

题目 1. 下面是两种晶片生产技术的说明:

A: 使用直径 15cm 的晶圆(每晶圆成本为 12 元), 可制作 84 枚晶片, 其缺陷参数为  $0.020$  个 /  $\text{cm}^2$ 。

B: 使用直径 20cm 的晶圆(每晶圆成本为 15 元), 可制作 100 枚晶片, 其缺陷参数为  $0.031$  个 /  $\text{cm}^2$ 。

(1) 分别求出每种技术下的工艺良率:

(2) 分别求出每种技术下晶片的单位价格。

$$(1) A: \frac{1}{(1 + (0.02 \times \pi \times (15 \div 2)^2 \div 2))^2} = 13.07\%$$

$$B: \frac{1}{(1 + (0.031 \times \pi \times (20 \div 2)^2 \div 2))^2} = 2.91\%$$

$$(2) A: 12 \div 84 \div 13.07\% \approx 1.09 \text{ (元)}$$

$$B: 15 \div 100 \div 2.91\% \approx 5.15 \text{ (元)}$$

题目 2. 计算机体系结构的 8 个伟大思想之三是“加速经常性事件”。

假设某 CPU 只支持加法、乘法和跳转三种指令。经统计, 执行某基准程序时, CPU 执行这三种指令的时间占比分别为 70%、21%、9%。

(1) 若只优化加法指令, 使其执行速度提升 25%, 则 CPU 整体的平均运行时间降低为原来的多少?

(2) 若只优化乘法指令, 使 CPU 整体的平均运行时间变得与 (1) 中一样, 则需要使乘法指令的执行速度提升多少?

(3) 若只优化跳转指令, 能否使得 CPU 整体的平均运行时间降低到 (1) 中水平? 请通过计算证明你的答案。

$$(1) n = \frac{70\%}{1+25\%} + 30\% = 86\%$$

$$(2) n = 70\% + \frac{21\%}{1+x} + 9\% = 86\%$$

$$\text{得 } x = 200\%$$

$$(3) n = 70\% + 21\% + \frac{9\%}{1+x} = 86\%$$

$$\text{得 } x = -280\% < 0$$

故不能只优化跳转指令

题目 3. 用性能公式的子集作为性能评价的指标是片面的, 不准确的。假设我们有两种 CPU, 其中  $C_1$  的时钟频率为 5 GHz,  $C_2$  的时钟频率为 3 GHz。某基准程序 P 在  $C_1$  上的 CPI 为 2, 需要执行  $5 \times 10^9$  条指令; 在  $C_2$  上的 CPI 为 1.8, 需要执行  $3.3 \times 10^9$  条指令。

(1) 试用以上 CPU 和基准程序证明, “时钟频率越高, 性能越好”的观点是错误的;

(2) 试用以上 CPU 和基准程序证明, “MIPS 越高, 性能越好”的观点也是错误的;

(3) 请思考: 为什么相同的程序在不同的 CPU 上执行的指令条数可能不同呢?

$$(1) \frac{2 \times 5 \times 10^9}{5} > \frac{1.8 \times 3.3 \times 10^9}{3}$$

故  $C_1$  的频率高但性能不如  $C_2$ , 得证

(2) 同上,  $C_1$  的 MIPS 大于  $C_2$ , 但性能不如  $C_2$ . 得证

(3) 因为不同 CPU 指令集不同, 导致指令条数可能不同

题目 4. 如下是一道考研真题:

(2021, 12) 2017 年公布的全球超级计算机 TOP 500 排名中, 我国“神威·太湖之光”超级计算机蝉联第一, 其浮点运算速度为 93.0146 PFLOPS, 说明该计算机每秒钟内完成的浮点操作次数约为 ( ) D

- A.  $9.3 \times 10^{13}$  次
- B.  $9.3 \times 10^{15}$  次
- C. 9.3 千万亿次
- D. 9.3 亿亿次

实验题 1. Verilog OJ 平台 <https://verilogoj.ustc.edu.cn/oj/> 上第 47 题【双边沿检测】。在作业中请不要附上代码, 而是提交带 id 的 AC 截图。AC 截图可以作为图片单独上传, 也可以放入作业的 PDF 文件中一起上传。

Status

Welcome, 2202111026

### 提交结果

1/1 个通过测试用例, 获 10/10 分

状态: Accepted

提交时间: 几秒前

测试用例 0 (行为模仿真): Accepted

10/10 分 >

Code:

Copy

Download

```
1 module top_module (
2     input clk,
3     input in,
4     output out
5 );
6     reg button1;
7     reg button2;
8     always@(posedge clk)
9         button1 <= in;
10    always@(posedge clk)
11        button2 <= button1;
12    assign out = (button1 & ~button2) | (~button1 & button2);
13 endmodule
```