2024电子技术与系统期末试题解析

24级AI学组

一、不定项选择题

1. B【资料原文】
2. ACD【资料原文】
3. A
4. B【两个复合管都能起作用，就看输入连的是哪个管子，组合就是哪个类型的】
5. D【滤波电路的作用是去掉交流成分，虽然还有纹波，不是完全的直流，但是频率没有变化，不是高频变低频；变成方波需要滞回比较器；交流变直流需要整流电路】
6. 与或式：A′B′C′+A′BC′+AB′C′+AB′C+ABC 或与式：(A+B+C′)(A+B′+C′)(A′+B′+C)
7. D【延迟的原因包括电容充电】
8. ACD【B应该是 y=~a，按位取反】
9. D【并行大多数情况不改变延迟，不会增加延迟。注意延迟的含义是第一个任务的延迟】
10. D【浮点数就是科学计数法】
11. AC【B：代码密度是CISC的优势，D：RISC通常拥有大量通用寄存器】
12. AB【C: 数据处理指令不一定是3操作数，比如cmp,mov; D: 多种寻址方式违反了准则1】
13. BD【每个任务的延迟确实会因为等待而增加；效率与最大单级延迟有关】
14. ABD
15. AB【C和D不是虚拟存储器的直接作用，反而虚拟地址翻译可能带来额外开销】

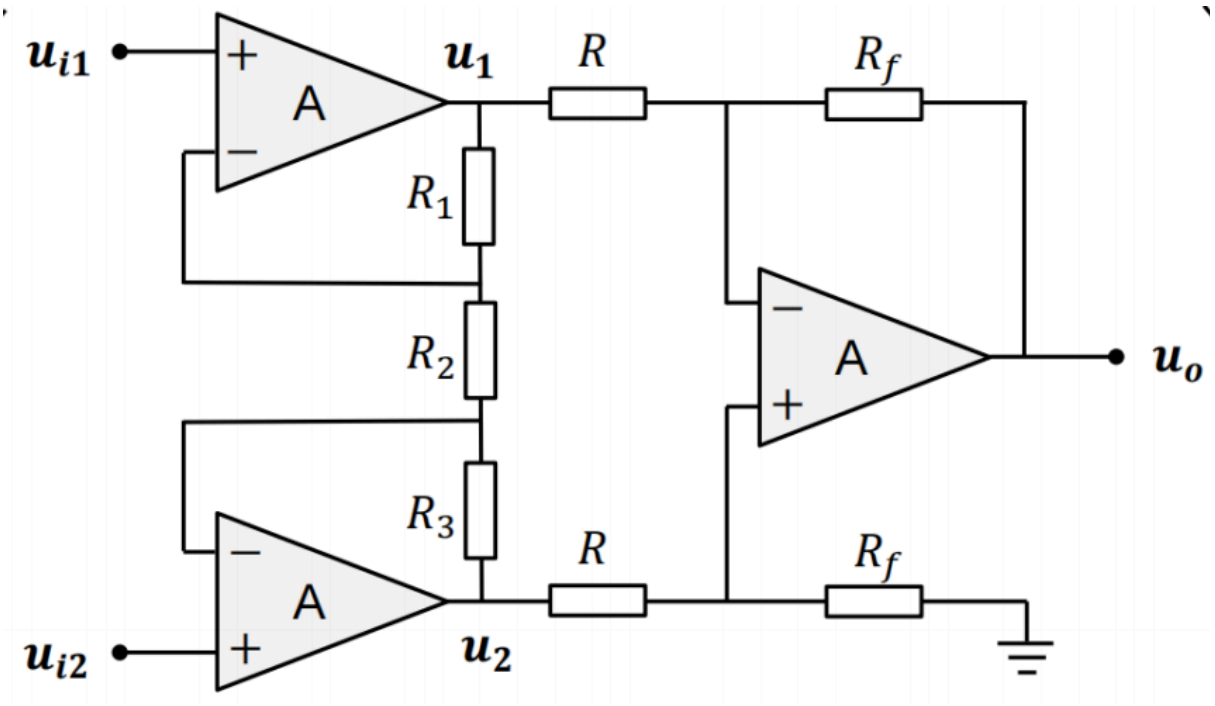
二、问答画图题

1. 自激振荡电路的四个部分：放大电路，稳伏环节，正反馈网络，选频网络。作用如下：
    a. 放大电路保证电路能够有从起振到动态平衡的过程，使电路获得一定幅值的输出量
    b. 正反馈网络使放大电路的输入信号等于反馈信号

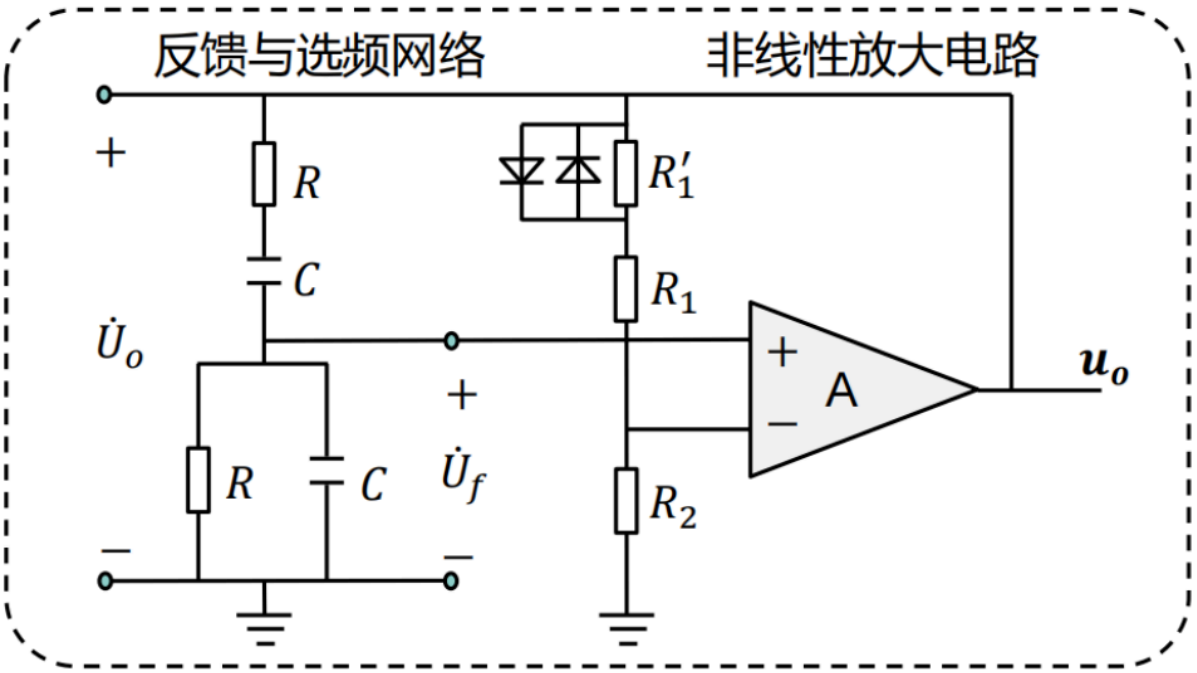
c. 稳伏环节是放大电路的非线性环节，稳定输出信号幅值

d. 选频网络确定电路的振荡频率，选择单一频率起振

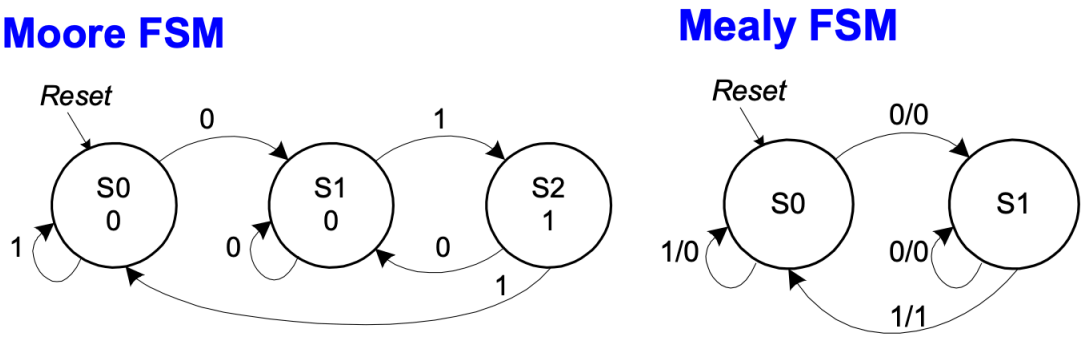
2. 仪表运放电路，PPT原图



3. 文氏桥选频的自激振荡电路，PPT原图



4. 定义：(A+B)′ = A′B′，(AB)′ = A′+B′。表达式为 Y = (A′+B′)(C′+D′)+E′。注意推气泡法的要求是非门只能出现在输入，并且经过的与门要变成或门，或门也要变成与门。
5. Moore 状态机的输出由当前状态决定，Mealy 状态机的输出则由当前状态和输入共同决定。PPT 例子：判断是否连续两个输入为 1:



6. 冲突类型1：数据冲突

   原因：`ADD` 指令在 EX (周期4) 阶段才计算出 R0 的结果，并在 WB (周期6) 阶段才写回寄存器堆。而紧随其后的 `LDR` 指令需要在 EX (周期5) 阶段使用 R0 作为基地址来计算内存地址（或者在 ID 阶段读取 R0）。由于 `LDR` 试图读取 R0 时，`ADD` 还没有将结果写回，导致 “读后写” (Read After Write) 冲突。

   解决方案：

   1. 数据前递：
      原理：硬件直接将 `ADD` 指令在 ALU 阶段计算出的结果，通过旁路（Bypass network）直接传输给 `LDR` 指令的 ALU 输入端。
   2. 指令调度：

      原理：编译器在不改变程序逻辑的前提下，将第2条和第3条指令之间插入一条无关指令（例如将第1条 `MOV` 挪到中间，或者找其他无依赖指令）

   冲突类型2：控制冲突

   原因：`BNZ` 是条件跳转指令，它是否跳转取决于 `SUBS` 产生的标志位（Zero Flag）。`BNZ` 在 ID (译码) 阶段完成解析，但此时 `SUBS` 可能刚执行完甚至还在执行中。流水线为了保持满载，通常会在知道是否跳转之前就预取了下一条指令。如果跳转发生，预取的指令就是错误的。

   解决方案：

   1. 分支预测：

      原理：硬件通过历史记录（动态预测）或约定规则（静态预测，如“总是由于”或“总是不跳”）来猜测分支方向

2. 不知道，真不会做了。

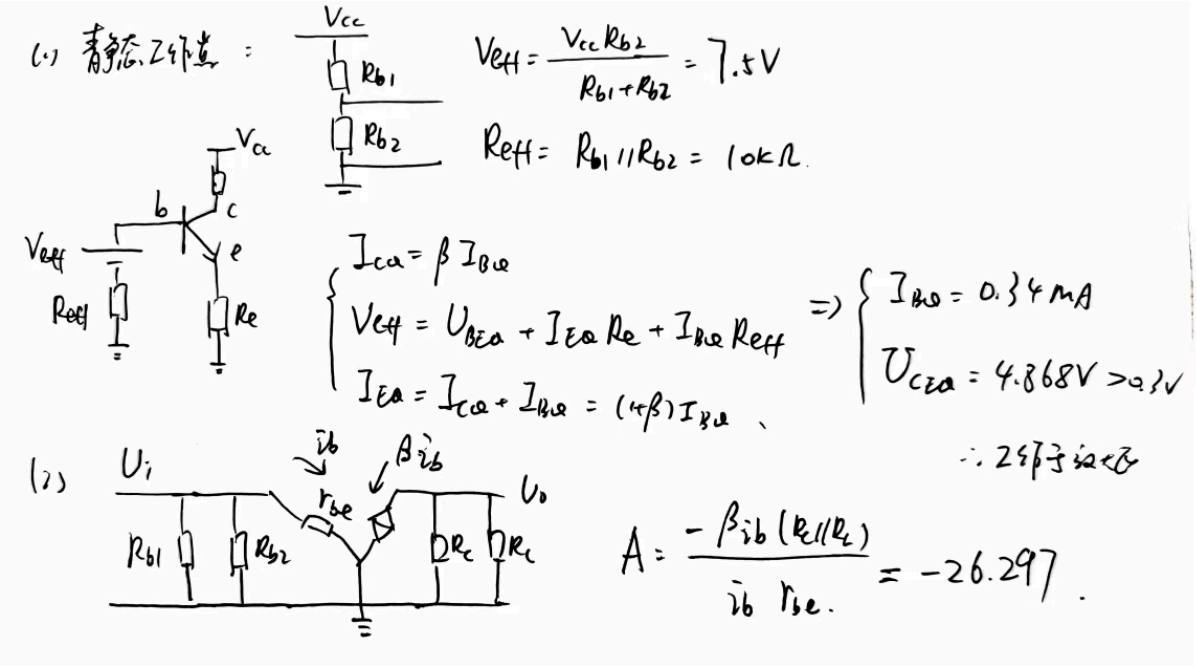
7. 直接映射：每个主存块只能映射到Cache中的一个固定位置。优：硬件简单，成本低，判断速度快。缺：冲突率高（抖动），灵活性差。

   全相联映射：主存块可以映射到Cache中的任意位置。优：冲突率最低，灵活性最高。缺：比较电路复杂（需要并发比较所有Tag），硬件成本高，速度慢。

   组相联映射：将Cache分组，主存块映射到固定组，但可存入组内任意块。优：折中方案，兼顾了冲突率和硬件复杂度。缺：设计复杂度介于两者之间。

三、计算题

1. 作业原题。使用戴维南等效可以简化计算，注意动态电路时电容视作导线，Re不起作用。



2. (1) 13.5625 = 2^3 + 2^2 + 2^0 + 2^-1 + 2^-4，因此其定点数表示为 0b0000110110010000；42.3125=2^5 + 2^3 + 2^1 + 2^-2 + 2^-4，因此其定点数表示为 0b0010101001010000。相加得到：0b0011011111100000，即 0x37E0

   (2) 两个浮点数分别为 0b0(01111111)(1000000...) 和 0b0(10000000)(1010000...)。注意指数位不一样，注意指数位有 127 的偏移量，可知两数分别为 (1.1) 和 (11.01)，相加得到 (100.11)。所以尾数为 (0011)，指数为 129，最终结果是 0b0(10000001)(0011000...)。也即 0x40980000。

3. （1）由于是数据的AMAT，只考虑D-cache

   AMAT = HitTime + MissRate *MissPenalty = 1ns + 0.08 * 50ns = 5ns

   （2）LDR：5 + (0.08 * 50) = 5 + 4 = 9 周期

STR：4 + (0.08 * 50) = 4 + 4 = 8 周期

（3）权重 CPI = Ratio_i * Cycles_i

Total CPI = 1.35 + 1.6 + 0.3 + 2.2 = 5.45

（4）**New CPI** = 5.45 + 0.05 * 50 = 7.95

四、编程题

1. 这是 Moore 型状态机，并且有 3 个状态。首先列出状态转换表：

State	Input	NextState
S0	A	S1
S0	A′	S0
S1	B	S2
S1	B′	S0
S2	X	S0

由此，我们不妨采用 one-hot 编码，就可以得到状态输出表（真值表）：

(S0,S1,S2)	A	B	(S0′,S1′,S2′)	Q
(1,0,0)	1	X	(0,1,0)	0
(1,0,0)	0	X	(1,0,0)	0
(0,1,0)	X	1	(0,0,1)	0
(0,1,0)	X	0	(1,0,0)	0
(0,0,1)	X	X	(1,0,0)	1

列出布尔表达式如下

- S0′ = S0A′ + S1B′ + S2
- S1′ = S0A
- S2′ = S1B
- Q = S2

最后附上实现代码：

```
module FSM(input logic clk, input logic rst, input logic A, input logic B, outp
ut logic Q);
    typedef enum logic [1:0] { S0,S1,S2 } statetype;
    statetype state, next_state;
```

```
    always_ff @(posedge clk) begin
        if (rst) state <= S0;
        else state <= next_state;
    end

    always_comb begin
        case (state)
            S0: if (A) next_state = S1;
                else next_state = S0;
            S1: if (B) next_state = S2;
                else next_state = S0;
            S2: next_state = S0;
            default: next_state = S0;
        endcase
    end
    assign Q = (state == S2);
endmodule
```

2. C code

```
for (i=0; i<100; i++) {
    if (b[i] < a[i]) z[i] = a[i] - b[i];
    else z[i] = b[i] - a[i];
}
```

```
    MOV R0, #0              ; i = 0

LOOP
    ; --- 1. 循环条件检查 ---
    CMP R0, #100
    BGE DONE

    ; --- 2. 加载数组元素 a[i] 和 b[i] ---
    ; 使用【寄存器移位】寻址模式: [Base, Offset, LSL #2]
    ; int占4字节，所以地址偏移量 = i * 4 (即 i 左移 2 位)
    LDR R4, [R1, R0, LSL #2]  ; R4 = a[i]
    LDR R5, [R2, R0, LSL #2]  ; R5 = b[i]
```

```
    ; --- 3. 比较 b[i] 和 a[i] ---
    CMP R5, R4

    ; --- 4. 使用条件执行指令计算绝对值 ---
    SUBLT R6, R4, R5        ; if (b < a) R6 = a[i] - b[i]

    SUBGE R6, R5, R4

    ; --- 5. 存储结果 z[i] ---
    STR R6, [R3, R0, LSL #2]

    ; --- 6. 循环变量自增并跳转 ---
    ADD R0, R0, #1
    B LOOP

DONE
```