Zkouška LSP 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

Zkouška LSP 2023-06-23

AI odvozená verze —V PDF nejsou oficiální odpovědi; níže jsou odvozené poznámky/řešení.

Informace o zkoušce

- Datum: 2023-06-23
- Jazyk: čeština
- Obsahuje statistické grafy

Úloha 1 —Simulace RS záchytu (4 b)

Zadání: Pro dané hodnoty vstupů A, B, C v časech t0—t4 určete výstupy X a Y.

Vstupní posloupnost:

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

AI odvozená odpověď: – **t0**: A=1 → Reset, X=0, Y=1 – **t1**: A=1, B·C=0 → Reset drží, X=0, Y=1 – **t2**: A=0, B·C=0 → držení – **t3**: A=0, B·C=1 → Set, X=1, Y=0 – **t4**: A=0, B·C=0 → držení – Hypotéza: **X=00011, Y=11100** (ověřte dle skutečného schématu)

Úloha 2 —Shannonův rozklad (6 b)

Zadání: Rozložte $X = f(A, B, C, X)$ do tvaru:

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

Úloha 3 —Ekvivalentní logické funkce (4 b)

Zadání: Zaškrtněte všechny logické funkce, které jsou ekvivalentní:

```
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);
```

Úloha 4 —VHDL funkce (průměr AS7) (6 b)

Zadání: Napište VHDL funkci pro výpočet zaokrouhleného celočíselného průměru pole typu AS7. Požadavek: co nejrychlejší hardwarová realizace.

```
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 pro zaokrouhlení
end function;
```

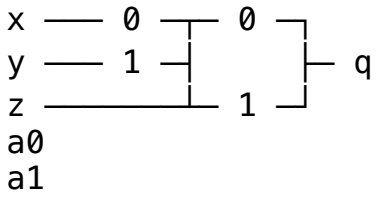
Tipy na optimalizaci: – paralelní scítání (adder tree) – vyhnout se dělení (nahradit násobením/posuvem) – 7 ≈ $2^3 - 1$ lze využít

---

Úloha 5 —Realizace multiplexoru (8 b)

Zadání: Realizujte daný multiplexor pouze pomocí hradel AND, NAND, OR, NOR a invertorů.

Symbolicky:



Rovnice:

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

Zjednodušení:

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

Úloha 6 — VHDL popis multiplexoru (6 b)

Zadání: Popište obvod z Úlohy 5 co nejjednoduběji ve VHDL.

```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;
```

Úloha 7 — Prediktor skoků (6 b)

> **Poznámka (není zkouběno):** Podle poznámek k roku 2026 se výpočty prediktoru skoků netestují; lze přeskočit.

Zadání: Program v C hledá minimum v poli. Předpokládejte, že `for` je přeložen jako `do-while`. Spočítejte počet chybných predikcí.

```c
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];
}
```

Místa pro odpověď: – 1bitový prediktor (počátečně Not–Taken): misses = ? – 2bitový predik–tor (počátečně Weakly Taken): misses = ?

Úloha 8 — Výpočet cache (10 b)

> **Poznámka (není zkouběno):** Podle poznámek k roku 2026 se výpočty cache missů netestují; lze přeskočit.

Zadání: 32bit procesor, cache 128 B, 2–way set associative, délka řádku 1 slovo, LRU. Sekvence přístupů: `0x14, 0x94, 0x14, 0x94, 0x114, 0x14`

Parametry cache: – Slovo = 4 bajty – Řádek 1 slovo = 4 bajty – 128 B / 4 B = 32 řádků – 2–way: 32 / 2 = 16 sad – Rozklad adresy: – Offset: 2 bity – Index sady: 4 bity – Tag: zbytek

---

Úloha 9 — Bonus: 3bitový Johnsonův čítač (10 b)

Zadání: Implementujte 3bitový synchronní Johnsonův čítač. – RESET='1' → Q="000" – DN=0 doprředu: 000,001,011,111,110,100,000… – DN=1 dozadu

```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;
```

Shrnutí témat

Hlavní okruhy

1. Simulace RS záchytu
2. Shannonův rozklad
3. Ekvivalentní logické funkce
4. **VHDL funkce** (rychlý průměr)
5. Realizace multiplexoru
6. VHDL popis multiplexoru
7. **Prediktor skoků**
8. **Výpočet cache** (2–way)
9. Johnsonův čítač