

LSP zkouška — 20. ledna 2025 (vcčetně oficiálních odpovědí)

CVUT FEL (CVUT) – České vysoké učení technické v Praze | Czech Technical University in Prague

中文版 | English | Čeština

Ověřeno podle oficiálních odpovědí v PDF

Informace o zkoušce

- Datum: 2025年1月20日
- Jazyk: 英语
- Obsahuje oficiální odpovědi

Úloha 1 – RS锁存器仿真 (RS Latch Simulation) (5分)

Zadání: 给定输入A, B, C在时间t0–t4的值, 写出X和Y输出的值 [English] Given inputs A, B, C values at times t0–t4, write the values of X and Y outputs

Vstupní sekvence:

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

Oficiální odpověď' (Official Answer): – X = 11001 (t0=1, t1=1, t2=0, t3=0, t4=1) – Y = 01100 (t0=0, t1=1, t2=1, t3=0, t4=0)

Doplňující poznámky: RS锁存器 (RS Latch) 分析步骤: 先看Reset信号A, 再看Set信号B·C

Úloha 2 – Shannon展开 (Shannon Expansion) (6分)

Zadání: 将 $X=f(A,B,C,X)$ 分解为Shannon展开形式 [English] Decompose $X=f(A,B,C,X)$ using Shannon expansion

官方答案 (卡诺图 Karnaugh Map) :

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

C 0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	1	0

Doplňující poznámky: $f_0 = f|_0$ (X为0时的函数) , $f_1 = f|_1$ (X为1时的函数)

Úloha 3 – 等价逻辑函数 (Equivalent Logic Functions) (4分)

Zadání: 勾选所有具有等价函数的逻辑函数 [English] Check all logic functions that have an equivalent function

```

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

```

Oficiální odpověď: $y1 \equiv y4$

Doplňující poznámky: 可用卡诺图或代数化简验证, 两者展开后表达式相同

Úloha 4 – 10位加法器运算 (10-bit Adder Arithmetic) (4分)

Zadání: 1023+1023+1023+1023在10位加法器上的结果 [English] Result of 1023+1023+1023+1023 on a 10-bit adder

Výpočet: $-1023 = 2^{10} - 1 = 0x3FF$ (10位全1) $- 4 \times 1023 = 4092 - 4092 \bmod 1024 = 1020$

Oficiální odpověď: – a) unsigned: $(1024-1) \times 4 = 4096-4 \equiv -4 \equiv 1020 \pmod{1024}$ – b) signed: 1023在有符号中表示-1, $4 \times (-1) = -4$

Doplňující poznámky (Two's Complement): – 10位全1 = 1023 (unsigned) = -1 (signed)
– 关键: signed解释下 $1023 = -1$

Úloha 5 – 全加器设计 (Full Adder Design) (6分)

Zadání: 设计一个全加器电路 [English] Design a full adder circuit

全加器公式 (Full Adder Formulas):

Sum = A \oplus B \oplus Cin
Cout = (A \wedge B) \vee (Cin \wedge (A \vee B))

Doplňující poznámky: – Sum用两级XOR门 – Cout = 多数表决器 (Majority Function)

Úloha 6 – 用NOR门实现XOR (XOR using NOR Gates) (5分)

Zadání: 仅使用NOR门实现XOR功能 [English] Implement XOR using only NOR gates

Oficiální odpověď:

$A \oplus B = (A \text{ nor } B) \text{ nor } ((A \text{ nor } A) \text{ nor } (B \text{ nor } B))$

Doplňující poznámky (De Morgan's Theorem): – $A \text{ nor } A = \text{NOT } A$ – 需要4个NOR门实现XOR
– $\text{XOR} = (A+B) \cdot (\bar{A}+\bar{B}) = (A+B) \cdot \text{NOT}(A \cdot B)$

Úloha 7 – Gray码转换器VHDL (Gray Code Converter) (10分)

Zadání: 用单个并发语句描述电路 [English] Describe the circuit using a single concurrent statement

Oficiální odpověď:

```

library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity Test20250120q7 is
  port(x: in std_logic_vector(3 downto 0);
        y: out std_logic_vector(3 downto 0));
end entity;
architecture rtl of Test20250120q7 is
begin
  y <= ('0' & x(3 downto 1)) xor x; -- Binary to Gray Code
end architecture rtl;

```

Doplňující poznámky (Gray Code Conversion): – 二进制转格雷码: $G = B \text{ XOR } (B \gg 1)$ –
 '0' & x(3 downto 1) = 右移1位并在高位补0 – 格雷码特性: 相邻值只有1位不同

Úloha 8 – 分支预测器和Cache (Branch Prediction & Cache) (10分)

Mimo rozsah zkoušky: 根据2026年1月考试说明, 分支预测器和Cache相关内容本次不考, 整题可战略性跳过。

Zadání: 分析程序的分支预测和Cache miss [English] Analyze the branch prediction and cache miss of the program

程序:

```

int i, j; double arr[2000];
for (i=0; i<5; i++) {
  for (j=0; j<2000; j++) arr[j]++;
}

```

A) 分支预测器 (Branch Predictor)

Zadání: 计算1位和2位分支预测器的miss次数 [English] Calculate the number of misses for 1-bit and 2-bit branch predictors

Oficiální odpověď: – 1位预测器 (初始NT) : 2 (外循环) + 5×2 (内循环) = 12次miss – 2位预测器 (初始WT) : 1 (外循环) + 5×1 (内循环) = 6次miss

Doplňující poznámky: – 1位: 每个循环入口miss + 出口miss = 2次 – 2位: 容忍一次预测错误, 仅出口miss = 1次

B) Cache Miss计算 (Cache Miss Calculation)

Zadání: 64位处理器, 直接映射 (Direct-Mapped), 32KB cache, 块长4字 (4-word block) [English] 64-bit processor, direct-mapped, 32KB cache, 4-word block

Oficiální odpověď: – 2000个double × 8字节 = 16000字节 < 32KB (可完全放入cache) – 每块4字 × 8字节 = 32字节 = 4个double – Cache miss = 2000 / 4 = 500次miss

Doplňující poznámky (Cache Calculation Formula):

$$\begin{aligned}
 \text{Miss} &= \frac{\text{Total Size}}{\text{Block Size}} \\
 &= 2000 / (32 / 8) \\
 &= 2000 / (32/8) = 500
 \end{aligned}$$

Shrnutí

Rychlé odpovědi (高频考点!)

题号	类型	答案
1	RS仿真	X=00110, Y=10001
3	等价函数	$y1 \equiv y4$
4	10位运算	unsigned:1020, signed:-4
7	Gray码VHDL	$y \leq ('0' \& x(3 \text{ downto } 1)) \text{ xor } x$
8A	分支预测	1位:12, 2位:6
8B	Cache	500次miss

Nutné vzorce

10位有符号/无符号运算

- 1023在10位unsigned = 1023
- 1023在10位signed = -1 (因为是全1)
- $4 \times 1023 \bmod 1024 = 1020$ (unsigned)
- $4 \times (-1) = -4$ (signed)

分支预测miss计算 For循环编译为do-while:

```
// for (i=0; i<N; i++) body;
//
i = 0;
do {
    body;
    i++;
} while (i < N); //
```

1位预测器 (初始NT): - 循环开始: 预测NT, 实际T → **miss**, 变为T - 循环中: 预测T, 实际T → hit - 循环结束: 预测T, 实际NT → **miss** - 每个循环2次miss

2位预测器 (初始WT): - 循环开始: 预测T, 实际T → hit - 循环结束: 预测ST, 实际NT → **miss** - 每个循环1次miss

Cache Miss计算

$$\begin{aligned} \text{Cache Miss} &= \frac{\text{总数据量}}{\text{块大小}} \\ &= \left(\frac{\text{数据量}}{\text{块大小}} \right) / \left(\frac{\text{数据量}}{\text{块大小}} \right) \\ &= \frac{\text{数据量}}{\text{块大小}} \end{aligned}$$

对于64位处理器: - 字大小 = 8字节 - double = 8字节 - 每块4字 = 32字节 = 4个double - 2000个double
→ $2000/4 = 500$ 次miss