

Logické kombinační obvody a jejich realizace

ZÁKLADNÍ ZNALOSTI LOGICKÝCH KOMBINAČNÍCH OBVODŮ

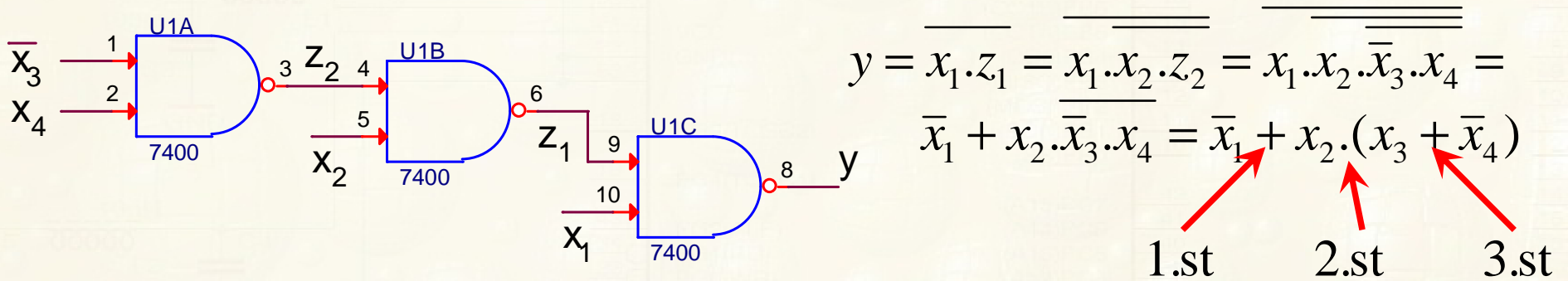
Logický kombinační obvod můžeme realizovat s obvody, které byly k tomuto účelu předurčeny. Existují i obvody vyvinuté pro jiné použití, s kterými se LKO dají realizovat.

- Hradla NAND
- Hradla NOR
- Hradla AND, OR, NEGACE
- AND-OR-INVERT
- Hradla EX-OR a AND
- Multiplexory
- Dekodér a hradla AND nebo OR
- Paměti ROM

IMPLEMENTACE LKO HRADLY NAND

V počátcích se vyráběly obvody realizující tzv. **úplný systém**. To znamená, že s nimi lze realizovat operace AND, OR i NEGACI.

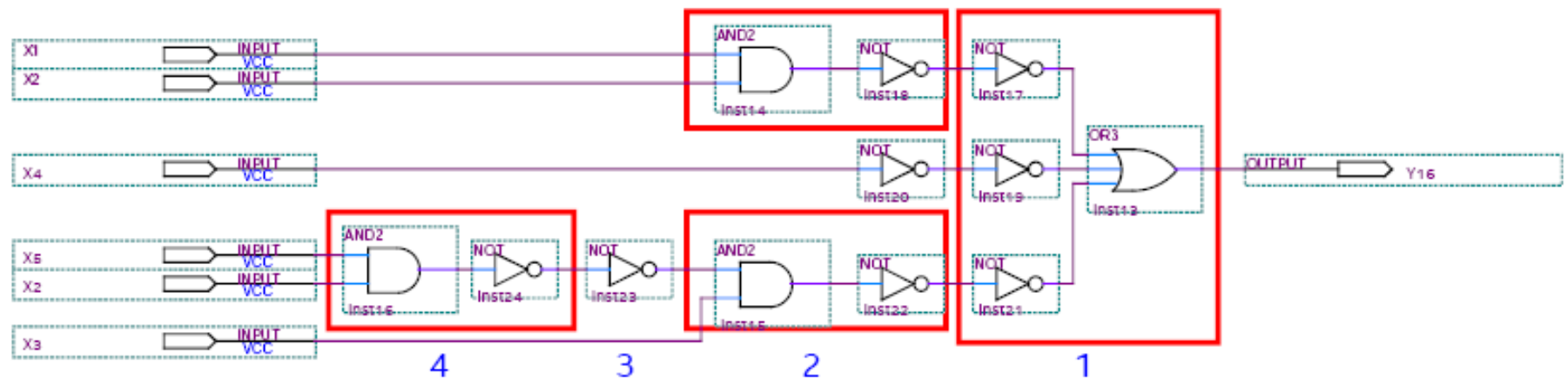
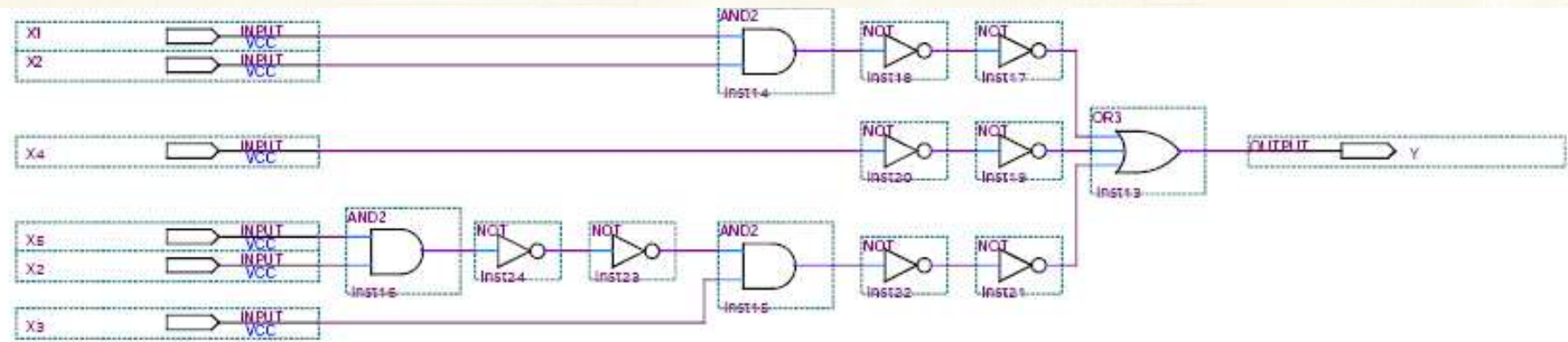
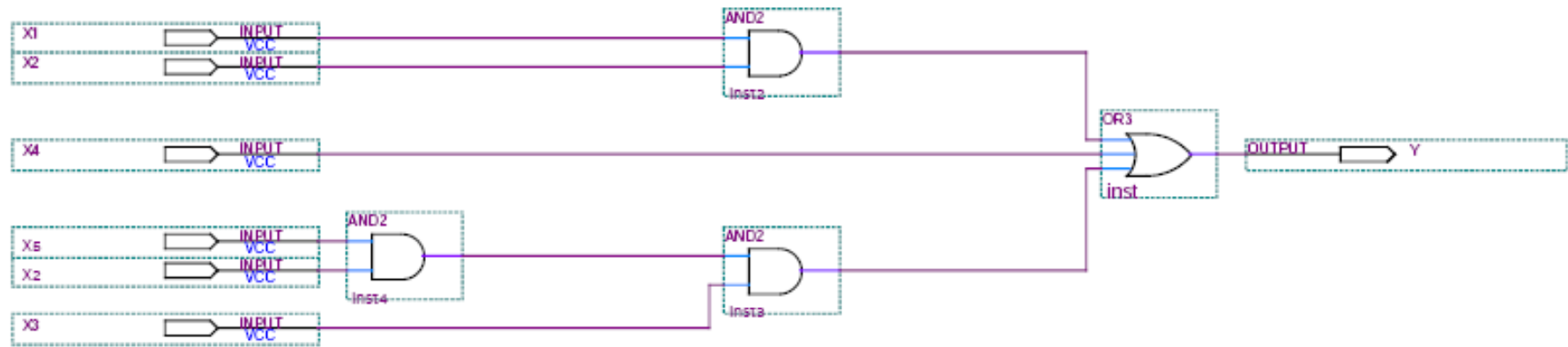
Pro návrh obvodu s logickými členy NAND, odvodíme souvislost mezi logickým výrazem a schématem.



Teorém 1: Každý kombinační obvod realizovaný logickými členy NAND, který má k stupňů ($k=1,2,3, \dots$) modeluje Booleovský výraz, který má v lichých stupních operaci logického součtu nebo negaci a v sudých stupních logického součinu nebo negaci. Vstupní proměnné lichých stupňů jsou ve výrazu v komplementu.

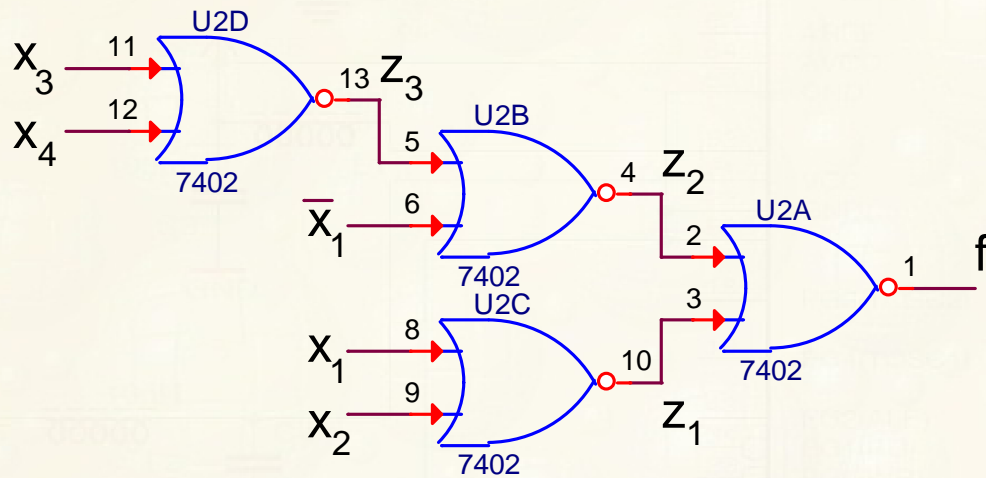
Stupně se začínají počítat od poslední matematické operace až k operaci, kterou je třeba vykonat jako první.

IMPLEMENTACE LKO HRADLY NAND



IMPLEMENTACE LKO HRADLY NOR

Obvody NOR jsou druhým typem obvodů, které realizují tzv. **úplný systém**. Mějme obvod realizovaný logickými členy NOR, pro který můžeme odvodit



$$\begin{aligned}
 y &= \overline{z_1 + z_2} = \overline{(x_1 + x_2) + (\bar{x}_1 + z_3)} = \\
 &= \overline{(x_1 + x_2) + (\bar{x}_1 + (x_3 + x_4))} = \\
 &= \overline{(x_1 + x_2) \cdot (\bar{x}_1 + (x_3 + x_4))} = \\
 &= \overline{(x_1 + x_2) \cdot (\bar{x}_1 + (\bar{x}_3 \cdot \bar{x}_4))}
 \end{aligned}$$

2.st
1.st
2.st
3.st

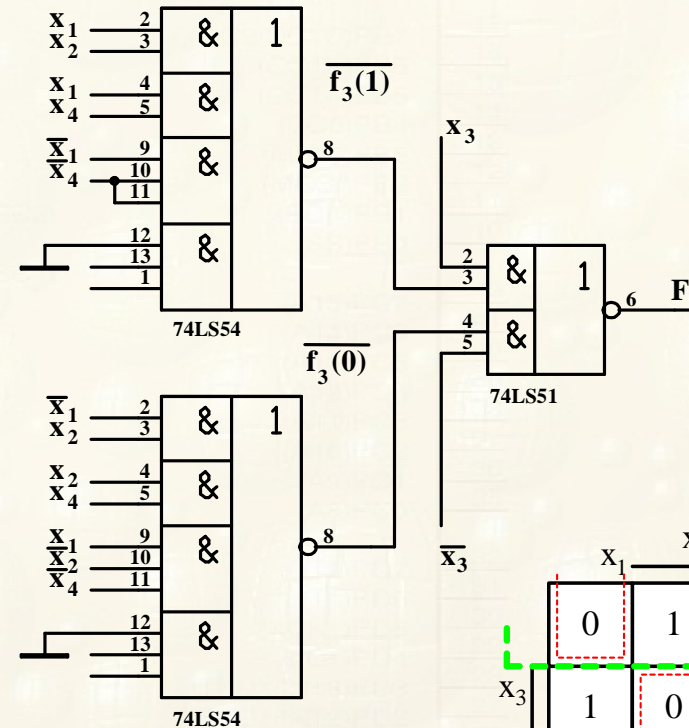
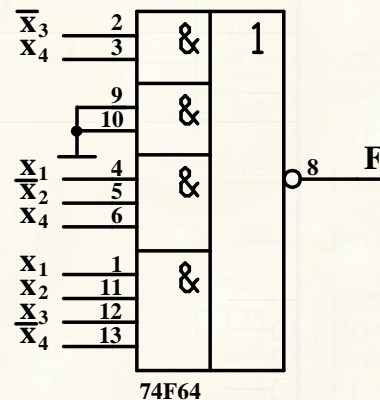
Teorém 2. Každý kombinační obvod realizovaný logickými členy NOR, který má k stupňů ($k=1,2,3, \dots$) modeluje Booleovský výraz, který má v lichých stupních operace logický součin nebo negaci a v sudých stupních logický součet nebo negaci. Vstupní proměnné lichých stupňů jsou ve výrazu v komplementu.

Stupně se počítají stejně jako u NAND. Rozdíl ve složitosti implementace NAND a NOR se využívá v PLA, PAL, GAL, CPLD.

IMPLEMENTACE LKO HRADLY AND-OR-INVERT

Pro realizaci rychlejší dvoustupňové formy LKO vznikly obvody AND-OR-INVERT (stejné zpoždění jako hradlo NAND nebo NOR). Obvod realizuje negaci součtové formy \Rightarrow realizujeme návrh jedno-
stupňového obvodu součtovou formou z nul v Karnaughově mapě.

	x_1	x_2		
	1	1	1	1
x_3	1	1	0	1
x_4	1	0	1	1
	0	0	0	0



Pokud nelze obvod realizovat jednostupňově, rozdělíme funkci (mapu) podle jedné proměnné (například x_3) na dvě části. V sudých stupních realizujeme součtovou formou z log.1 a v lichých z log.0.

	x_1	x_2		
	0	1	0	1
x_3	1	0	1	1
x_4	0	1	1	0
	0	0	1	1

Labels on the right side of the map: $f_3(0)$, $f_3(1)$, $f_3(1)$, $f_3(0)$

Změna polarity (EX-OR) \Rightarrow Výběr výhodnější formy v CPLD

IMPLEMENTACE LKO HRADLY AND A EX-OR

Obvody EX-OR se využívají v **paritním generátoru**, **bitovém komparátoru** nebo jako **programovatelný invertor**. Logickou funkcí je neshoda dvou bitů.

$$A \oplus B = A.\bar{B} + \bar{A}.B$$

Za pomoci Booleovy algebry můžeme pro operaci EX-OR odvodit

$$X \oplus X = 0 \quad X \oplus \bar{X} = 1 \quad 1 \oplus X = \bar{X} \quad 0 \oplus X = X$$

$$X + Y = X \oplus Y \oplus X.Y = X \oplus \bar{X}.Y \quad X \oplus Y = \overline{X \otimes Y}$$

$$X.(Y \oplus Z) = X.Y \oplus X.Z$$

Odvodíme zda je možné realizovat libovolnou logickou funkci pomocí obvodů AND a EX-OR. Mějme libovolnou funkci dvou proměnných v úplné součtové formě

$$f(x_2, x_1) = a_0.\bar{x}_2.\bar{x}_1 + a_1.\bar{x}_2.x_1 + a_2.x_2.\bar{x}_1 + a_3.x_2.x_1 \quad \text{kde } a_i = 0 \text{ nebo } 1$$

Operaci OR můžeme nahradit operací \oplus , protože v daný čas nabývá hodnoty log.1 pouze jeden výraz

$$f(x_2, x_1) = a_0.\bar{x}_2.\bar{x}_1 \oplus a_1.\bar{x}_2.x_1 \oplus a_2.x_2.\bar{x}_1 \oplus a_3.x_2.x_1 \quad \text{kde } a_i = 0 \text{ nebo } 1$$

IMPLEMENTACE LKO HRADLY AND A EX-OR

Nahradíme-li negace proměnných x_1 a x_2 z výše uvedených vztahů výrazem $1 \oplus x_1$ a $1 \oplus x_2$ dostaneme

$$\begin{aligned} f(x_2, x_1) &= a_0 \cdot \bar{x}_2 \cdot \bar{x}_1 \oplus a_1 \cdot \bar{x}_2 \cdot x_1 \oplus a_2 \cdot x_2 \cdot \bar{x}_1 \oplus a_3 \cdot x_2 \cdot x_1 = \\ &= a_0 \cdot (1 \oplus x_2) \cdot (1 \oplus x_1) \oplus a_1 \cdot (1 \oplus x_2) \cdot x_1 \oplus a_2 \cdot x_2 \cdot (1 \oplus x_1) \oplus a_3 \cdot x_2 \cdot x_1 = \\ &= a_0 \oplus (a_0 \oplus a_2) \cdot x_2 \oplus (a_0 \oplus a_1) \cdot x_1 \oplus (a_0 \oplus a_1 \oplus a_2 \oplus a_3) \cdot x_2 \cdot x_1 = \\ &= c_0 \oplus c_2 \cdot x_2 \oplus c_1 \cdot x_1 \oplus c_3 \cdot x_2 \cdot x_1 \end{aligned}$$

Libovolnou funkci lze vyjádřit pomocí obvodů EX-OR a hradel AND realizující součin neinvertovaných vstupních proměnných. Uvedený výraz se označuje jako **Reed–Mullerova kanonická forma**.

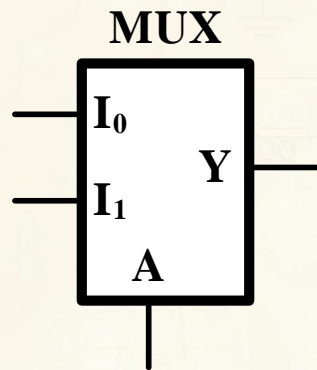
Pro zjednodušení dlouhého odvození je možné vycházet z platnosti vztahu např.

$$\begin{aligned} f(x_4, x_3, x_2, x_1) &= \bar{x}_4 \cdot \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 \oplus \bar{x}_4 \cdot \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 \oplus \bar{x}_4 \cdot x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 \oplus \bar{x}_4 \cdot x_3 \cdot \bar{x}_2 \cdot x_1 = \\ &= \bar{x}_4 \cdot \bar{x}_3 \cdot \bar{x}_2 \cdot (\bar{x}_1 \oplus x_1) \oplus \bar{x}_4 \cdot x_3 \cdot \bar{x}_2 \cdot (\bar{x}_1 \oplus x_1) = \bar{x}_4 \cdot \bar{x}_2 \cdot (\bar{x}_3 \oplus x_3) = \bar{x}_4 \cdot \bar{x}_2 \end{aligned}$$

Po zjednodušení je potřeba již nahradit negované proměnné výše uvedenými vztahy.

IMPLEMENTACE LKO MULTIPLEXERY

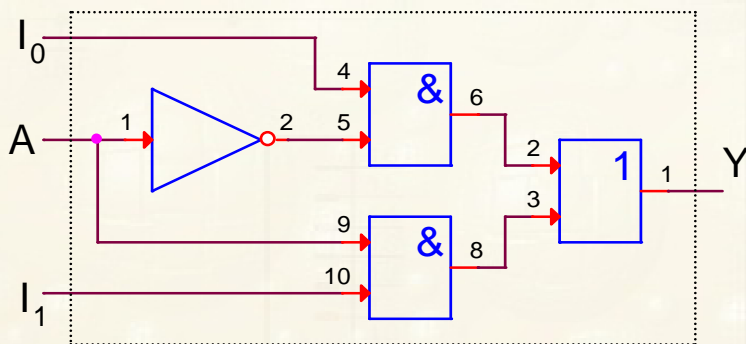
Multiplexer = LKO předurčený pro realizaci elektronického přepínače logických signálů.



I_1	I_0	A	Y
X	0	0	0
X	1	0	1
0	X	1	0
1	X	1	1

Z mapy snadno odvodíme následující logický výraz.

$$Y = \bar{A}.I_0 + A.I_1$$



Multiplexery

- ❑ S dvoustavovým výstupem (15x)
- ❑ S třístavovým výstupem (25x)
- ❑ **Varianty** 4x1ze2 (157,257),
2x1ze4 (153,253), 1x1z8 (151,251)
a 1x1z16 (150,250).

V PLD (Aktek) se využívá bez hazardní realizace k realizaci LKO.

REALIZACE LKO MULTIPLEXEREM

Pro libovolnou funkci 3 proměnných můžeme obecně psát

$$\begin{aligned} F(x_1, x_2, x_3) &= \overline{x_1}.\overline{x_2}.\overline{x_3}.f(0) + \overline{x_1}.\overline{x_2}.x_3.f(1) + \overline{x_1}.x_2.\overline{x_3}.f(2) + \overline{x_1}.x_2.x_3.f(3) + \\ &\quad \overline{x_1}.x_2.\overline{x_3}.f(4) + \overline{x_1}.x_2.x_3.f(5) + x_1.\overline{x_2}.\overline{x_3}.f(6) + x_1.\overline{x_2}.x_3.f(7) \\ &= \overline{x_1}.\overline{x_2}.(\overline{x_3}.f(0) + x_3.f(4)) + \overline{x_1}.x_2.(\overline{x_3}.f(1) + x_3.f(5)) + \\ &\quad \overline{x_1}.x_2.(\overline{x_3}.f(2) + x_3.f(6)) + x_1.\overline{x_2}.(\overline{x_3}.f(3) + x_3.f(7)) \end{aligned}$$

kde $f(i) \in \langle 0, 1 \rangle$ je funkční hodnota pro danou kombinaci proměnných. Vybereme-li pro adresovací vstupy např. proměnné (x_1 a x_2), pak na vstupy I_x patří obsah závorek s (x_3) a $f(i)$. Podle hodnoty $f(i)$ bude závorka nabývat hodnot 1, 0, x_3 a negace x_3 .

Teorém 3. Každým multiplexerem s N adresovacími vstupy můžeme realizovat logickou funkci o $N+1$ vstupních proměnných.

Algebraický návrh LKO:

$$\begin{aligned} F(x_1, x_2, x_3) &= \overline{x_1}.\overline{x_2} + \overline{x_1}.x_3 + \overline{x_1}.x_2 = \overline{x_1}.\overline{x_2}.1 + (\overline{x_1} + \overline{x_1}).\overline{x_2}.x_3 + \overline{x_1}.x_2.1 + \overline{x_1}.x_2.0 = \\ &\quad \overline{x_1}.\overline{x_2}.(0) + \overline{x_1}.\overline{x_2}.(1) + \overline{x_1}.x_2.(1 + \overline{x_3}) + \overline{x_1}.x_2.(\overline{x_3}) \Rightarrow \\ &\quad I_0 = 0, \quad I_1 = 1, \quad I_2 = 1, \quad I_3 = \overline{x_3} \end{aligned}$$

REALIZACE LKO MULTIPLEXEREM

Návrh z upravené Karnaughovy mapy

$$F(x_1, x_2, x_3) = x_1 \cdot \overline{x_2} + x_2 \cdot \overline{x_3} + \overline{x_1} \cdot x_2$$

Adresující proměnné

Zbývající proměnná

	x_1		x_2	
	0	1	1	1
x_3	0	1	0	1
	$I_0=0$	$I_1=1$	$I_3=\overline{x_3}$	$I_2=1$

Pro $A=x_1$ a $B=x_2$

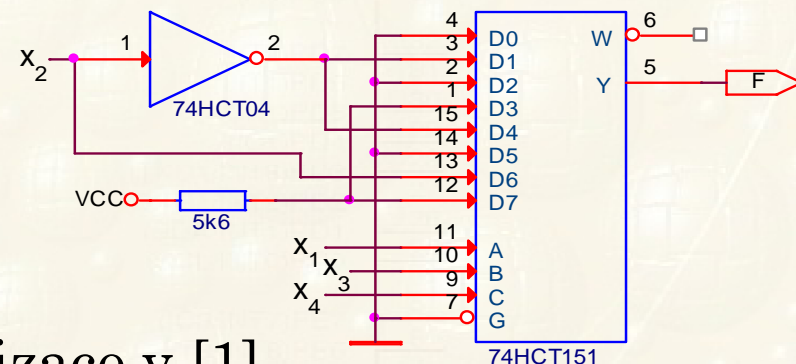
	x_1		x_2	
	0	1	1	1
x_3	0	1	0	1
	$I_0=0$	$I_2=1$	$I_3=\overline{x_3}$	$I_1=1$

Pro $A=x_2$ a $B=x_1$

Pro funkci 4 proměnných

$$F(x_1, x_2, x_3, x_4) = x_1 \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}$$

	x_1		x_3		x_4			
	0	1	1	0	0	1	0	1
x_2	0	0	1	0	1	1	0	0
	$I_0=0$	$I_1=\overline{x_2}$	$I_3=1$	$I_2=0$	$I_6=x_2$	$I_7=1$	$I_5=0$	$I_4=\overline{x_2}$

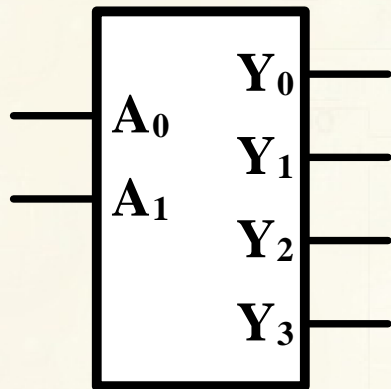


Další možnosti a dvoustupňové realizace v [1].

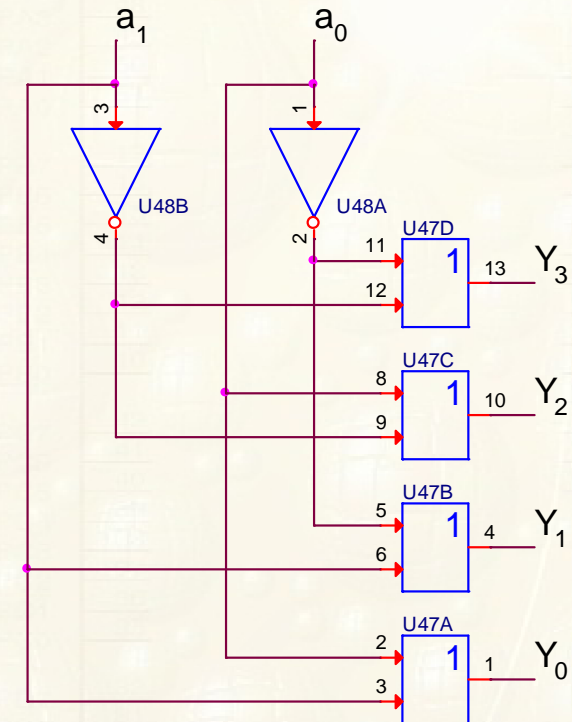
[1] J.Podlešák, P.Skalický : Spínací a číslicová technika, ČVUT, Praha 1994

DEKODÉR

Dekodér je LKO, který **adresu (binární číslo)** převede na aktivaci jednoho z $m=2^n$ výstupů, kde n počet bitů adresy. Pro dvou bitovou adresu bude mít obvod 4 výstupy.



a_1	a_0	$Y_3 Y_2 Y_1 Y_0$
0	0	1 1 1 0
0	1	1 1 0 1
1	0	1 0 1 1
1	1	0 1 1 1



Dekodér

- ❑ Dílčí část paměťových obvodů
- ❑ **Adresový dekodér** - v μP k aktivaci paměti a periférií
- ❑ Umožňuje ušetřit počet vodičů pro řízení
 - Dynamického displeje
 - Řádků klávesnice
 - Indikovat stavy systému.

$$Y_0 = A_1 + A_0$$

$$Y_1 = A_1 + \overline{A_0}$$

$$Y_2 = \overline{A_1} + A_0$$

$$Y_3 = \overline{A_1} + \overline{A_0}$$

IMPLEMENTACE LKO DEKODÉRY A HRADLY AND NEBO OR

Dekodér je obecně převodník jednoho kódu na kód jiný. Například (BIN→7segment, HEX→7segment). Nejčastěji používaný je **dvojkový dekodér** - převod binárního čísla na kód 1 z 2^n , který se dá využít k realizaci LKO.

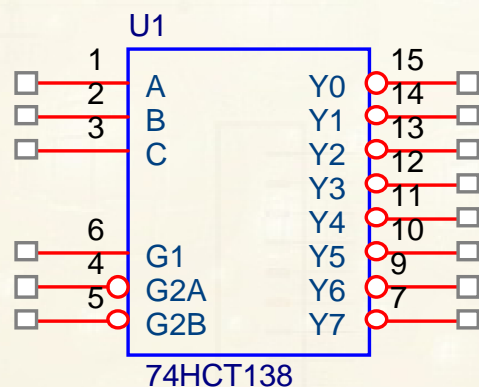
Nejznámější jsou 74139 a 74138, který se používá v μP systémech k realizaci aktivačních signálů pro adresové prostory pamětí a periférií.

$$Y_0 = A + B + C + \overline{G1} + G2A + G2B$$

$$Y_1 = \overline{A} + B + C + \overline{G1} + G2A + G2B$$

...

$$Y_7 = \overline{A} + \overline{B} + \overline{C} + \overline{G1} + G2A + G2B$$



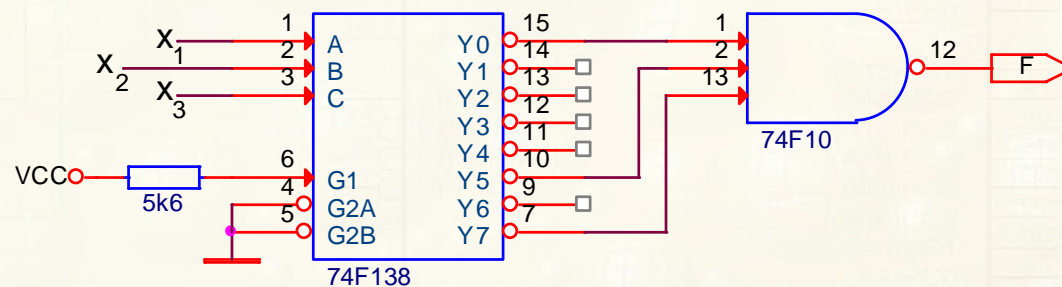
VSTUPY						VÝSTUPY							
C	B	A	G1	G2 A	G2 B	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	1	0	0	1	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	1	1	1	1	0	1
0	1	0	1	0	0	1	1	1	1	1	0	1	1
0	1	1	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	0	0	1	1	0	1	1	1	1	1
1	1	0	1	0	0	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	1	1	1	1	1	1	1
x	x	x	0	x	x	1	1	1	1	1	1	1	1
x	x	x	x	1	x	1	1	1	1	1	1	1	1
x	x	x	x	x	1	1	1	1	1	1	1	1	1

REALIZACE LKO S DEKODÉREM

Bude-li obvod aktivován tj. $G1=1$, $G2A=G2B=0$, potom výstupy indikují jednotlivé kombinace proměnných přivedené na vstupy C, B a A. Např. $C=1$, $B=1$ a $A=0$ (binárně číslo 6) \Rightarrow výstup $Y_6=0$ a zbývající $Y_j=1$. Například logickou funkci tří proměnných

$$F(x_1, x_2, x_3) = x_1 \cdot \bar{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3$$

pak můžeme zrealizovat následujícím obvodem.



Na aktivační vstupy můžeme přivést

- Konkrétní aktivní úroveň
- Další vstupní proměnné
- Výstupy jiných dekodérů

Hlavní použití obvodu - Realizace **adresového dekodéru**

- **Šetření počtu V/V** vývodů procesoru
Petr Skalický, katedra radioelektroniky

IMPLEMENTACE LKO PAMĚTMI (ROM NEBO RAM)

Paměť – součástka generující v aktivovaném stavu na svých výstupech uloženou hodnotu z adresy přivedené na její adresové vstupy. Pro paměť s 5 adresovými vstupy a 8 výstupy můžeme psát tyto rovnice

$$Q_7 = \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_7(0) + \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1} \cdot A_0 \cdot f_7(1) + \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot A_1 \cdot \overline{A_0} \cdot f_7(2) + \\ \overline{A_4} \cdot \overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_7(3) + \overline{A_4} \cdot \overline{A_3} \cdot A_2 \cdot A_1 \cdot \overline{A_0} \cdot f_7(4) + \overline{A_4} \cdot A_3 \cdot \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_7(5) + \text{atd.}$$

...

$$Q_1 = \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_1(0) + \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1} \cdot A_0 \cdot f_1(1) + \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot A_1 \cdot \overline{A_0} \cdot f_1(2) + \\ \overline{A_4} \cdot \overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_1(3) + \overline{A_4} \cdot \overline{A_3} \cdot A_2 \cdot A_1 \cdot \overline{A_0} \cdot f_1(4) + \overline{A_4} \cdot A_3 \cdot \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_1(5) + \text{atd.}$$

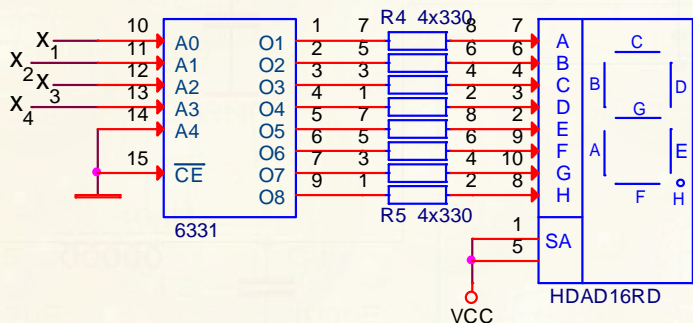
$$Q_0 = \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_0(0) + \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1} \cdot A_0 \cdot f_0(1) + \overline{A_4} \cdot \overline{A_3} \cdot \overline{A_2} \cdot A_1 \cdot \overline{A_0} \cdot f_0(2) + \\ \overline{A_4} \cdot \overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_0(3) + \overline{A_4} \cdot \overline{A_3} \cdot A_2 \cdot A_1 \cdot \overline{A_0} \cdot f_0(4) + \overline{A_4} \cdot A_3 \cdot \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} \cdot f_0(5) + \text{atd.}$$

kde $f_j(k)$ je funkční hodnota na j -tém výstupu pro k -tou kombinaci (**adresu**) přivedou na adresové vstupy.

Teorém 4. Každou paměť s N adresovacími vstupy a M výstupy můžeme realizovat M logických funkcí o N vstupních proměnných.

REALIZACE LKO PAMĚTÍ

Realizace dekodéru BCD → 7- segmentový displej. Druhý a čtvrtý sloupec tabulky = **pravdivostní tabulka**. Přivedeme-li x1 na A0, x2 na A1, atd., pak druhý sloupec = **adresa paměti**. Bude-li segment A připojen na Q0, B na Q1, atd. pak čtvrtý sloupec představuje obsah daného paměťového místa.



Stav Adresa	Proměnné x4,x3,x2,x1	Zobrazený znak	Segment G F E D C B A
0	0000	□	1 0 0 0 0 0 0
1	0001		1 1 0 0 1 1 1
2	0010	2	0 0 1 0 0 1 0
3	0011	3	0 0 0 0 0 1 1
4	0100	4	0 1 0 0 1 0 1
5	0101	5	0 0 0 1 0 0 1
6	0110	6	0 0 0 1 0 0 0
7	0111	7	1 1 0 0 0 1 1
8	1000	8	0 0 0 0 0 0 0
9	1001	9	0 0 0 0 0 0 1
10 až 15	1010 až 1111	nic	1 1 1 1 1 1 1

Výpis obsahu paměti

Adresa	Obsah jednotlivých míst			
0x0000	0x40	0x67	0x12	0x03
0x0004	0x25	0x09	0x08	0x63
0x0008	0x00	0x01	0x7F	0x7F
0x000C	0x7F	0x7F	0x7F	0x7F

Pro programátory obvykle potřebujeme tzv. INTEL HEX

:10000000406712032509086300017F7F7F7F7F60

:00000001FF

REALIZACE PAMĚTÍ V PLD POMOCÍ VHDL

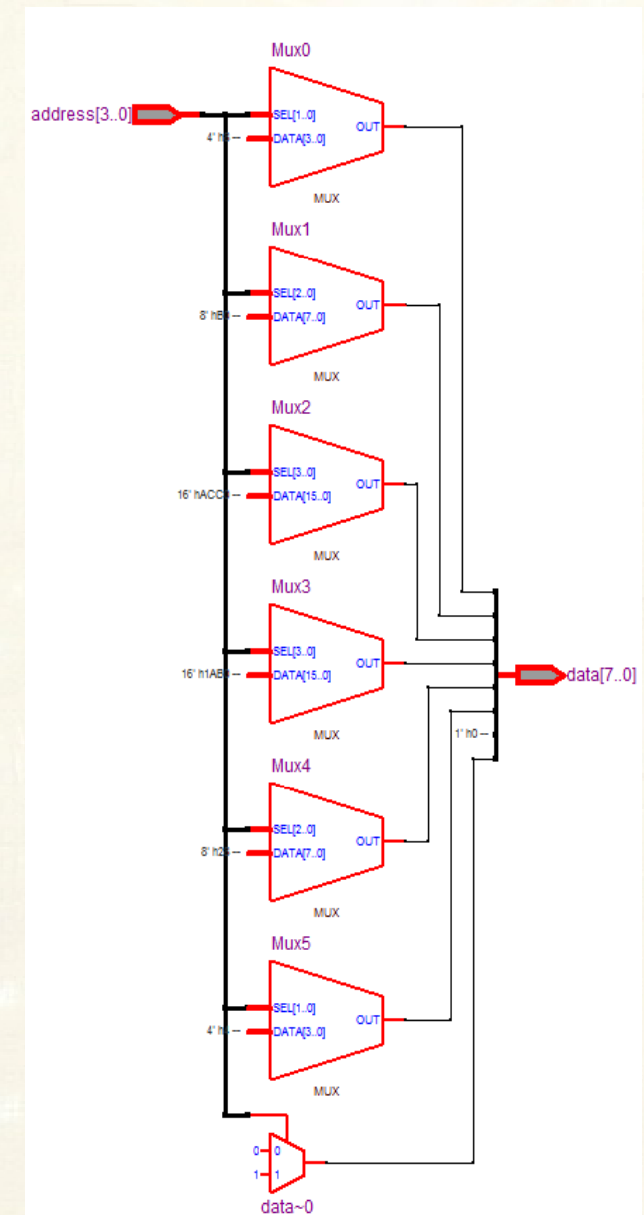
Zápis realizaci ROM ve VHDL (Quartus II)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ROM16x8 is
port (address : in INTEGER range 0 to 15;
      data : out std_logic_vector (7 downto 0));
end entity ROM16x8;

architecture kvadrator of ROM16x8 is
type rom_array is array (0 to 15) of
std_logic_vector (7 downto 0);

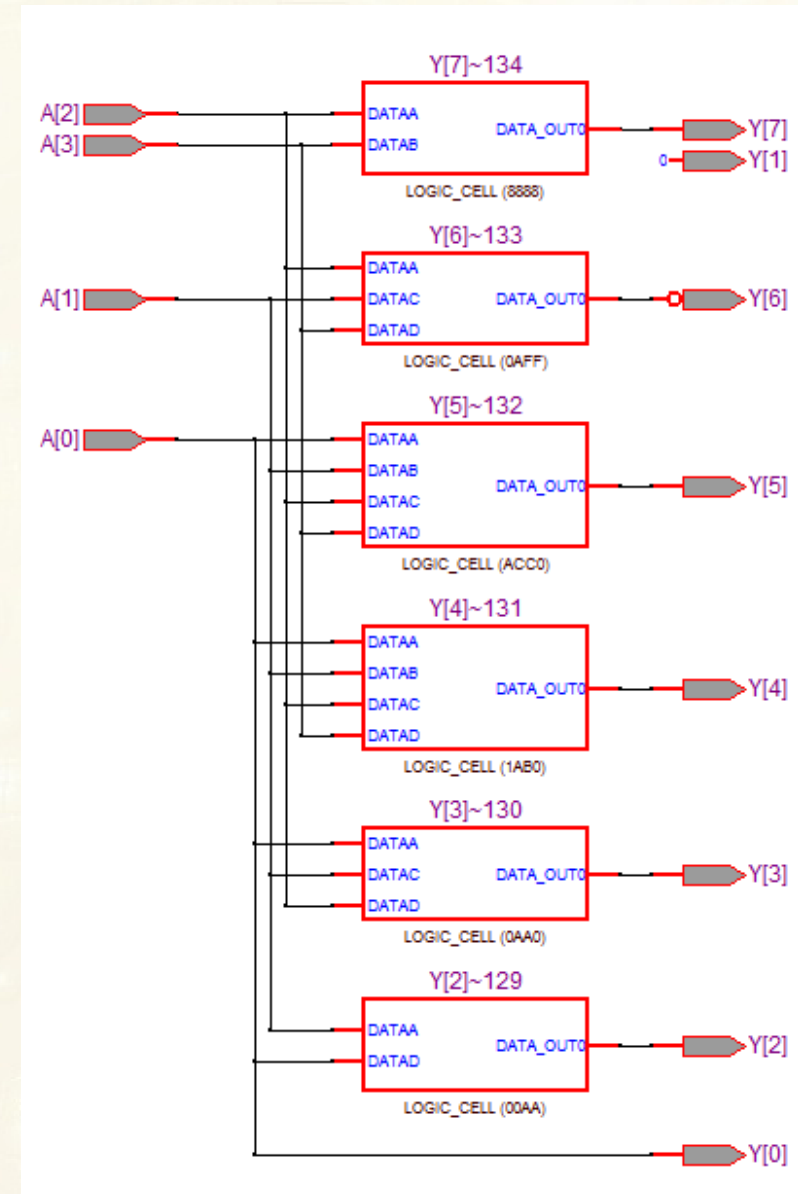
constant rom : rom_array :=
  ("00000000", "00000001",
   "00000100", "00001001",
   "00010000", "00011001",
   "00100100", "00110001",
   "01000000", "01010001",
   "01100100", "01111001",
   "10010000", "10101001",
   "11000100", "11100001");
begin
  data <= rom(address);
end architecture kvadrator;
```



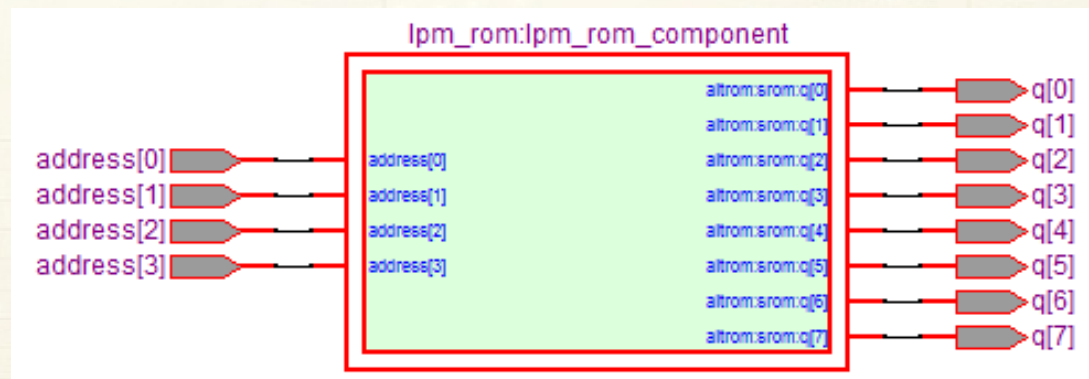
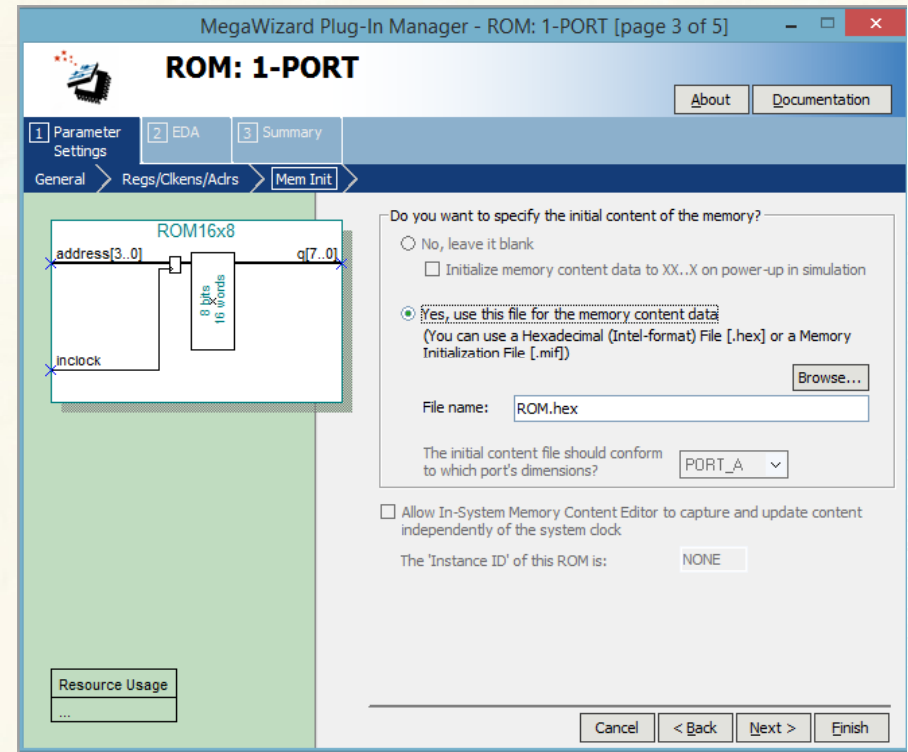
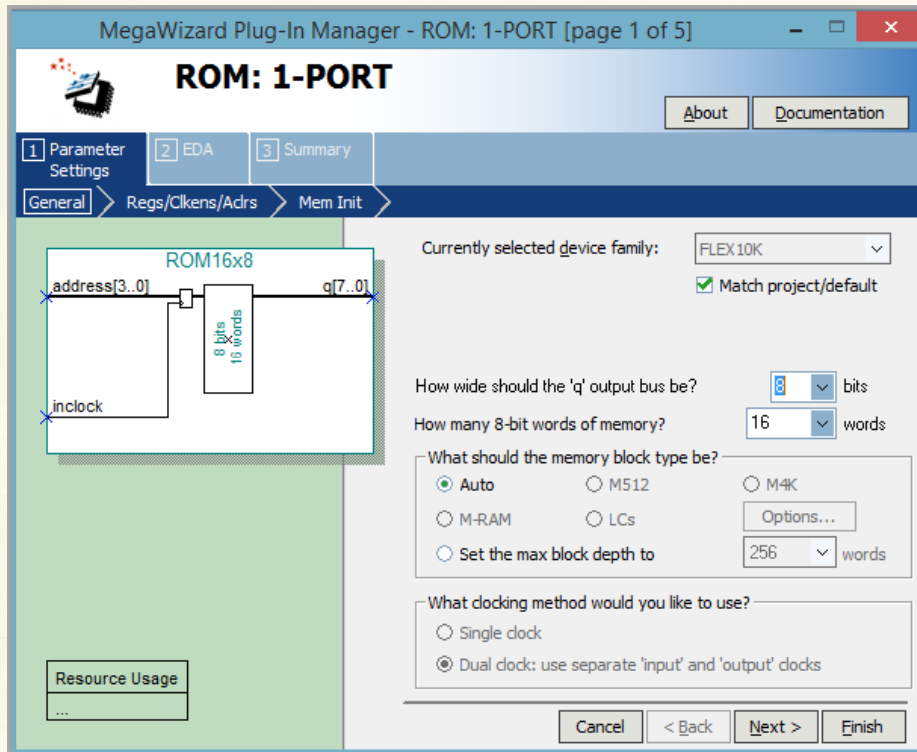
REALIZACE PAMĚTÍ V PLD POMOCÍ AHDL

Zápis realizaci ROM v AHDL (Quartus II)

```
SUBDESIGN ROM_AHDL
( A[3..0]      : INPUT;
  Y[7..0]      : OUTPUT; )
BEGIN
  TABLE
    A[3..0] => Y[7..0];
    B"0000" => B"00000000";
    B"0001" => B"00000001";
    B"0010" => B"00000100";
    B"0011" => B"00001001";
    B"0100" => B"00010000";
    B"0101" => B"00011001";
    B"0110" => B"00100100";
    B"0111" => B"00110001";
    B"1000" => B"01000000";
    B"1001" => B"01010001";
    B"1010" => B"01100100";
    B"1011" => B"01111001";
    B"1100" => B"10010000";
    B"1101" => B"10101001";
    B"1110" => B"11000100";
    B"1111" => B"11100001";
  END TABLE;
END;
```



REALIZACE LKO PAMĚTÍ RAM



Obsah paměti RAM může být realizován ze soubodu Intel HEX

:100000000010409101924314051647990A9C4E118

:00000001FF

➤ **Aritmetické obvody**

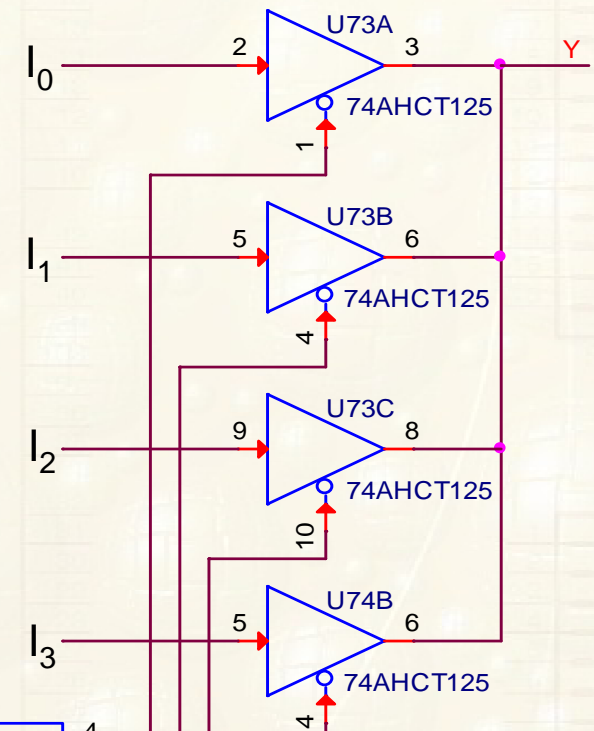
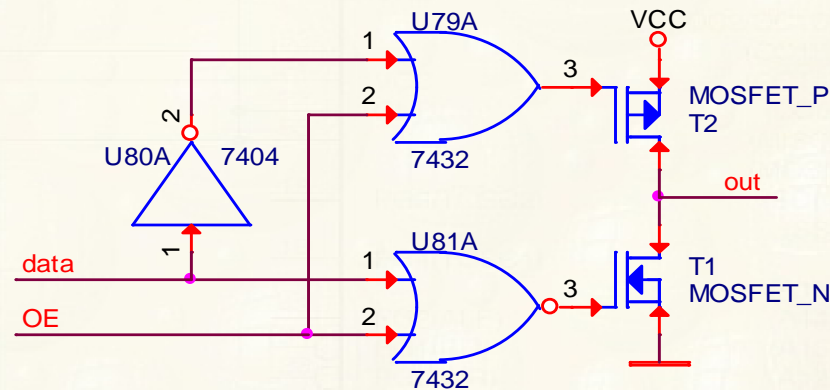
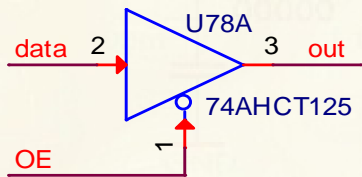
- Obvody pro porovnání čísel
- Sčítačky
- Odečítačky
- Násobičky
- Děličky
- ALU – aritmeticko-logické jednotky
- Obvody pro úpravu čísel (vytváření doplňků)
- „Barelshifter“ - vícebitové aritmetické/logické posuny čísel

➤ **Kódovací a dekódovací obvody**

- Převodníky kódů (BCD, BCD+3, BIN, Gray, Grey+3, Teploměřový kód, 7 segment, Prioritní kodér, Paritní generátor atd.
- Samoopravné blokové kódy

OBVODY S TŘÍSTAVOVÝM VÝSTUPEM

Třetím stavem u těchto obvodů je stav Z (vysoká impedance). Oba tranzistory v koncovém stupni jsou v nevodivém stavu. Vývod **out** vůči Vcc a GND představuje velký odpor (stovky kΩ až MΩ) a malou kapacitu (jednotky pF). Aktivačním vývodem je vstup OE.

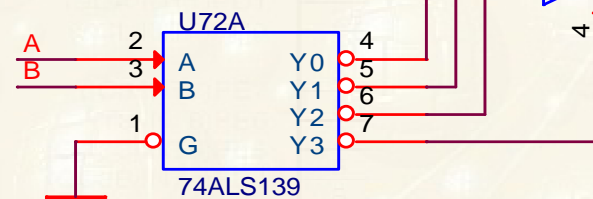


data	OE	Y
0	0	0
1	0	1
X	1	Z

Výstupy mohou být spojovány, **aktivní smí být pouze jeden.**

Příklad MUX 1 ze 4 pomocí dekodéru a třístavových budičů.

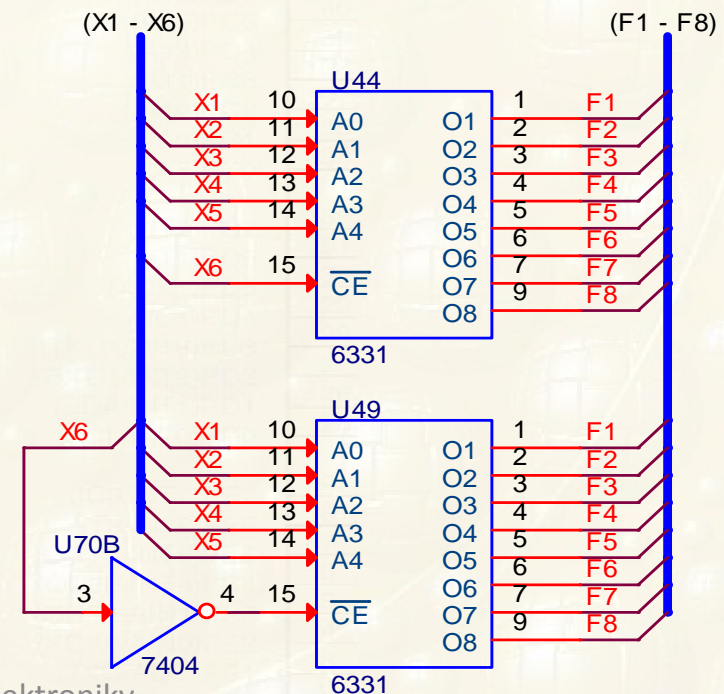
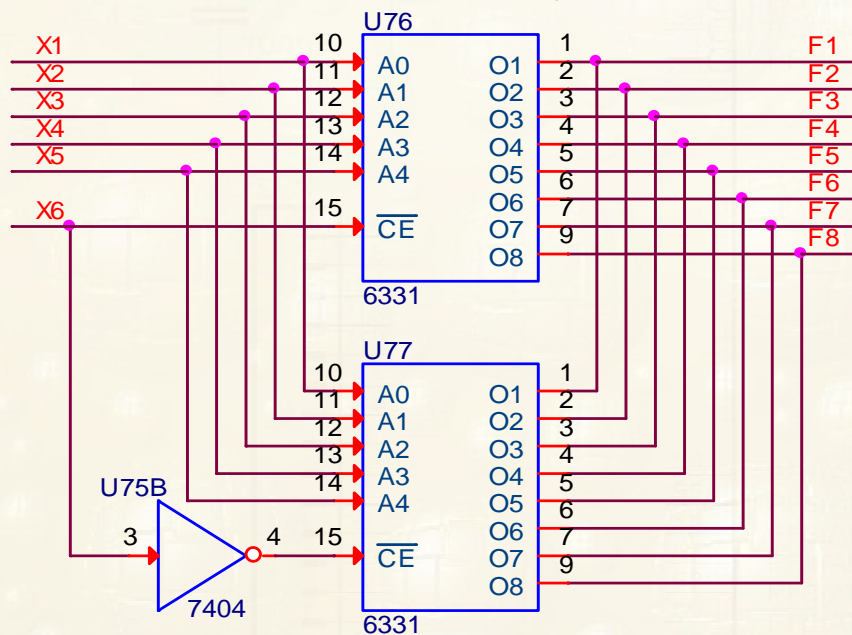
Aktivací jednoho budiče přeneseme hodnotu z jednoho vstupu I_n na výstup.



POUŽITÍ TŘÍSTAVOVÝCH OBVODŮ

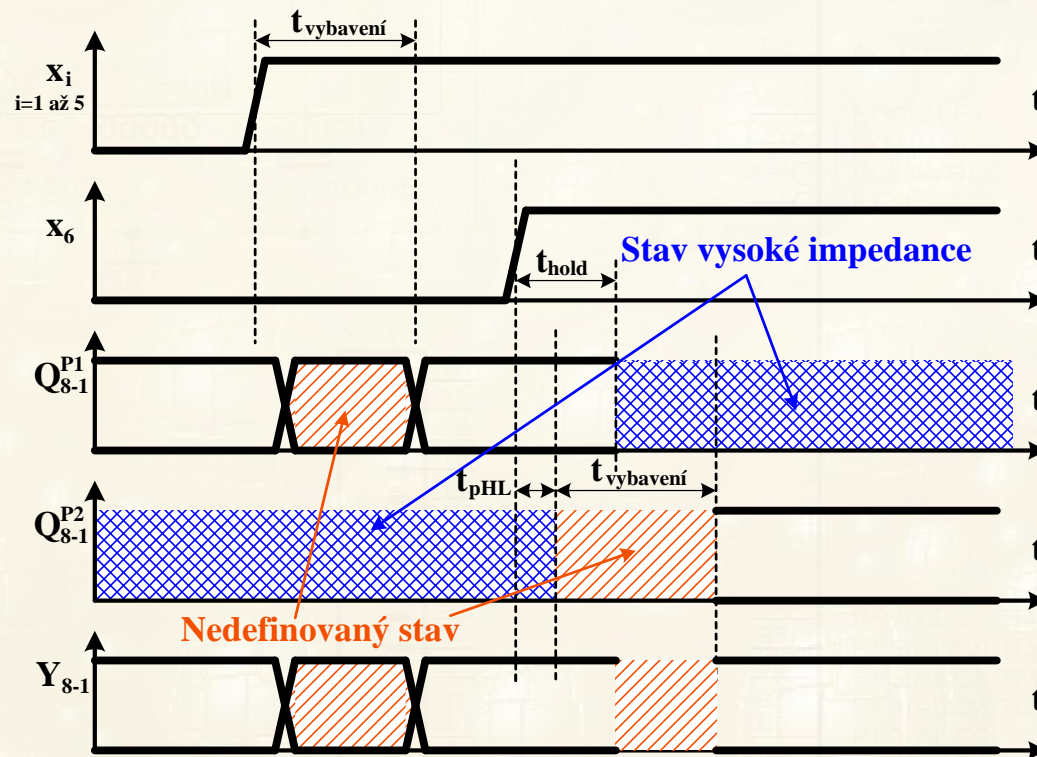
Nejčastější použití v systémech, kde přenášíme informace oběma směry mezi mnoha obvody po společných přenosových vodičích (**datová sběrnice**). Mikropočesory, přístrojové a komunikační sběrnice, propojení řady registrů, atd.).

Příklad - **zvětšení kapacity** paměti pomocí dvou nebo více obvodů. Na obrázku realizace 8 funkcí o 6 proměnných s pamětmi PROM 32x8bitů. Pět proměnných přivedeme na adresovací vstupy, šestou na **aktivační vstup OE** paměti. Každá paměť tak realizuje ½ kombinací požadovaných funkcí.



POUŽITÍ TŘÍSTAVOVÝCH OBVODŮ

Uvedený **obvod nemá stejné vlastnosti** jako realizace s obvody s dvoustavovým výstupem. Je potřeba zajistit, aby současně **nebyly aktivní obě paměti** (zkrat mezi výstupy s rozdílným výstupem). Výstupy budou ovlivněny časovými parametry třístavových budičů (doba aktivace/deaktivace budiče a tvorba aktivačních signálů). U pamětí navíc přistoupí ještě **čas vybavení** (od platné adresy nebo aktivačního signálu paměti).



Analogicky můžeme postupovat i s ostatními obvody, které mají třístavový výstup. Např. Registry, Latch, MUX (řada 25x), atd.

