LSP Exam 2023-06-23

Course: B0B35LSP – Logické systémy a procesory | BE5B35LSP – Logic Systems and Processors **University**: CVUT FEL (ČVUT) – České vysoké učení technické v Praze | Czech Technical University in Prague **Keywords**: LSP, Exam, Zkouška, 2023-06-23, Shannon expansion, branch predictor, cache, VHDL

CN Version | EN Version | CZ Version

LSP Exam 2023-06-23

AI-derived version – No official answers in the PDF; the following is an inferred walkthrough/notes.

Exam Info

- Date: 2023-06-23
- Language: Czech
- Contains statistical graphs

Q1 – RS Latch Simulation (4 pts)

Task: Given inputs A, B, C values at times t0–t4, write the values of outputs X and Y.

Input sequence:

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

AI-derived answer: – **t0**: A=1 → Reset, X=0, Y=1 – **t1**: A=1, B·C=0 → Reset holds, X=0, Y=1 – **t2**: A=0, B·C=0 → hold – **t3**: A=0, B·C=1 → Set, X=1, Y=0 – **t4**: A=0, B·C=0 → hold – Hypothesis: **X=00011, Y=11100** (verify with the actual circuit)

Q2 – Shannon Expansion (6 pts)

Task: Decompose $X = f(A, B, C, X)$ into:

$$X = (\overline{X} \wedge f_0(A, B, C)) \vee (X \wedge f_1(A, B, C))$$

Q3 – Equivalent Logic Functions (4 pts)

Task: Check all logic functions that have an equivalent function:

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

Q4 – VHDL Function (Mean of AS7) (6 pts)

Task: Write a VHDL function to compute the rounded integer mean of an AS7 array. Aim for the fastest hardware calculation.

```
type AS7 is array(0 to 6) of integer range -2**15 to 2**15-1;

function meanAS7(x: AS7) return integer is
  variable sum : integer;
begin
  sum := x(0) + x(1) + x(2) + x(3) + x(4) + x(5) + x(6);
  return (sum + 3) / 7;  -- +3 for rounding
end function;
```

Optimization tips: – Parallel additions (adder tree) – Avoid division (use multiply/shift approximations) – 7 ≈ $2^3 - 1$ can be exploited

---

Q5 – Multiplexer Circuit Implementation (8 pts)

Task: Implement the given multiplexer circuit using only AND, NAND, OR, NOR gates and inverters.

Diagram (symbolic):

```
x ── 0 ─┬─ 0 ─┐
y ── 1 ─┤     ├─ q
z ──────┴─ 1 ─┘
a0
a1
```

Equations:

```
q = (not a1 and not a0 and x) or
    (not a1 and a0 and y) or
    (a1 and z)
```

Simplified:

```
q = (not a1 and ((not a0 and x) or (a0 and y))) or (a1 and z)
```

Q6 – Multiplexer VHDL Description (6 pts)

Task: Describe the circuit from Q5 in the simplest VHDL.

```
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity yyy is
  port(x, y, z, a1, a0: in std_logic;
       q: out std_logic);
end entity;
architecture dataflow of yyy is
begin
  q <= z when a1='1' else
       y when a0='1' else
       x;
end architecture;
```

Q7 – Branch Predictor (6 pts)

Not on Exam note: According to the 2026 exam notes, branch-predictor calculation tasks are not tested; you can skip strategically.

Task: A C program finds the minimum in an array. Assume the `for` loop is compiled to a `do-while`. Compute misprediction counts.

```
int data[] = { 0, 1, 2, 3, 4, -5, -6, -7, 8, 9 };
int min = INT_MAX;
for (int i = 0; i < sizeof(data)/sizeof(int); i++)
{
  if (data[i] < min) min = data[i];
}
```

Answer placeholders: – 1-bit predictor (initial Not-Taken): misses = ? – 2-bit predictor (initial Weakly Taken): misses = ?

Q8 – Cache Computation (10 pts)

Not on Exam note: According to the 2026 exam notes, cache-miss calculation tasks are not tested; you can skip strategically.

Task: 32-bit processor, cache is only 128 bytes, 2-way set associative, line size 1 word, LRU replacement. Access sequence: `0x14, 0x94, 0x14, 0x94, 0x114, 0x14`

Cache parameters: – Word size = 4 bytes – Line size 1 word = 4 bytes – 128 bytes / 4 bytes = 32 lines – 2-way set associative: 32 / 2 = 16 sets – Address split: – Offset: 2 bits (byte offset) – Set index: 4 bits (16 sets) – Tag: remaining bits

---

Q9 – Bonus: 3-bit Johnson Counter (10 pts)

Task: Implement a 3-bit synchronous Johnson counter. – When RESET='1'→ Q="000"– When DN=0 count forward: 000,001,011,111,110,100,000…– When DN=1 count backward

```vhdl
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity JC3 is
  port (clk, DN, RESET: in std_logic;
        Q: out std_logic_vector(2 downto 0));
end entity;
architecture rtl of JC3 is
  signal q_int : std_logic_vector(2 downto 0) := "000";
begin
  process(clk)
  begin
    if rising_edge(clk) then
      if RESET = '1' then
        q_int <= "000";
      elsif DN = '0' then
        q_int <= q_int(1 downto 0) & (not q_int(2));
      else
        q_int <= (not q_int(0)) & q_int(2 downto 1);
      end if;
    end if;
  end process;
  Q <= q_int;
end architecture;
```

Key Topics Summary

Focus points in this exam

1. RS latch simulation
2. Shannon expansion
3. Equivalent logic functions
4. **VHDL function writing** (fast mean)
5. Multiplexer circuit implementation
6. Multiplexer VHDL description
7. **Branch predictor calculation**
8. **Cache computation** (2-way set associative)

9. Johnson counter design