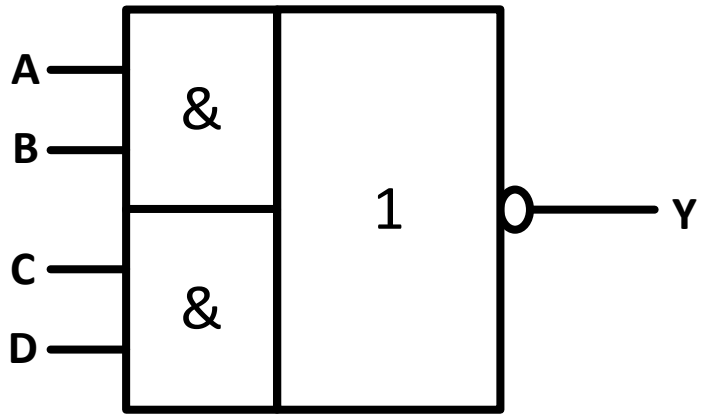
TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Používané logické funkce a jejich popis ve VHDL

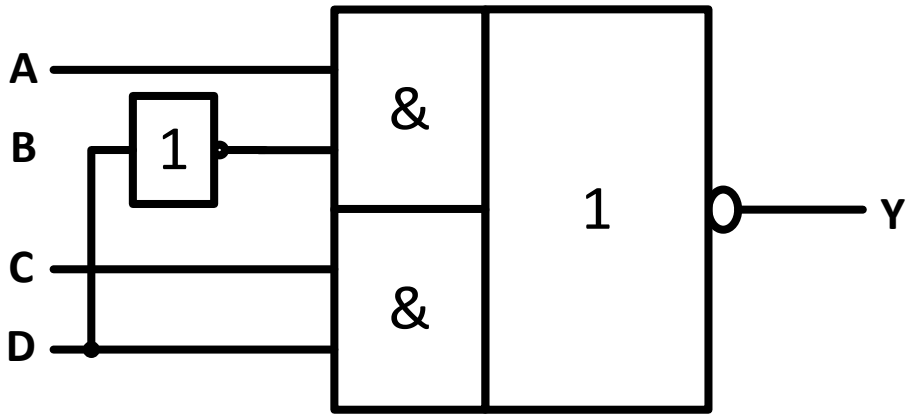
Prof. Ing. Ondřej Novák, CSc.
ITE

Složitější obvody

- **And-or-invert**



Schématická značka

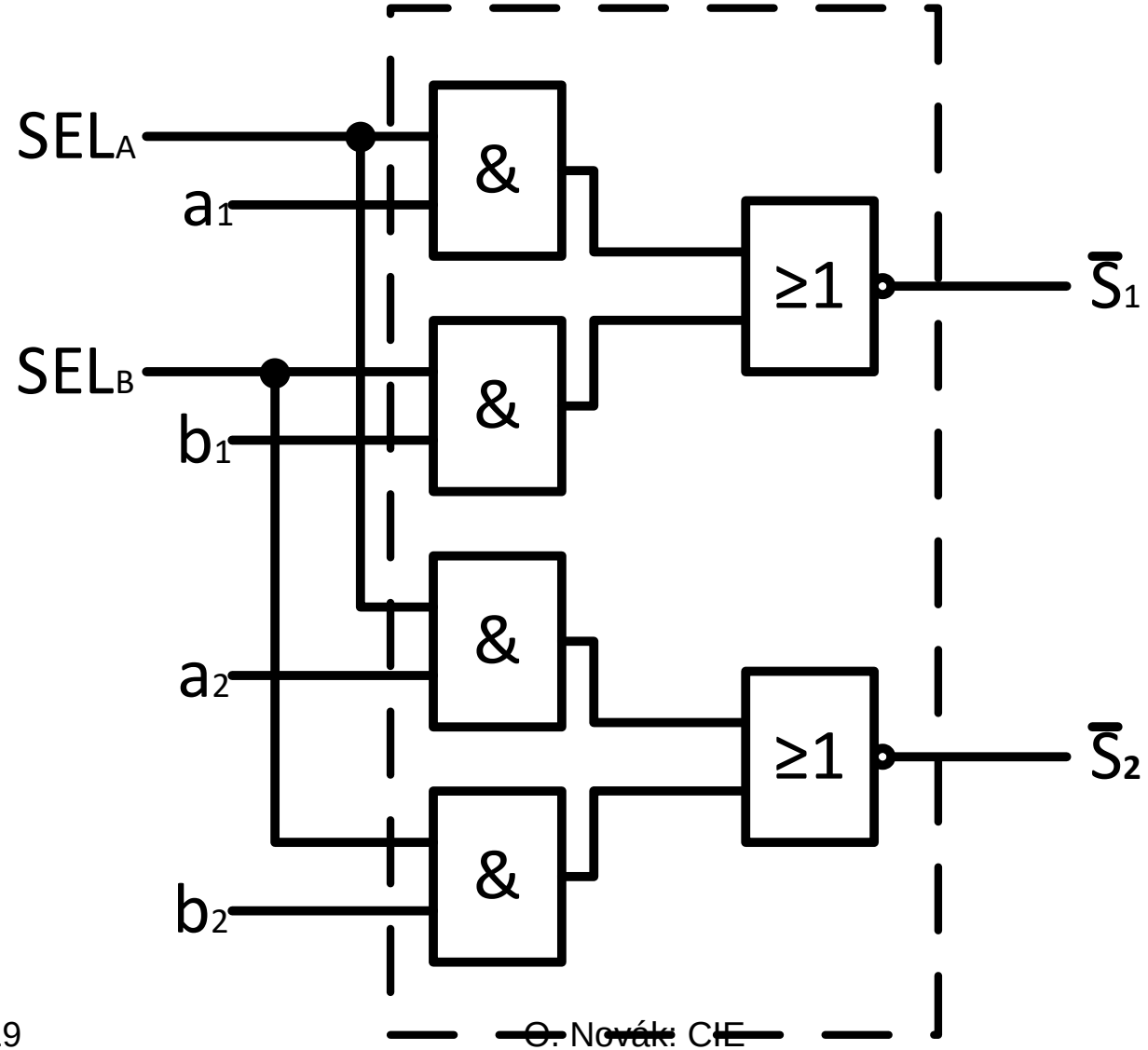


Jednobitová výhybka

Jednobitová výhybka – popis VHDL

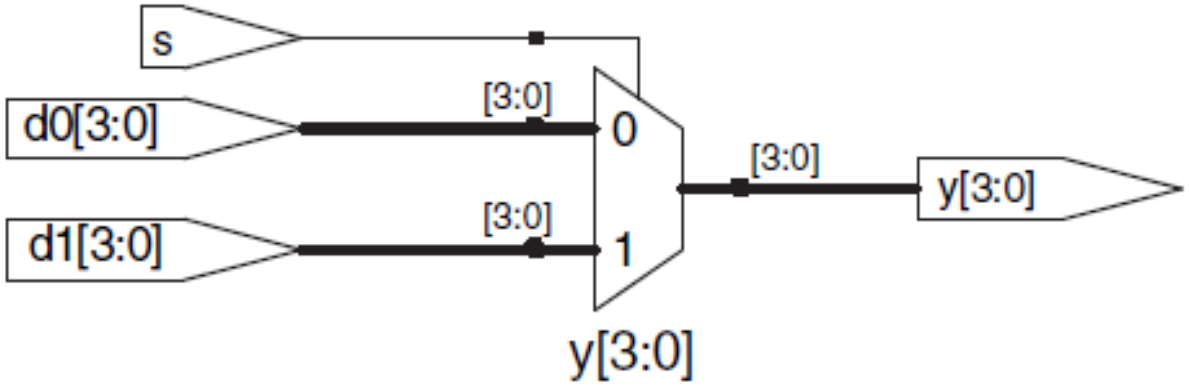
```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity vyhybka is
port(       A: in STD_LOGIC;
            C: in STD_LOGIC;
            D:in STD_LOGIC;
            Y: out STD_LOGIC);
end;

architecture synth1 of vyhybka is
begin
Y <= not A when D = '0' else not C;
end;
```

Vícebitová výhybka

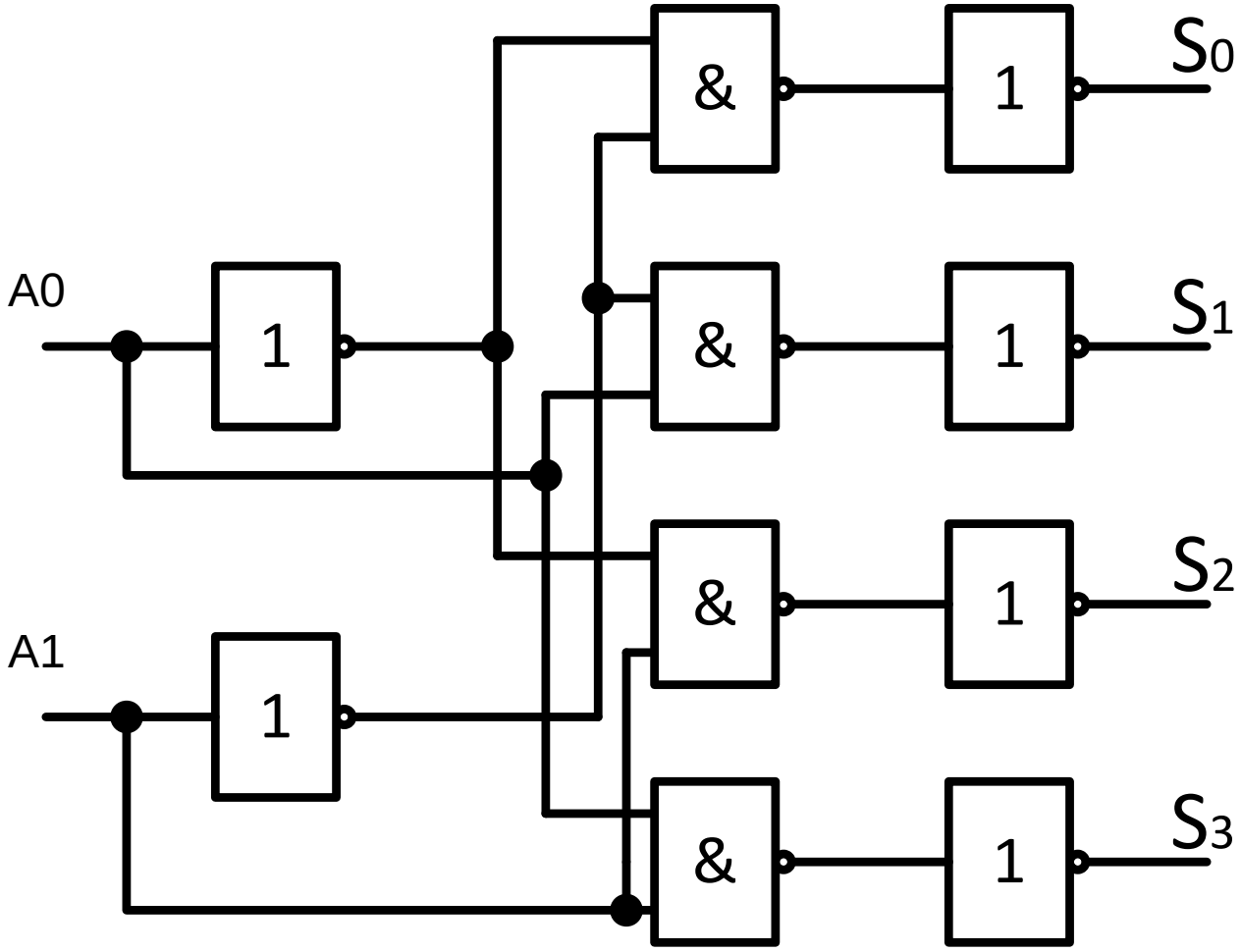


podmíněné přiřazení signálů – vícebitová výhybka

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity mux2 is
port(d0, d1: in STD_LOGIC_VECTOR(3 downto 0);
s: in STD_LOGIC;
y: out STD_LOGIC_VECTOR(3 downto 0));
end;

architecture synth of mux2 is
begin
y <= d1 when s='1' else d0;
end;
```



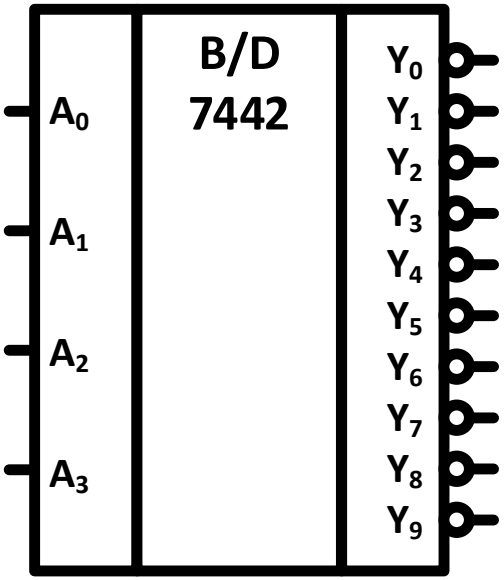
Principiální schéma dekodéru

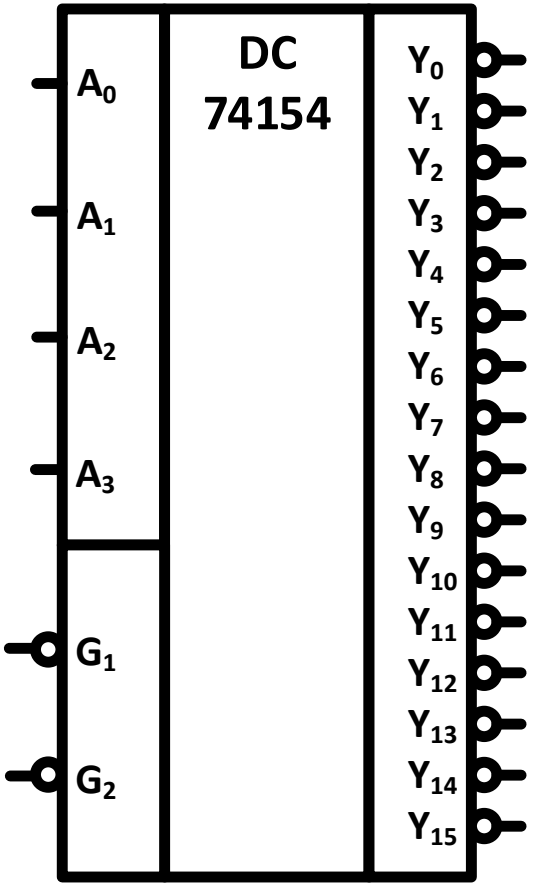


Dekodér

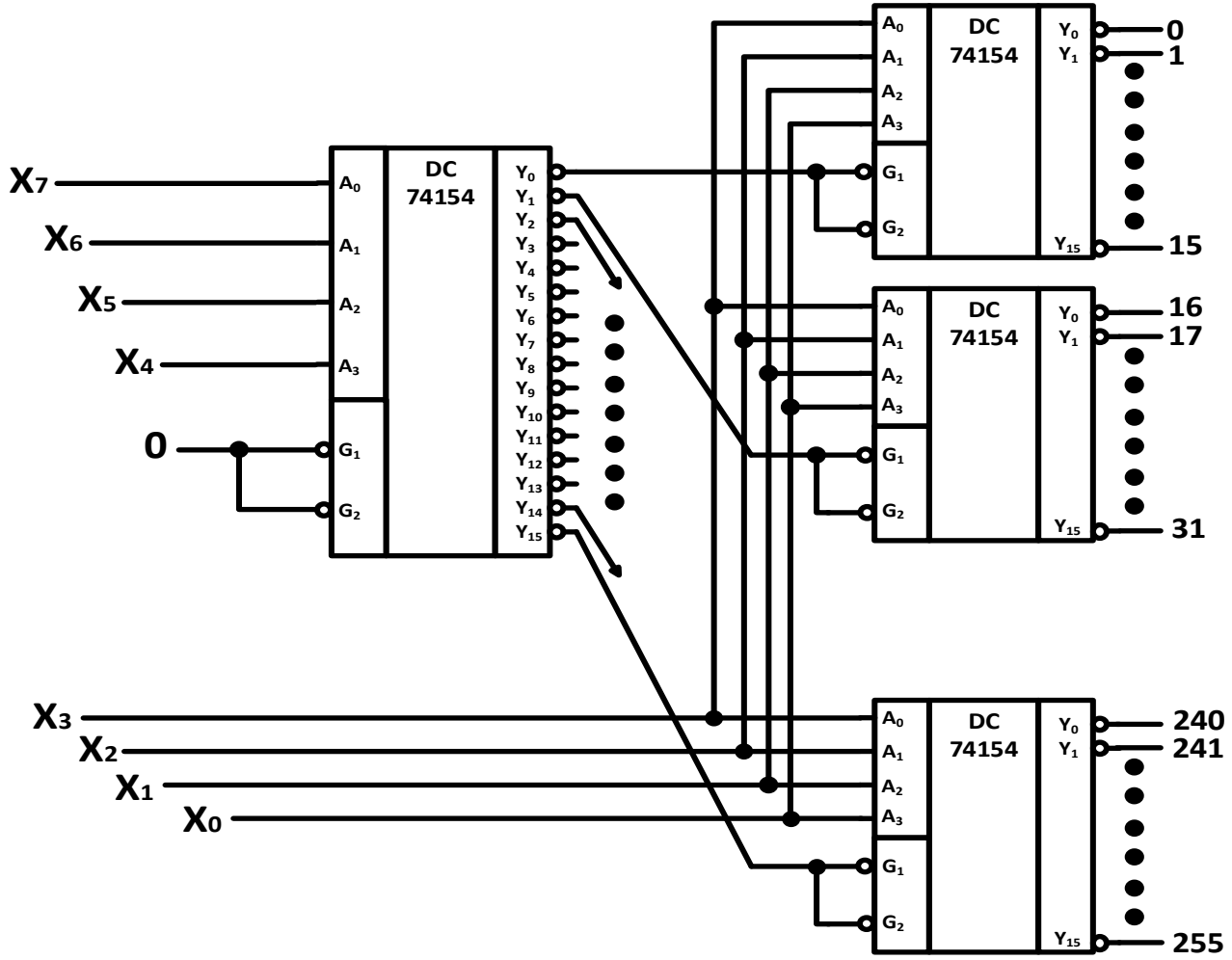
```
library ieee; use ieee.std_logic_1164.all;
entity dekoder is
port       ( A : in std_logic_vector(1 downto 0);
S : out std_logic_vector(3 downto 0) );
end;

architecture dataflow of dekoder is
begin
with A select
S <=     "1110" when "00",
         "1101" when "01",
         "1011" when "10",
         "0111" when others;
end;
```

B/D a B/H dekodéry





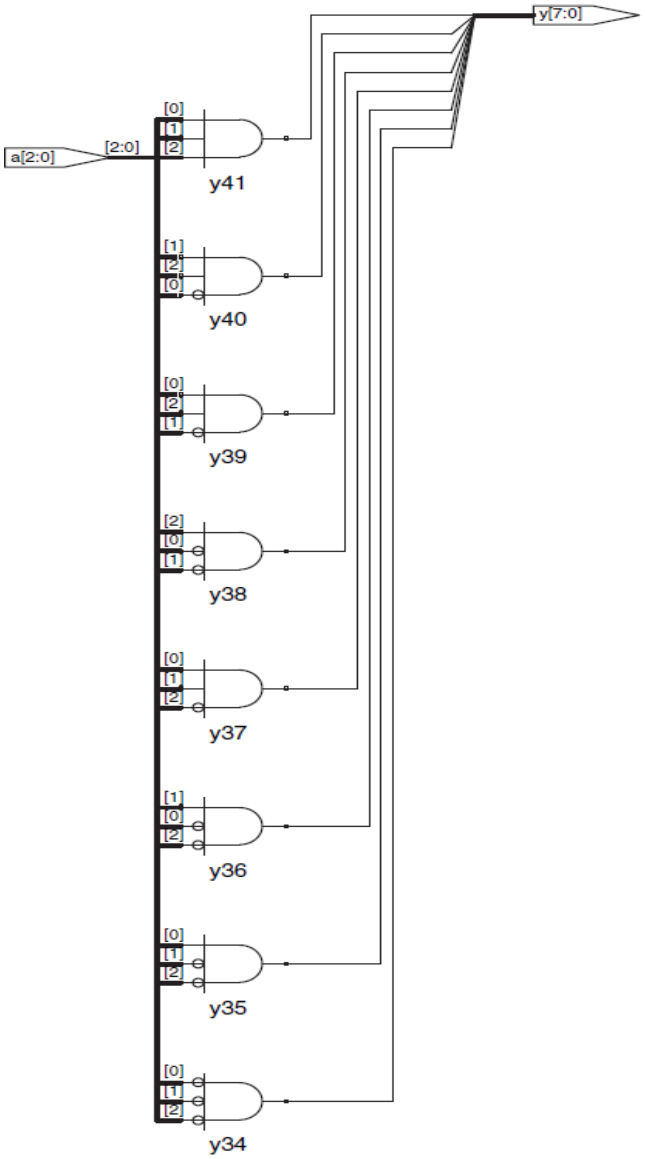
Rozšíření dekodéru



Rozšíření na kód 1 z 256 v negativní logice

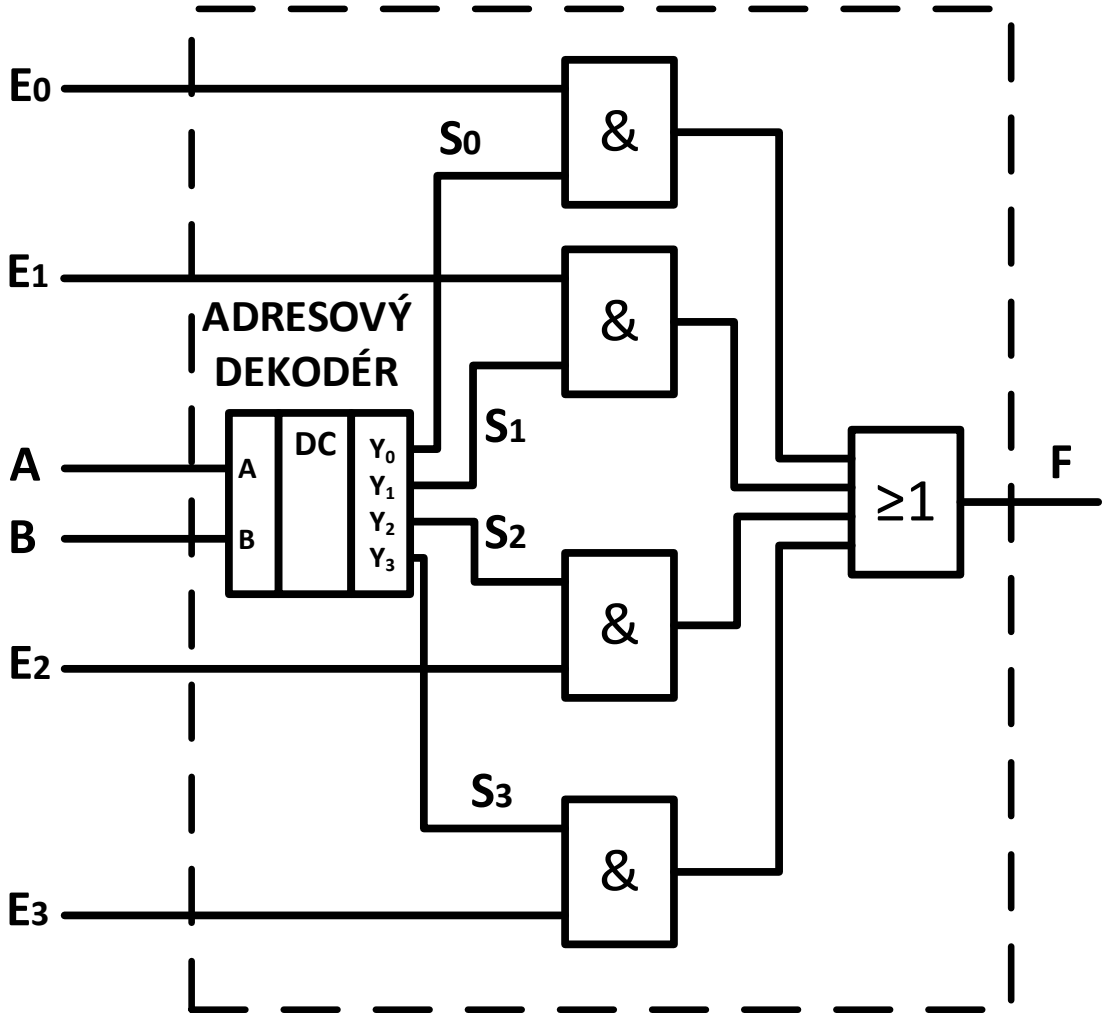
Dekodér VHDL pomocí procesu

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity decoder3_8 is
port(a: in STD_LOGIC_VECTOR(2 downto 0);
y: out STD_LOGIC_VECTOR(7 downto 0));
end;
architecture synth of decoder3_8 is
begin
process(a) begin
case a is
when "000" => y<= "00000001";
when "001" => y<= "00000010";
when "010" => y<= "00000100";
when "011" => y<= "00001000";
when "100" => y<= "00010000";
when "101" => y<= "00100000";
when "110" => y<= "01000000";
when "111" => y<= "10000000";
end case;
end process;
end;
```

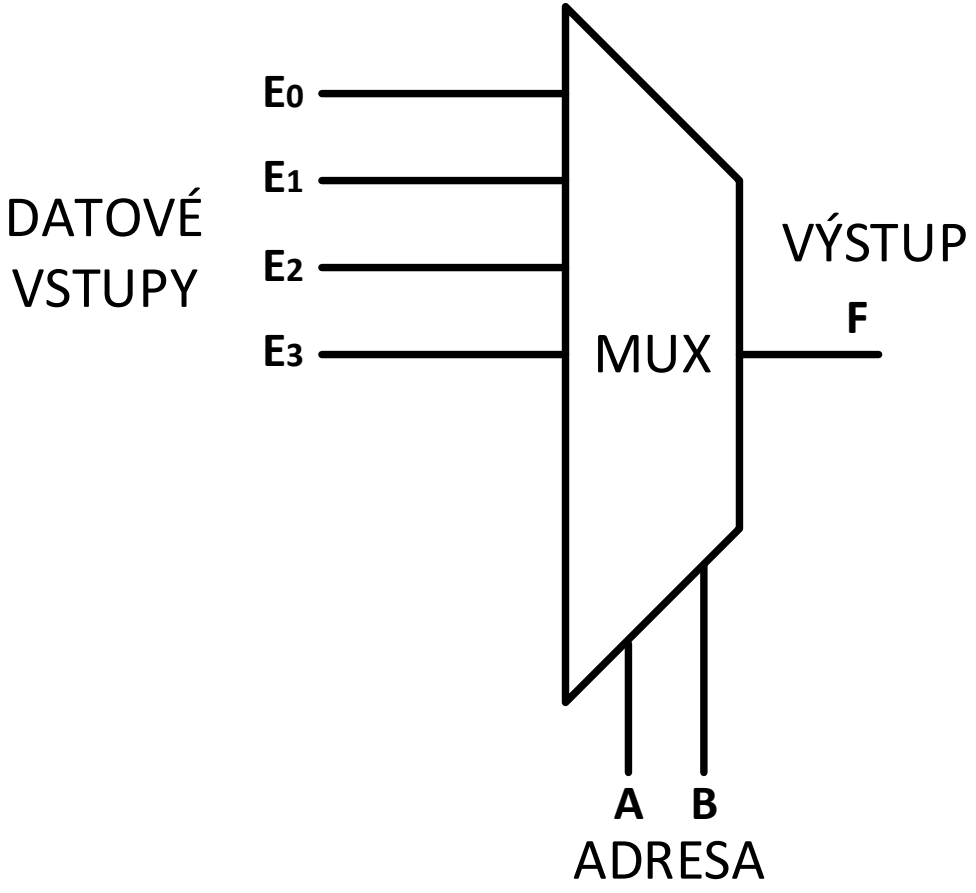


Multiplexor

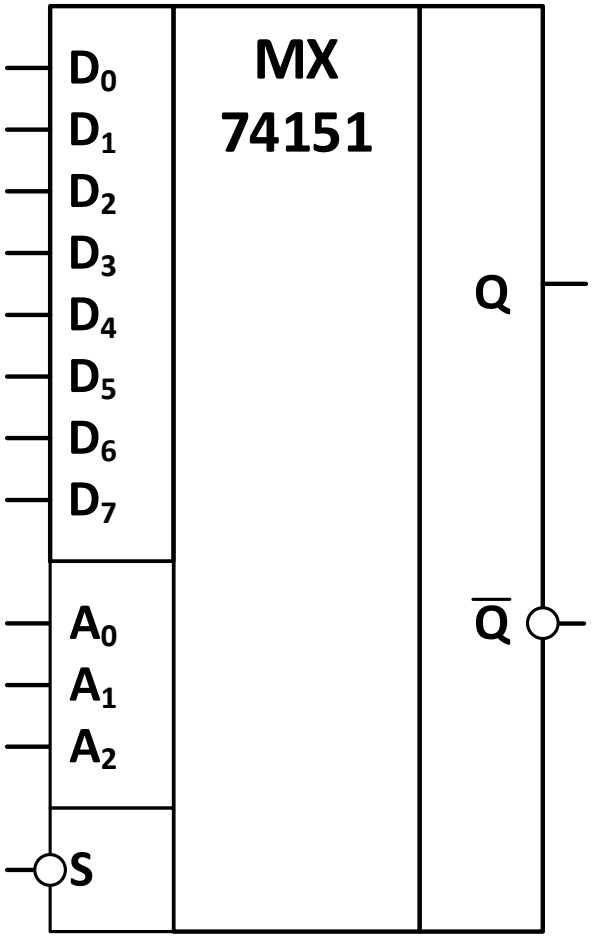
- **Na výstup (F) přivádí jeden ze vstupních signálů (E) v závislosti na adrese (A a B)**



Multiplexor



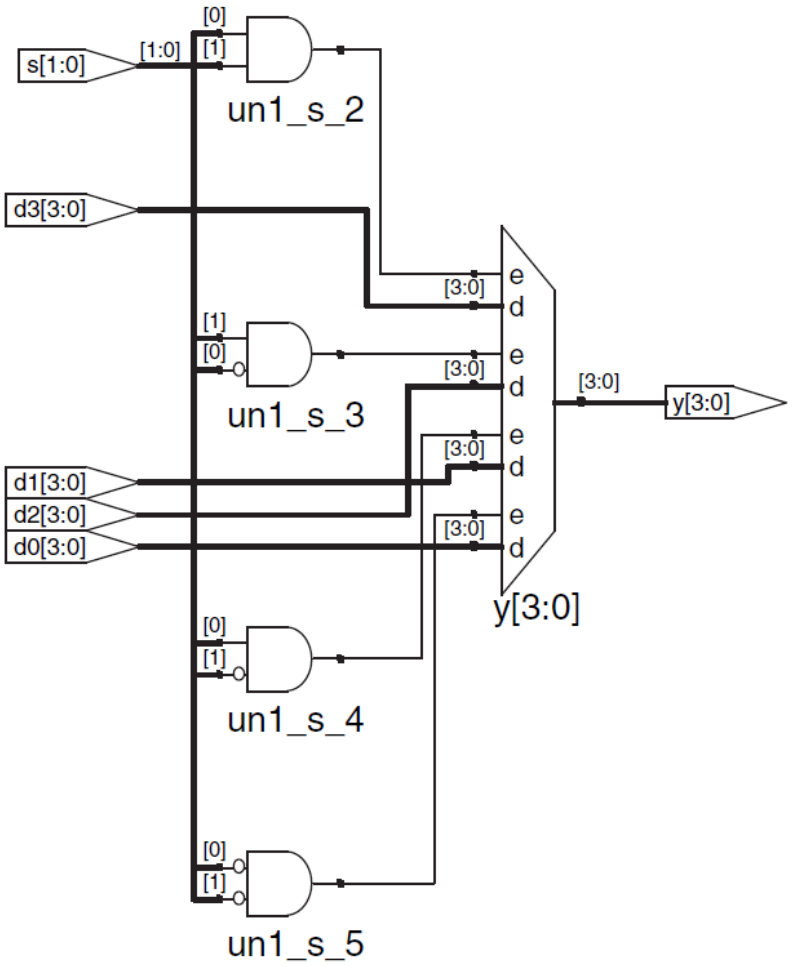
Obecné značení



Multiplexor MX74151

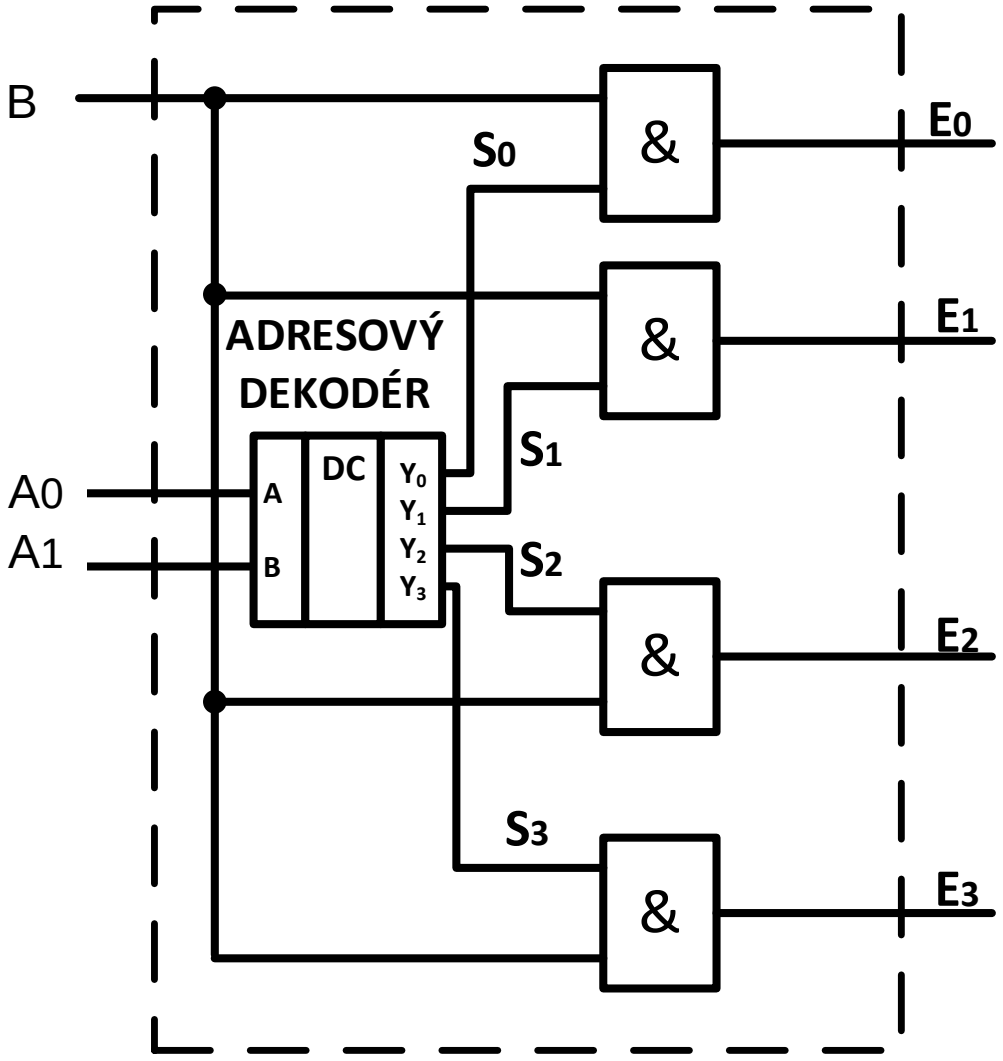
Multiplexor

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity mux4 is
port (d0, d1,d2, d3: in STD_LOGIC_VECTOR
(3 downto 0);
s: in STD_LOGIC_VECTOR (1 downto 0);
y: out STD_LOGIC_VECTOR (3 downto 0));
end;
architecture synthl of mux4 is
begin
y <= d0 when s = "00" else
d1 when s = "01" else
d2 when s = "10" else
d3;
end;
```



Demultiplexor

- **inverzní funkce k multiplexoru**



Demultiplexor ve VHDL

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity demux is
port(A: in STD_LOGIC_VECTOR(1 downto 0);
B :in STD_LOGIC ;
E: out STD_LOGIC_VECTOR(3 downto 0));
end;
```

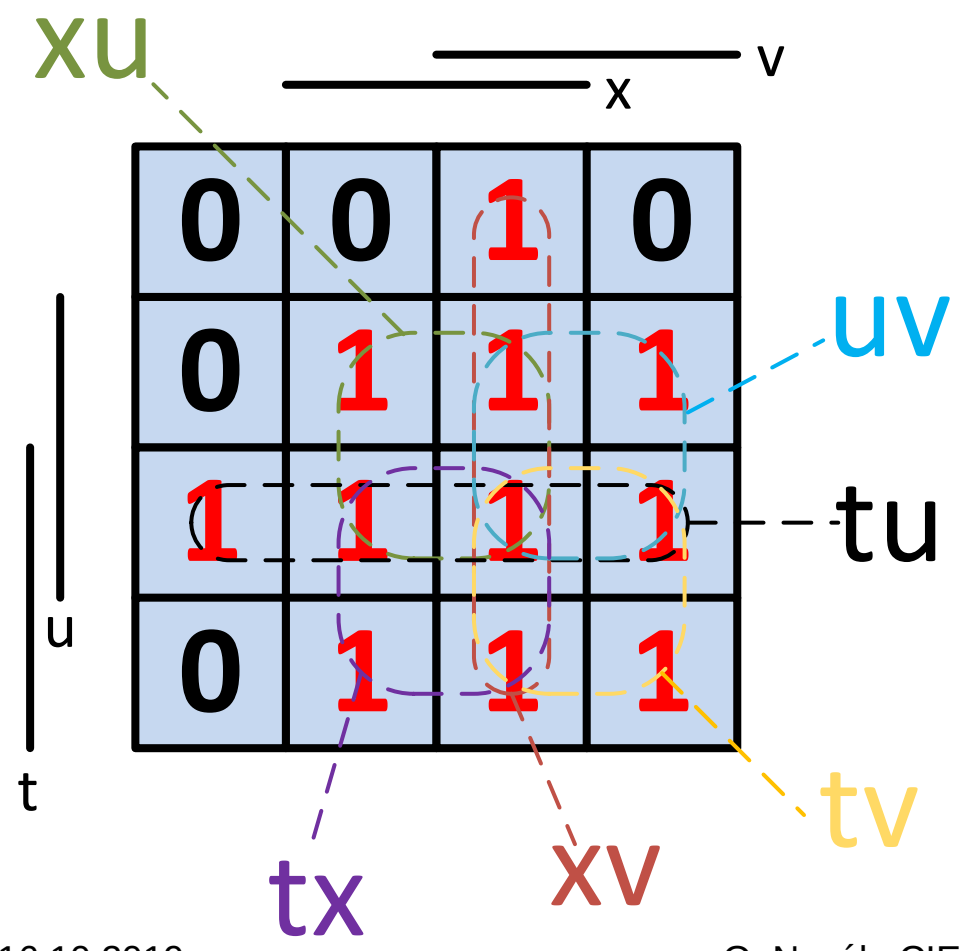
```
architecture synth of demux is
begin
process(A,B) begin
if  B='1' then
case A is
when "00" => E<= "0001";
when "01" => E<= "0010";
when "10" => E<= "0100";
when others => E <= "1000";
end case;
else
E <= "0000";
end if;
end process;
end;
```

Příklad

- **realizace logické funkce pomocí hradel NAND, dekodéru a multiplexoru**

- **<u>Zadání</u>: Zabezpečená místnost se 4 senzory: T a U - tlakové senzory, V optický senzor, X - infračervený senzor. Sestrojte obvod, který aktivuje alarm S (alarm aktivní pro log.1), jestliže alespoň dva ze senzorů T, U, V a X jsou aktivovány (na jejich výstupu je log. 1).**

Pravdivostní tabulka a minimalizace pomocí Karnaughovy mapy



index	t	u	v	x	s
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Úprava výrazu:

MNDF:

$$s = vx + ux + uv + tx + tv + tu$$

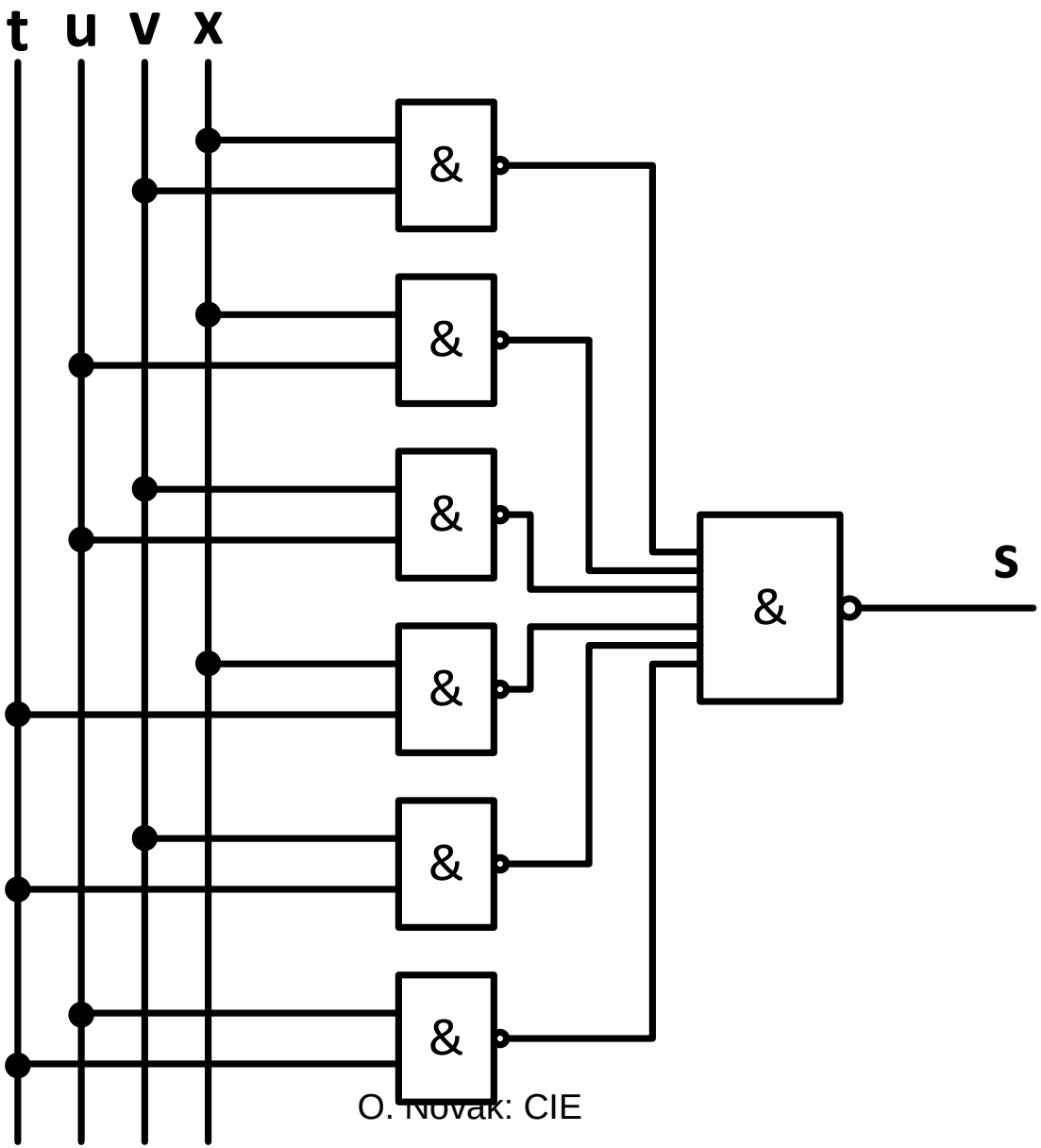
Úprava pomocí De-Morganova pravidla:

$$s = \overline{\overline{vx + ux + uv + tx + tv + tu}}$$

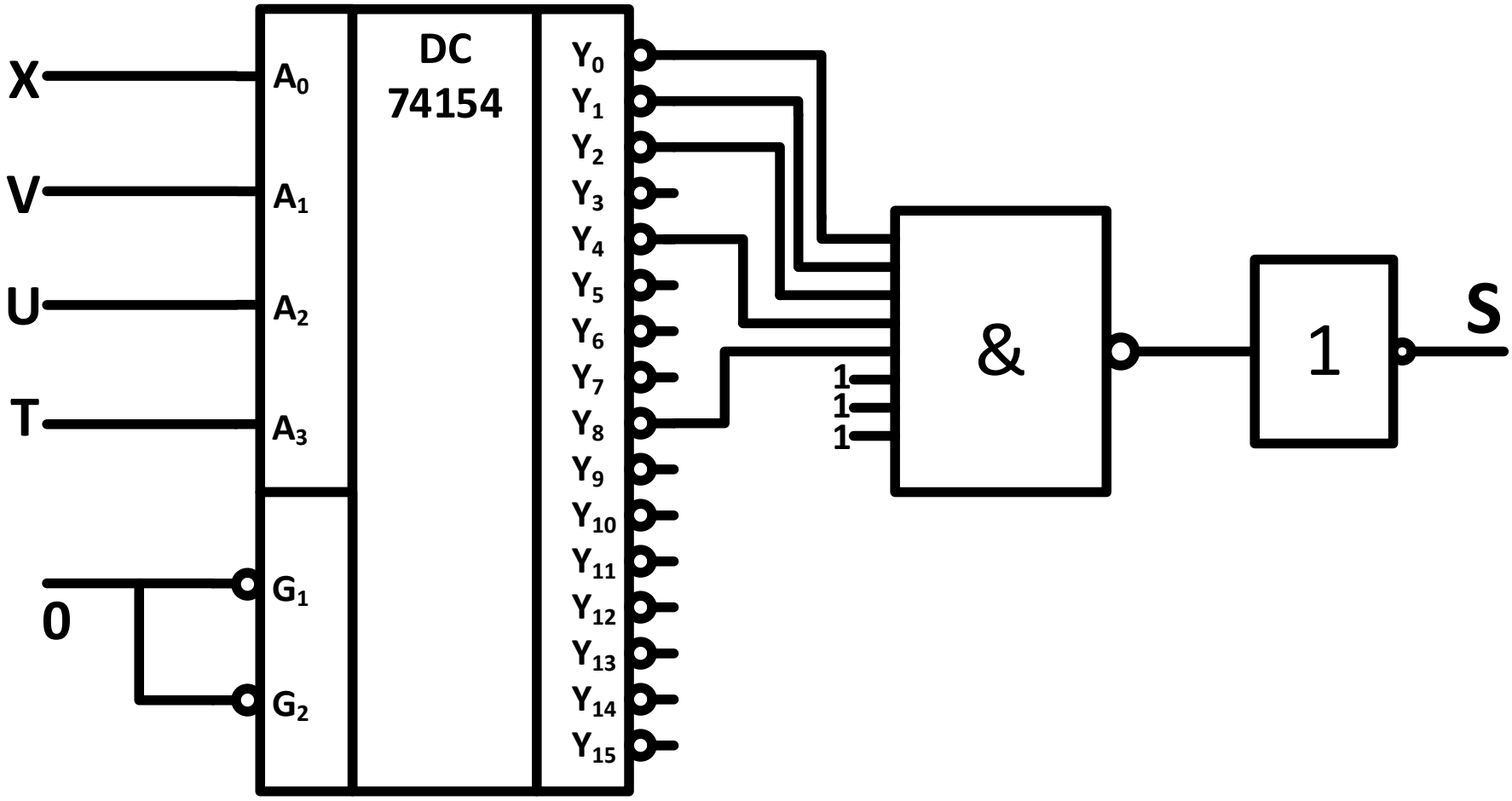
Výraz upravený pro realizaci hradly NAND:

$$s = \overline{\overline{vx} * \overline{ux} * \overline{uv} * \overline{tx} * \overline{tv} * \overline{tu}}$$

Realizace hradly NAND:

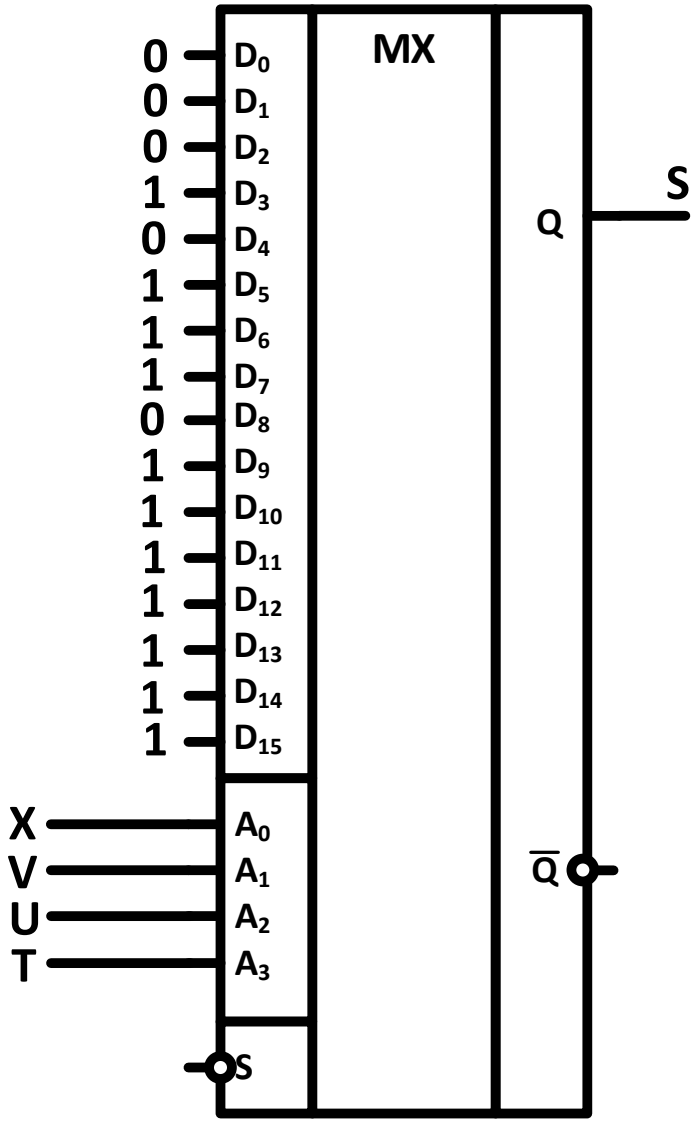


Realizace pomocí dekodéru:



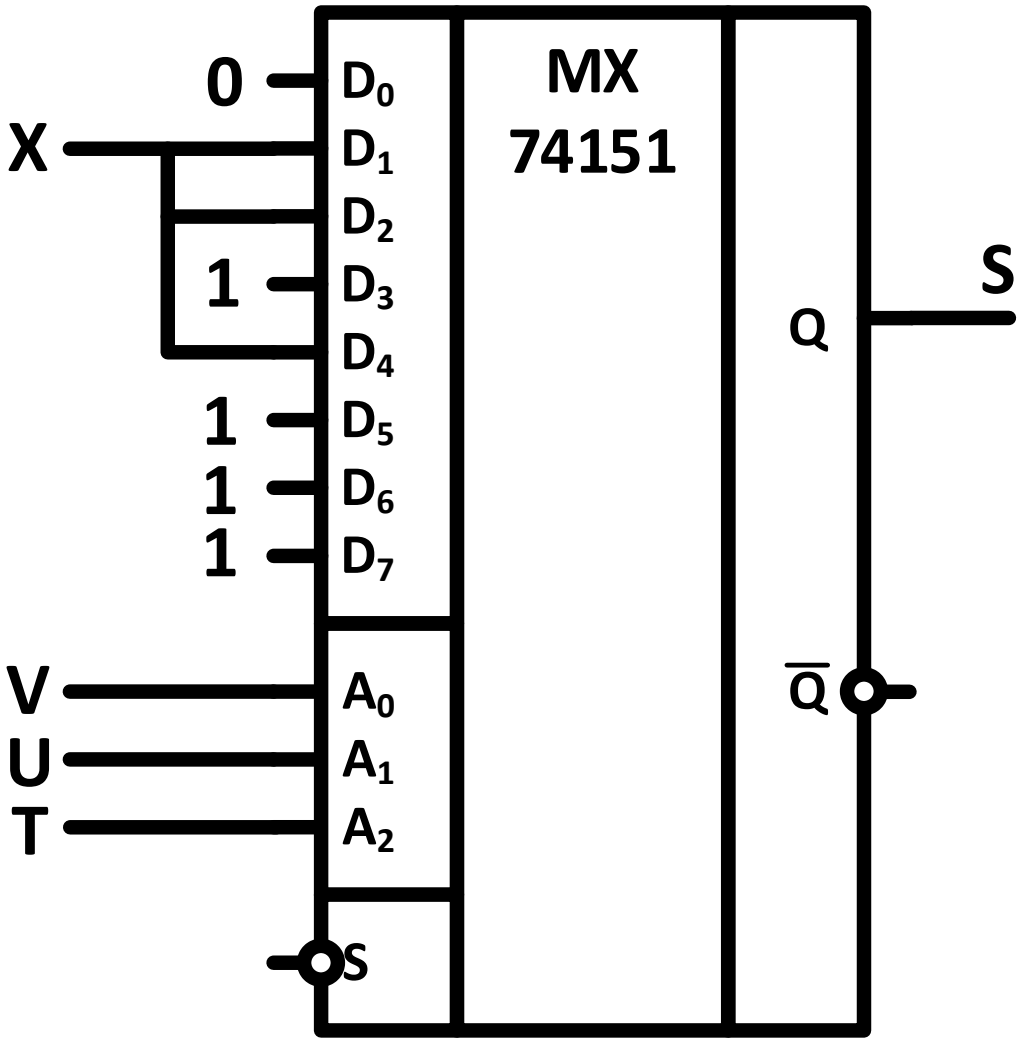
Realizace pomocí multiplexoru:

index	t	u	v	x	s
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

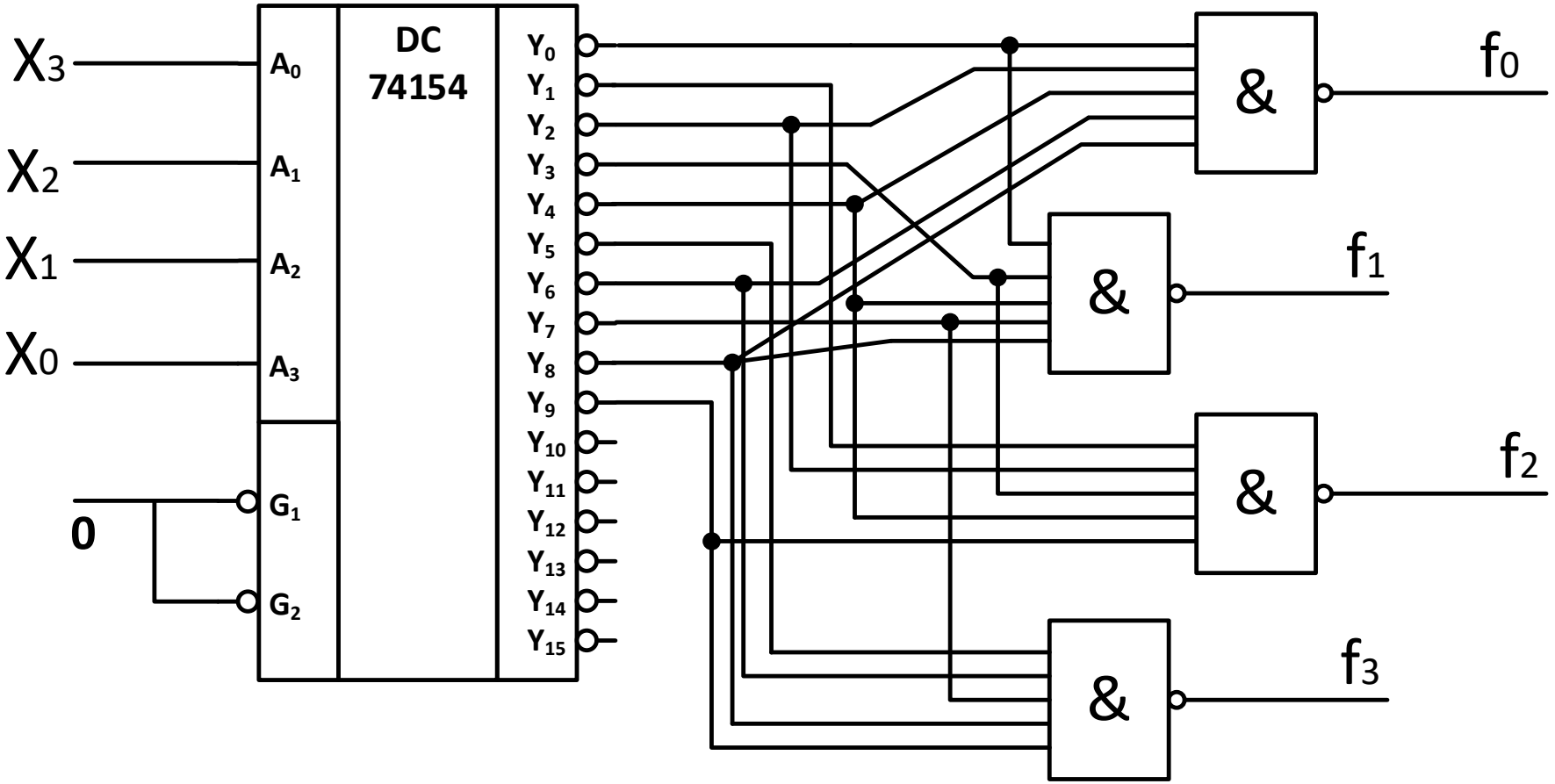


Realizace pomocí multiplexoru:

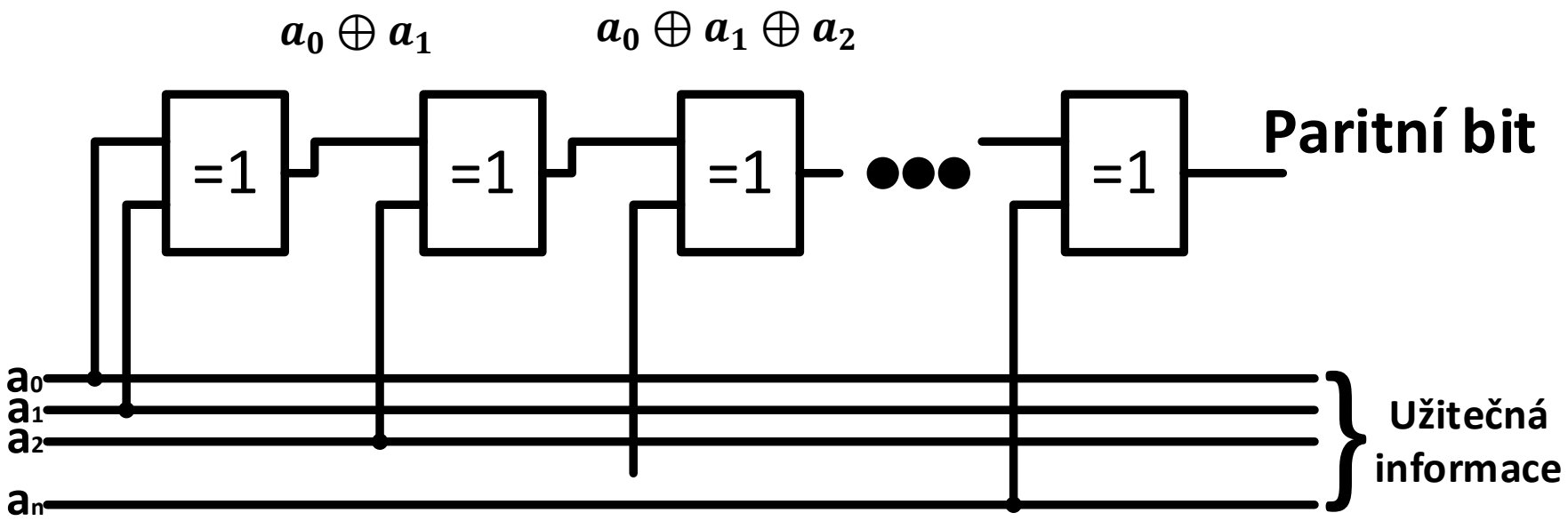
index	t	u	v	x	s	
0	0	0	0	0	0	0
1	0	0	0	1	0	
2	0	0	1	0	0	x
3	0	0	1	1	1	
4	0	1	0	0	0	x
5	0	1	0	1	1	
6	0	1	1	0	1	1
7	0	1	1	1	1	
8	1	0	0	0	0	x
9	1	0	0	1	1	
10	1	0	1	0	1	1
11	1	0	1	1	1	
12	1	1	0	0	1	1
13	1	1	0	1	1	
14	1	1	1	0	1	1
15	1	1	1	1	1	



Realizace skupiny log. funkcí jedním dekodérem



Kontrola / generování parity



Parita ve VHDL

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity parita is
port (a: in STD_LOGIC_VECTOR(8 downto 0);
p: out STD_LOGIC);
end;

architecture synth of parita is
begin
p <= a(0) xor a(1) xor a(2) xor a(3) xor a(4) xor a(5) xor a(6) xor a(7);
end
```