LSP Exam 2022-01-19

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, 2022-01-19, truth table, De Morgan, RS latch, VHDL

CN Version | EN Version | CZ Version

LSP Exam 2022-01-19

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

Exam Info

- Date: 2022-01-19
- Language: Czech
- Total: 50 points (Part 1: 25 pts ⩾ 9 + Part 2: 25 pts ⩾ 9)

Q1 – Truth Table (7 pts)

Task: Write the truth table for function Y based on the circuit diagram.

Hint: You can use Shannon expansion to simplify the computation.

K-map layout:

```
        B
      A  00 01 11 10
    ┌─────────────────
DC 00│
   01│
   11│
   10│
```

Q2 – De Morgan's Laws (3 pts)

Task: Rewrite logic expression F so that the `not` operator appears only directly in front of variables.

Original:

```
F = not ( (not A and not C) xor (A and B and C) )
```

Approach: 1. Expand XOR: $A \oplus B = (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$ 2. Apply De Morgan: $\overline{A \vee B} = \overline{A} \wedge \overline{B}$

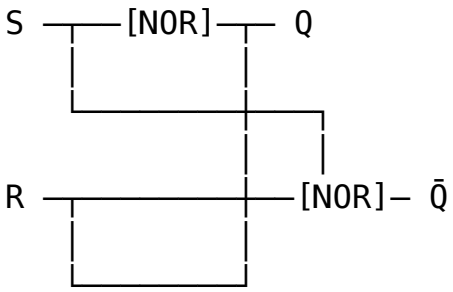
Answer (one possible form):

```
F = ((A or C) and (not A or not B or not C)) xor
    ((not A and not C) or (A and B and C))
-- simplified:
F = (A and not B) or (not A and not C) or (B and C)
```

Q3 – RS Latch Design (10 pts)

Task: Draw an RS latch using NOR gates and NAND gates, and complete the truth tables.

RS latch with NOR gates



Truth table (NOR): | S | R | Q | Q⁻¹ |—|—|—|—| | 0 | 0 | hold Q | hold Q⁻¹ | 0 | 1 | 0 | 1 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 0 (invalid) |

RS latch with NAND gates

Truth table (NAND): active–low inputs | S | R | Q | Q⁻¹ |—|—|—|—| | 0 | 0 | 1 | 1 (invalid) | | 0 | 1 | 1 | 0 | | 1 | 0 | 0 | 1 | | 1 | 1 | hold Q | hold Q⁻¹

Q4 – Moore/Mealy Automaton Definition (5 pts)

Task: Complete the Moore/Mealy automaton definition.

$$M =< X, S, Z, \omega, \delta, s_0 \in S >$$

Answer: – **X**: Finite input alphabet – **S**: Finite set of states – **Z**: Finite output alphabet – **δ**: State transition function (Moore: $S \times X \to S$, Mealy: $S \times X \to S$) – **ω**: Output function (Moore: $S \to Z$, Mealy: $S \times X \to Z$) – **s_0**: Initial state

---

Q5 – VHDL Code Analysis (10 pts)

Task: Analyze the compressed VHDL code and draw the circuit diagram.

```vhdl
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity zzz is port (a : in std_logic; data : in std_logic; q, qn : out std_
architecture rtl of zzz is
  signal qv : std_logic;
begin
  process(a)
  begin
    if rising_edge(a) then
      qv <= not data;
      q <= qv;
    end if;
    qn <= not qv;
  end process;
end;
```

Functional analysis: – Triggered on rising edge of `a` – `qv` stores the inverted `data` – `q` outputs the previous `qv` (one-cycle delay) – `qn` is the inversion of `qv` – This behaves like a 1-bit shift register / pipeline with inversion

---

Q6 – Full Adder Design (3 pts)

Task: Draw a complete one-bit full adder with inputs A, B, Cin and outputs Y (Sum), Cout.

Equations:

$$\text{Sum} = A \oplus B \oplus Cin$$
$$\text{Cout} = (A \wedge B) \vee (Cin \wedge (A \oplus B))$$

Q7 – VHDL Circuit Description (7 pts)

Task: Describe the given circuit diagram in the shortest VHDL code using vector operations.

```vhdl
library ieee; use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Zahadum is port(
  -- declare ports according to the schematic
);
end entity;
architecture rtl of Zahadum is
begin
```

```vhdl
  -- use vector operations to simplify
end architecture rtl;
```

Q8 – Concurrent VHDL Description (5 pts)

Task: Analyze the function of the circuit from Q1 and describe it using simplified concurrent statements (not a direct rewrite of the boolean equations).

```vhdl
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity yyy is port(a,b,c,d: in std_logic; y: out std_logic); end entity;
architecture dataflow of yyy is
begin
  -- simplified concurrent statements
end architecture;
```

Q9 – Bonus: SOS Beacon Encoder (10 pts)

Task: Write an SOS signal encoder using advanced VHDL constructs. Signal sequence: `01010100011101110111000101010 0`

```vhdl
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity SOS is
  port (clk : in std_logic;
        X: in std_logic_vector(4 downto 0);
        Morse, STOP: out std_logic);
end entity;
architecture rtl of SOS is
  constant SOS_PATTERN : std_logic_vector(29 downto 0) :=
    "01010100011101110111000101010 0";
begin
  process(X)
    variable idx : integer;
  begin
    idx := to_integer(unsigned(X));
    if idx < 30 then
      Morse <= SOS_PATTERN(29 - idx);
      STOP <= '0';
    else
      Morse <= '0';
      STOP <= '1';
    end if;
  end process;
end architecture;
```

Key Topics Summary

Focus points in this exam

1. Building truth tables
2. Applying De Morgan's laws
3. **RS latch (NOR and NAND implementations)**
4. Moore/Mealy automaton definition
5. VHDL code analysis (shift–register behavior)
6. Full adder design
7. Vector operations in VHDL
8. Circuit simplification / functional reasoning
9. Morse/SOS encoder