2025-06-19 LSP Exam Solution / Řešení zkoušky / 考试解析

Course: B0B35LSP – Logické systémy a procesory | BE5B35LSP – Logic Systems and Processors **University**: CVUT FEL (CTU) – Czech Technical University in Prague **Keywords**: Zkouška, Exam, Test, Solutions, Vysledky, Answers, K-Map, RS Latch, Pipeline

CN Version | EN Version | CZ Version

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_0，令 X=1 求 f_1 3. 用卡诺图表示结果（必须！）

假设电路表达式为 X = NOR(A, Y) 且 Y = NOR(B·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_0(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_1(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))

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           ( 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 $\bar{A}\oplus\bar{B}=A\oplus B$ 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 | 0 | | 100+100+100 | 300 | 44 | 44 | 44 |

补码转换公式 (Two's Complement)：

```
unsigned_val  128:
   signed_val = unsigned_val - 256
:
   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 usporádaná sěstice $M = < X, S, Z, \omega, \delta, s_0 \in S >$

- **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
- **δ** 是状态转移函数 / is the state transition function：Moore: $S\times X\to S$，Mealy: $S\times X\to S$
- **ω** 是输出函数 / is the output function：Moore: $S\to Z$，Mealy: $S\times X\to Z$
- **s_0** 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ω	$S \to Z$	$S \times X \to Z$
输出依赖	仅状态	状态+输入

特性	Moore	Mealy
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  : M = <X, S, Z,  ,  , s >
   : S × X → S      (   )
   : S → Z          (     )

Mealy  : M = <X, S, Z,  ,  , s >
   : S × X → S      (   )
   : S × X → Z      (      )
```

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δ和ω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：

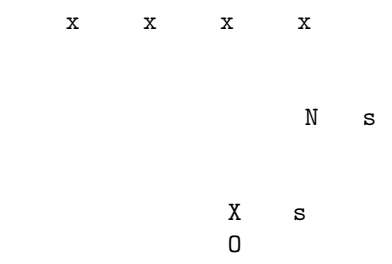
公式推导 (Formula Derivation)：

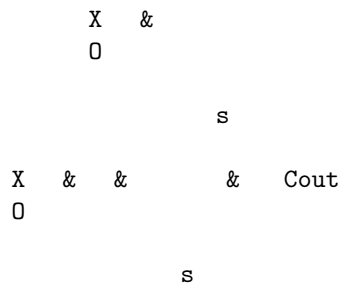
```
s  = x   1 = NOT x
s  = x   x
s  = x   (x  · x )
s  = x   (x  · x  · x )
...
s  = x   (x   · x   · ... · x )
```

进位输出 (Carry Out)：

```
Cout = x  · x  · x  · x   (   1   )
```

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
[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
  --
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):

```
y <= (others => x(3));  --
```

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)
`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++)       //   N T + 1 NT
  for (j=0; j<M; j++)     //    M T + 1 NT N
```

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 – 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];  //
```

- 外循环：同上
- if分支：取决于数据，平均约$\log_2(n)$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
