LSP 考试 2025 年 6 月 19 日

考试信息

- 日期：2025 年 6 月 19 日
- 语言：捷克语
- 类型：标准考试
- 已核对 **PDF** 官方答案

第 1 题 - RS 锁存器仿真 (RS Latch Simulation) (5 分)

题目： 输入 A、B、C 在时间 t0 到 t4 的值如图所示。写出 X 和 Y 输出的值。[**English**] Inputs A, B, C had values shown in the figure at times t0 to t4. Write the values of X and Y outputs.

输入序列：

```
A = 0 | 1 | 0 | 1 | 1
B = 1 | 0 | 1 | 1 | 1
C = 1 | 0 | 1 | 1 | 0
    t0  t1  t2  t3  t4
```

官方参考答案 (Official Answer):

```
X = 0 | 0 | 0 | 0 | 1
Y = 1 | 0 | 1 | 1 | 0
```

解题步骤： 1. **t0**: A=0, B·C=1 → Set 触发，Y 保持或变化 2. **t1**: A=1 → Reset 触发 3. **t2**: A=0, B·C=1 → Set 4. **t3**: A=1, B·C=1 → 冲突，看优先级 5. **t4**: A=1, B·C=0 → Reset 持续

注意： 需要根据考卷上的具体电路图来分析！

第 2 题 - Shannon 展开 (Shannon Expansion) (6 分)

题目： 将第 1 题电路的函数 X=f(A,B,C,X) 用 Shannon 展开分解为：X=(not X and f0(A,B,C)) or (X and f1(A,B,C))。[**English**] Decompose the function X=f(A,B,C,X) from question 1 into the form X=(not X and f0(A,B,C)) or (X and f1(A,B,C)) using Shannon expansion.

官方参考答案 (Official Answer) - 卡诺图：

```
f0:      B              f1:      B
       A  0  1                 A  0  1

C 0   0    0    1   0   C 0   0    0    1   0
  1   0    0    0   0     1   0    0    0   0
```

做题方法： 1. 从电路推导 X = f(A, B, C, X) 2. 令 X=0 求 f，令 X=1 求 f 3. 用卡诺图表示结果（必须!）

假设电路表达式为 $\mathbf{X = NOR(A, Y)}$ 且 $\mathbf{Y = NOR(B{\cdot}C, X)}$：

将 Y 代入：X = NOT(A OR Y) = NOT(A OR NOT(B·C OR X))

对 X 展开： - 当 X=0 时：f (A,B,C) = NOT(A OR NOT(B·C OR 0)) = NOT(A OR NOT(B·C)) = NOT A ·(B·C) - 当 X=1 时：f (A,B,C) = NOT(A OR NOT(B·C OR 1)) = NOT(A OR 0) = NOT A

Shannon 展开结果：

```
X = (NOT X AND f (A,B,C)) OR (X AND f (A,B,C))
```
其中：
```
f  = NOT A AND B AND C
f  = NOT A
```

考试技巧： 先从电路推导 X=f(A,B,C,X) 的完整表达式，再分别令 X=0 和 X=1 化简。

第 3 题 - 等价逻辑函数 (Equivalent Logic Functions) (4 分)

题目： 勾选所有与其他函数等价的逻辑函数。[**English**] Check all logic functions that have another equivalent function here.

```
y1 <= (A or B) and (not A or C);
y2 <= B or (A and C and B) or (not A and B and D);
y3 <= (not A and B) or (A and not B) or (C and B);
y4 <= (C and B) or (not A xor not B);
```

AI 参考解析 (AI Solution):

解题方法：用布尔代数化简或真值表 (Boolean Algebra / Truth Table)

化简 y1：

```
y1 = (A + B)(Ā + C)
   = AĀ + AC + BĀ + BC
   = AC + ĀB + BC        ( AĀ = 0)
```

化简 y2：

```
y2 = B + ABC + ĀBD
   = B(1 + AC + ĀD)
   = B                    ( 1 + x = 1)
```

化简 y3：

```
y3 = ĀB + AB + CB
   = A  B + CB           (异或 XOR)
```

化简 y4：

```
y4 = CB + (Ā  B)
   = CB + (A  B)         ( Ā  B = A  B)
   = CB + ĀB + AB
```

比较结果： - y1 = AC + ĀB + BC - y2 = B - y3 = ĀB + AB + CB = A B + CB - y4 = CB + ĀB + AB = A B + CB

答案：y3　y4 (两者等价)

考试技巧： - 使用 Ā B = A B 这个恒等式 - 若表达式复杂，可画卡诺图 (Karnaugh Map) 验证

第 4 题 - 8 位寄存器运算 (8-bit Register Arithmetic) (2 分)

题目： 将某运算结果的低位存入 1 字节寄存器，作为 8 位数的十进制值是多少？[**English**] If we store the lower bits of an operation result into a 1-byte register, what decimal value will it hold as an 8-bit number?

答案格式： - a) unsigned: ? - b) signed: ?

AI 参考解析 (AI Solution):

8 位寄存器运算原理 (8-bit Arithmetic): - 8 位无符号范围: 0 ~ 255 - 8 位有符号范围: -128 ~ 127 (补码表示) - 溢出处理: result mod 256

通用解题步骤: 1. 计算原始结果 R 2. 无符号值 = R mod 256 3. 有符号值: 若 R mod 256 128, 则有符号值 = (R mod 256) - 256

常见例题: | 运算 | 结果 | mod 256 | unsigned | signed | |---|---|---|---|---| | 255+253+251 | 759 | 247 | 247 | -9 | | 128+128 | 256 | 0 | 0 | | 100+100+100 | 300 | 44 | 44 | 44 |

补码转换公式 (Two's Complement):

```
若 unsigned_val  128:
    signed_val = unsigned_val - 256
否则:
    signed_val = unsigned_val
```

注意: 题目需要具体运算表达式才能计算，请根据考题给出的数值代入上述方法。

第 5 题 - Moore/Mealy 自动机定义 (Finite State Machine Definition) (3 分)

题目: 补全定义——必须数学上精确! [**English**] Complete the definition - it must be mathematically precise!

定义: Automat Moore (Mealy) je uspořádaná šestice M = < X, S, Z, , , s S >

- **X** 是有限输入字母表 / is a finite input alphabet
- **S** 是有限状态集合 / is a finite set of states
- **Z** 是有限输出字母表 / is a finite output alphabet
- 是状态转移函数 / is the state transition function: Moore: S×X→S, Mealy: S×X→S
- 是输出函数 / is the output function: Moore: S→Z, Mealy: S×X→Z
- **s** 是初始状态 / is the initial state

AI 参考解析 (AI Solution):

Moore 与 Mealy 自动机的关键区别 (Key Differences):

特性	Moore	Mealy
输出函数	$S \to Z$	$S \times X \to Z$
输出依赖	仅状态	状态 + 输入
输出时机	状态变化后	输入到达时
状态数量	通常更多	通常更少

完整定义 (Complete Definition):

```
Moore自动机: M = <X, S, Z,  ,  , s >
   : S × X → S      (状态转移函数)
   : S → Z          (输出仅取决于当前状态)

Mealy自动机: M = <X, S, Z,  ,  , s >
   : S × X → S      (状态转移函数)
   : S × X → Z      (输出取决于当前状态和输入)
```

记忆口诀: - Moore = "More states"→ 输出只看状态 - Mealy = "More efficient"→ 输出看状态 + 输入

考试重点: 必须精确写出 和 的定义域和值域!

第 6 题 - 加法器设计 (Incrementer Design) (7 分)

题目： 加法器可以不用全加器，用门电路更简单地实现。画出由门组成的电路图。[**English**] The adder can be implemented much simpler using gates without a full adder. Draw its schematic composed of gates.

AI 参考解析 (AI Solution):

+1 加法器 (Incrementer) 的设计原理：

对于 n 位 +1 加法器，无需全加器，只需 XOR 和 AND 门：

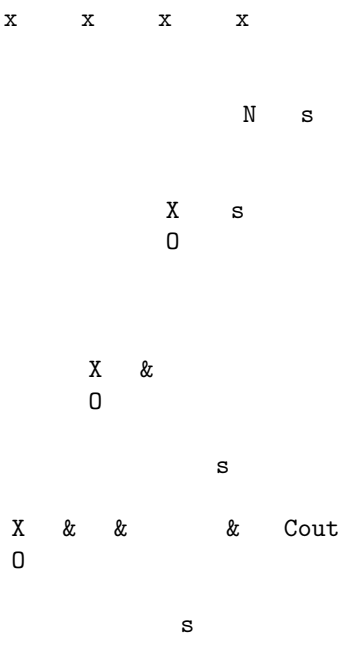
公式推导 (Formula Derivation)：

```
s  = x   1 = NOT x
s  = x   x
s  = x   (x  · x )
s  = x   (x  · x  · x )
...
s  = x   (x   · x   · ... · x )
```

进位输出 (Carry Out)：

```
Cout = x  · x  · x  · x   (当所有位都为1时产生进位)
```

4 位 +1 加法器电路图：



所需门电路： - 1 个 NOT 门 (或 XOR 门接 VCC) - 3 个 XOR 门 - 3 个 AND 门 (级联结构)

考试技巧： 记住模式 `s = x XOR (进位链)`，进位链是所有低位的 AND

第 7 题 - VHDL 位操作 (VHDL Bit Manipulation) (7 分)

题目： 用并发 (concurrent) VHDL 代码最优地描述图中电路，不使用顺序语句。整个架构只用一条语句可得满分，每多一条扣 1 分。[**English**] Optimally describe the circuit in the figure using concurrent VHDL code without sequential statements. Full points for using only one statement in the architecture block; one point deducted for each additional statement.

```vhdl
library ieee; use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Test20250619q7 is
  port(x: in std_logic_vector(3 downto 0);
       y: out std_logic_vector(3 downto 0));
end entity;
architecture rtl of Test20250619q7 is
begin
  -- 填写代码
end architecture rtl;
```

AI 参考解析 (AI Solution):

常见单行 VHDL 解答模式 (Common Single-Statement Patterns):

1. Gray 码转换器 (Gray Code Converter):

```vhdl
y <= ('0' & x(3 downto 1)) xor x;
```

解释: 右移一位后与原值异或 = Binary to Gray

2. 位翻转 (Bit Reversal):

```vhdl
y <= x(0) & x(1) & x(2) & x(3);
```

3. 循环左移 (Rotate Left):

```vhdl
y <= x(2 downto 0) & x(3);
```

4. 循环右移 (Rotate Right):

```vhdl
y <= x(0) & x(3 downto 1);
```

5. 取反 (Invert):

```vhdl
y <= not x;
```

6. 符号扩展 (Sign Extension):

```vhdl
y <= (others => x(3));  -- 全部填充符号位
```

VHDL 操作符优先级 (Operator Precedence): | 优先级 | 操作符 | |——–|——–| | 最高 | not, abs | | | *, /, mod, rem | | | +, -, & | | | =, /=, <, >, <=, >= | | | and, or, xor, nand, nor, xnor |

考试技巧: 观察电路输入输出关系，找出位操作模式（移位、异或、拼接等）

第 8 题 - VHDL 电路分析 (VHDL Circuit Analysis) (7+1 分)

题目: 根据以下代码, 使用门、多路选择器、+1 加法器、比较器和 DFF 寄存器的符号画出方框图。[**English**] Draw the block diagram for the circuit described by the following code, using symbols for gates, multiplexor, +1 adder, comparator, and DFF registers.

AI 参考解析 (AI Solution):

VHDL 到方框图的映射规则 (VHDL to Block Diagram Mapping):

VHDL 代码	对应元件
`if rising_edge(clk)`	DFF 寄存器 (D Flip-Flop)
`when...else` / `with...select`	多路选择器 (MUX)
`x + 1`	+1 加法器 (Incrementer)
`x = y` / `x < y`	比较器 (Comparator)

VHDL 代码	对应元件
and/or/not	逻辑门 (Logic Gates)

常见电路模式识别:

1. 计数器 (Counter):

```
process(clk)
begin
  if rising_edge(clk) then
    if cnt = MAX then cnt <= 0;
    else cnt <= cnt + 1;
    end if;
  end if;
end process;
```

元件: DFF + +1 加法器 + 比较器 + MUX

2. 移位寄存器 (Shift Register):

```
process(clk)
begin
  if rising_edge(clk) then
    reg <= reg(N-2 downto 0) & din;
  end if;
end process;
```

元件: 级联 DFF

3. 状态机 (FSM):

```
process(clk)
begin
  if rising_edge(clk) then
    state <= next_state;
  end if;
end process;
```

元件: DFF (状态寄存器) + 组合逻辑

画图步骤 (Drawing Steps): 1. 找出所有时序元件 (有 `rising_edge` 的信号) → 画 DFF 2. 识别算术运算 → 画加法器/比较器 3. 识别选择结构 → 画 MUX 4. 连接所有元件, 标注信号名

注意: 具体答案需要看题目给出的 VHDL 代码

第 9 题 - 分支预测 (Branch Prediction) (8 分)

非考点提示 (Not on Exam): 根据 2026 年 1 月考试说明, 分支预测器本次不考, 可战略性跳过。

题目: C 程序处理某数据。假设 for 循环编译为 do-while 形式, 处理器使用以下预测器, 计算分支预测错误次数。[**English**] A C program processes data. Assuming the for-loop is compiled as do-while and the processor uses the following predictors, calculate the number of branch mispredictions.

答案: - 1 位预测器 (初始 NT): misses = ? - 2 位预测器 (初始 WT): misses = ?

AI 参考解析 (AI Solution):

分支预测器原理 (Branch Predictor Principles):

1 位预测器 (1-bit Predictor)：- 状态：T (Taken) 或 NT (Not Taken) - 规则：预测错误就翻转状态 - 初始 NT：第一次跳转必错

2 位预测器 (2-bit Saturating Counter)：| 状态 | 预测 | T 时 | NT 时 | |—|—|—|—| | ST (Strongly Taken) | T | ST | WT | | WT (Weakly Taken) | T | ST | WN | | WN (Weakly Not-Taken) | NT | WT | SN | | SN (Strongly Not-Taken) | NT | WN | SN |

常见循环模式分析：

外循环 N 次，内循环 M 次：

```
for (i=0; i<N; i++)       // 外循环：N 次 T + 1 次 NT
  for (j=0; j<M; j++)     // 内循环：每轮 M 次 T + 1 次 NT，共 N 轮
```

1 位预测器 miss 计算：- 外循环：首次入循环 miss + 最后退出 miss = 2 次 - 内循环：每轮首次 miss + 每轮退出 miss = N×2 次 - **总计：2 + 2N 次 miss**

2 位预测器（初始 WT）miss 计算：- 外循环：最后退出 miss = 1 次 - 内循环：每轮退出从 ST→WT 不算 miss，最后退出才 miss - 规律：内循环每轮退出时 1 次 miss - **总计：1 + N 次 miss**（约为 1 位的一半）

典型例题：i<5, j<2000 - 1 位：2 + 2×5 = 12 次 miss - 2 位：1 + 5 = 6 次 miss

查找最小值程序分析：

```
for (i=1; i<n; i++)
  if (a[i] < min) min = a[i];  // 条件分支
```

- 外循环：同上
- if 分支：取决于数据，平均约 log (n) 次更新

考试技巧：画出预测器状态转移图，逐次跟踪预测结果
