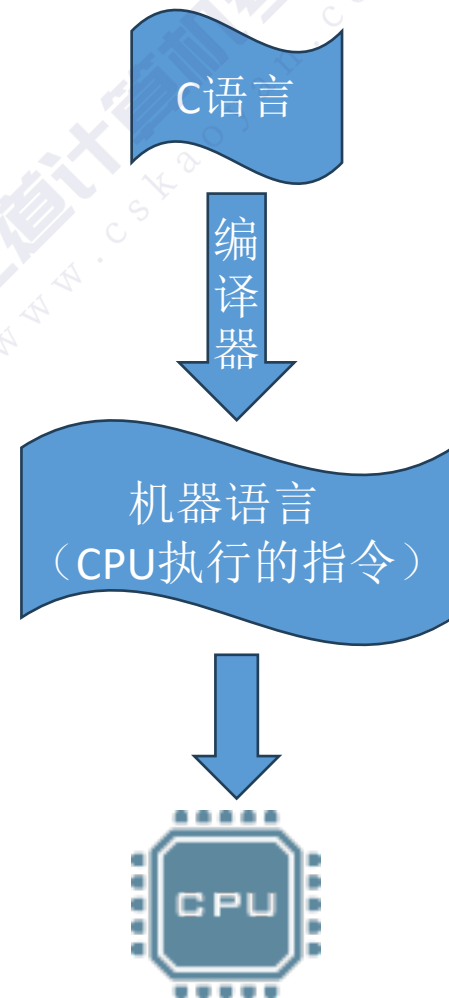
真题训练：2020年_43

43. (13 分) 有实现 $x \times y$ 的两个 C 语言函数如下：

```
unsigned umul(unsigned x, unsigned y) { return x*y; }  
int imul(int x, int y) { return x * y; }
```

假定某计算机 M 中 ALU 只能进行加减运算和逻辑运算。请回答下列问题。

- 1) 若 M 的指令系统中没有乘法指令，但有加法、减法和位移等指令，则在 M 上也能实现上述两个函数中的乘法运算，为什么？
- 2) 若 M 的指令系统中有乘法指令，则基于 ALU、位移器、寄存器以及相应控制逻辑实现乘法指令时，控制逻辑的作用是什么？
- 3) 针对以下三种情况：①没有乘法指令；②有使用 ALU 和位移器实现的乘法指令；③有使用阵列乘法器实现的乘法指令，函数 `umul()` 在哪种情况下执行时间最长？哪种情况下执行时间最短？说明理由
- 4) n 位整数乘法指令可保存 $2n$ 位乘积，当仅取低 n 位作为乘积时，其结果可能会发生溢出。当 $n = 32$, $x = 2^{31} - 1$, $y = 2$ 时，带符号整数乘法指令和无符号整数乘法指令得到的 $x \times y$ 的 $2n$ 位乘积分别是什么（用十六进制表示）？此时函数 `umul()` 和 `imul()` 的返回结果是否溢出？对于无符号整数乘法运算，当仅取乘积的低 n 位作为乘法结果时，如何用 $2n$ 位乘积进行溢出判断？



结论（小题、简答题常考）

在计算机内，实现乘法运算的三种常见方式对比：

- 阵列乘法器（快速乘法器中的一种）的运算速度最快——可1个时钟内完成运算
- 由 ALU、移位器、寄存器、控制逻辑 组成的乘法电路运算速度较快——通常需多个时钟才能完成运算
- 可以使用移位运算、加/减运算 等效实现乘法，但运算速度最慢

附件：用移位运算、加减运算等效实现乘法（可执行）

```
#include <stdio.h>

/*用移位运算、加法运算等效实现32bit无符号数乘法*/
unsigned int multiply_unsigned(unsigned int x, unsigned int y) {
    unsigned int result = 0;
    for (int i = 0; i < 32; i++) {
        // 提取乘数 y 的最低位
        unsigned int bit = y & 1;
        // 如果当前位为 1, 加上被乘数 x
        if (bit) {
            result += x;
        }
        // 被乘数 x 左移一位, 乘数 y 右移一位
        x <<= 1;
        y >>= 1;
    }
    return result;
}

int main() {
    unsigned int x, y;
    printf("王道友, 请输入两个无符号整数 x 和 y: ");
    scanf("%u %u", &x, &y);

    unsigned int product = multiply_unsigned(x, y);
    printf("结果: %u\n", product);

    return 0;
}
```