Logické kombinační obvody a jejich realizace

ZÁKLADNÍ ZNALOSTI LOGICKÝCH KOMBINAČNÍCH OBVODŮ

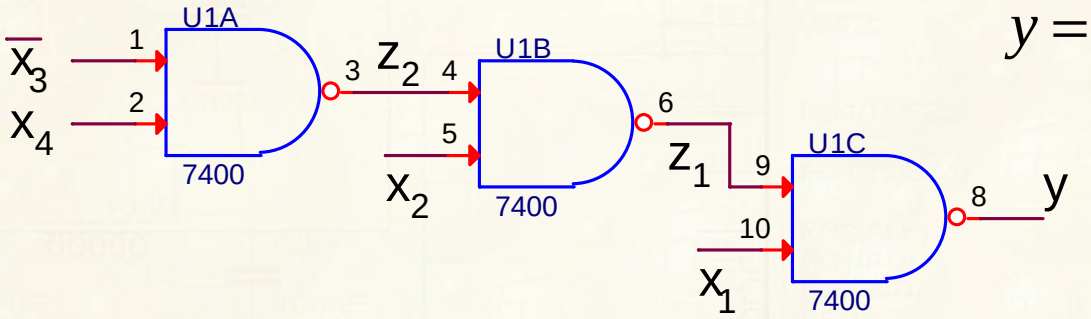
Logický kombinační obvod můžeme realizovat s těmito obvody, které byly k tomuto účelu předurčeny. Existují i obvody vyvinuté pro jiné použití, ale dají se s nimi LKO realizovat.

- 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, kdy byly problémy s výrobou, se nejprve vyráběly obvody, které realizují tzv. **úplný systém**. To znamená, že s nimi umíme realizovat operaci AND, OR i NEGACI.

Mějme obvod realizovaný logickými členy NAND, pro který můžeme odvodit



$$y = \overline{x_1.z_1} = \overline{x_1.x_2.z_2} = \overline{x_1.x_2.\overline{x}_3.x_4} =$$
$$\overline{x}_1 + x_2.\overline{x}_3.x_4 = \overline{x}_1 + x_2.(x_3 + \overline{x}_4)$$

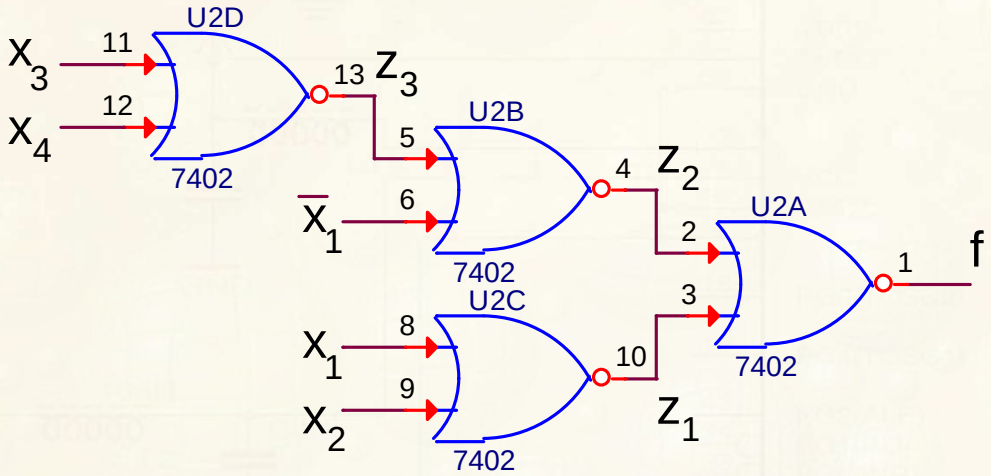
1.st 2.st 3.st

Teorém 1: Každý kombinační obvod realizovaný logickými členy NAND, který má k stupňů (k=1,2,3,) 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 po operaci, kterou je třeba vykonat jako první.

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



$$y = \overline{z_1 + z_2} = \overline{\overline{(x_1 + x_2)} + \overline{(\bar{x}_1 + z_3)}} =$$

$$\overline{\overline{(x_1 + x_2)} + \overline{(\bar{x}_1 + \overline{(x_3 + x_4)})}} =$$

$$(x_1 + x_2).\overline{(\bar{x}_1 + \overline{(x_3 + x_4)})} =$$

$$(x_1 + x_2).(\bar{x}_1 + (x_3 + x_4)) =$$

$$(x_1 + x_2).(\bar{x}_1 + (\bar{x}_3.\bar{x}_4))$$

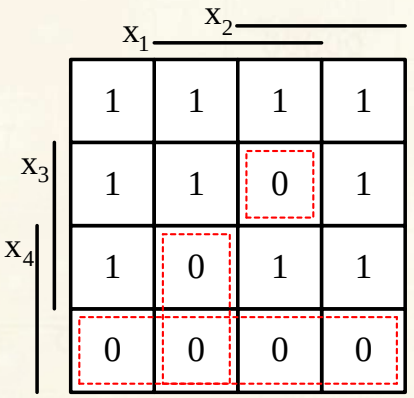
2.st 1.st 2.st 3.st

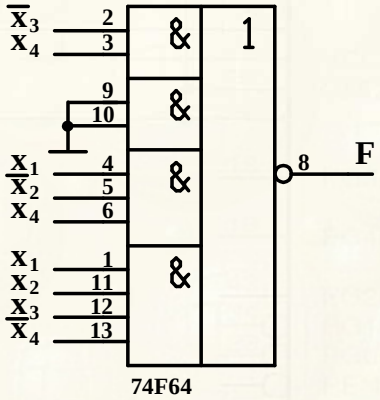
Teorém 2. Každý kombinační obvod realizovaný logickými členy NOR, který má k stupňů (k=1,2,3,) modeluje Booleovský výraz, který má v lichých stupních operace logický součin nebo negaci a v sudých stupních logický součet nebo negaci. Vstupní proměnné lichých stupňů jsou ve výrazu v komplementu.

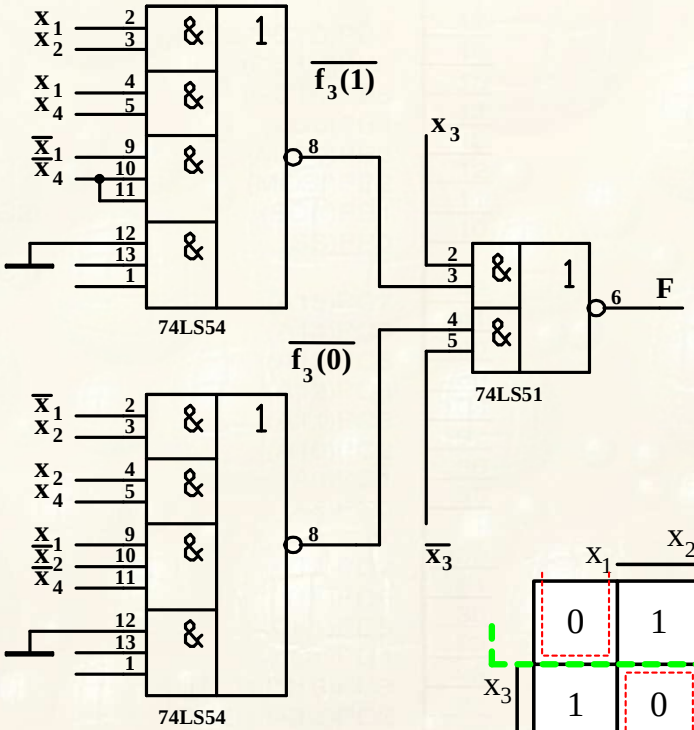
Stupně se počítají stejně jako u NAND. Složitost implementace NAND, NOR využití v PLA,PAL,GAL,CPLD.

IMPLEMENTACE LKO HRADLY AND-OR-INVERT

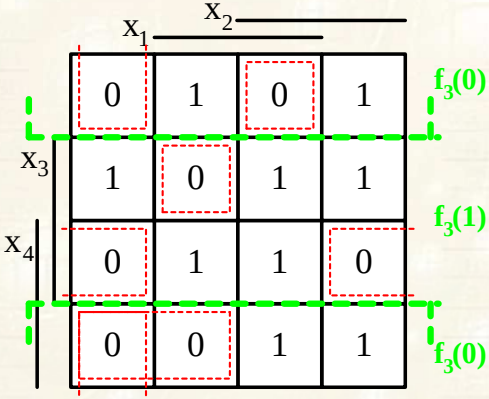
Pro realizaci rychlejších LKO (přenos ve sčítačkách) vznikly obvody AND-OR-INVERT (stejné zpoždění jako hradlo NAND nebo NOR). Obvod realizuje o negaci součtové formy, realizujeme návrh jednostupňového obvodu součtovou formou z nul v Karnaughově mapě.







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



IMPLEMENTACE LKO HRADLY AND A EX-OR

Obvody EX-OR se využívají v paritním generátoru, bitovém komparátoru nebo jako programovatelného invertoru. Jeho logická funkce je dána následujícím výrazem, který indikuje neshodu dvou bitů log.1.

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

Za pomoci Booleovy algebry můžeme pro operaci EX-OR odvodit

$$X \oplus X = 0 \qquad X \oplus \overline{X} = 1 \qquad 1 \oplus X = \overline{X} \qquad 0 \oplus X = X$$

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

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

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

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

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

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

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

$$f(x_2, x_1) = a_0.\bar{x}_2.\bar{x}_1 \oplus a_1.\bar{x}_2.x_1 \oplus a_2.x_2.\bar{x}_1 \oplus a_3.x_2.x_1 =$$

$$a_0.(1 \oplus x_2).(1 \oplus x_1) \oplus a_1.(1 \oplus x_2).x_1 \oplus a_2.x_2.(1 \oplus x_1) \oplus a_3.x_2.x_1 =$$

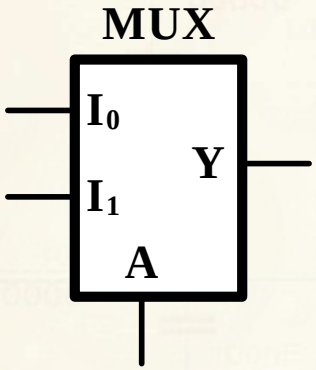
$$a_0 \oplus (a_0 \oplus a_2).x_2 \oplus (a_0 \oplus a_1).x_1 \oplus (a_0 \oplus a_1 \oplus a_2 \oplus a_3).x_2.x_1 =$$

$$c_0 \oplus c_2.x_2 \oplus c_1.x_1 \oplus c_3.x_2.x_1$$

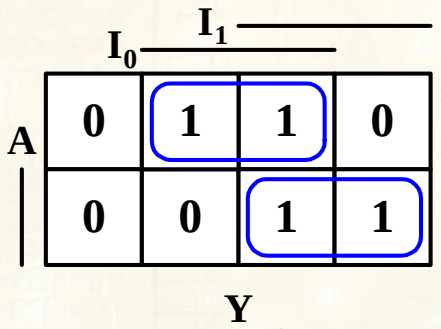
Libovolnou funkci lze vyjádřit pomocí obvodů EX-OR a hradel AND realizující součin neinvertovaných vstupních proměnných. Uvedený výraz se označuje jako Reed–Mullerova kanonická forma.

IMPLEMENTACE LKO MULTIPLEXERY

Multiplexer je LKO předurčený pro realizaci elektronického přepínače logických signálů. Pro přepínač 1 ze 2 vstupů můžeme nakreslit následující značku, kde vstup A představuje adresovací vstup pro výběr vstupu I_A.



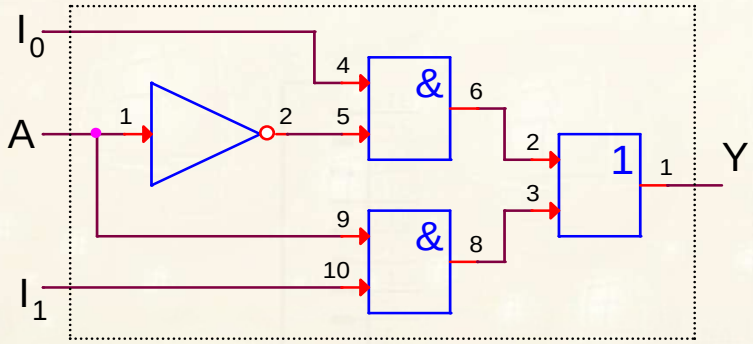
I_1	I_0	A	Y
X	0	0	0
X	1	0	1
0	X	1	0
1	X	1	1



Z mapy snadno odvodíme následující logický výraz,

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

jehož realizace je zobrazena.



Multiplexery se vyrábí v řadě 15x a 25x ve variantě 4x1ze2 (157,257), 2x1ze4 (153,253), 1x1z8 (151,251) a 1x1z16 (150,250).Řada 15x disponuje dvoustavovým výstupem a řada 25x disponuje třístavovým výstupem. V programovatelných obvodech se využívá bezhazardní realizace (Aktel).

REALIZACE LKO MULTIPLEXEREM

Pro libovolnou funkci 3 proměnných můžeme obecně psát

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

kde f(i)∈<0,1> je funkční hodnota pro danou kombinaci proměnných. Na adresovací vstupy MUX vybereme například proměnné (x_1 a x_2), na datové vstupy patří obsah závorek se vstupní proměnou (x_3) a funkčními hodnotami funkce. V závislosti na f(i) bude závorka nabývat hodnot 1, 0, x_3 a negace x_3. Návrh LKO: Algebraicky

$$F(x_1,x_2,x_3)=x_1.\overline{x_2}+x_2.\overline{x_3}+\overline{x_1}.x_2=x_1.\overline{x_2}.1+(x_1+\overline{x_1}).x_2.\overline{x_3}+\overline{x_1}.x_2.1+\overline{x_1}.\overline{x_2}.0=$$
$$\overline{x_1}.\overline{x_2}.(0)+x_1.\overline{x_2}.(1)+\overline{x_1}.x_2.(1+\overline{x_3})+x_1.x_2.(\overline{x_3})\Rightarrow$$
$$I_0=0,\qquad I_1=1,\qquad I_2=1,\qquad I_3=\overline{x_3}$$

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

REALIZACE LKO MULTIPLEXEREM

Z upravené Karnaughovy mapy

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

Adresující proměnné

Zbývající proměnná

x_3 \ 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

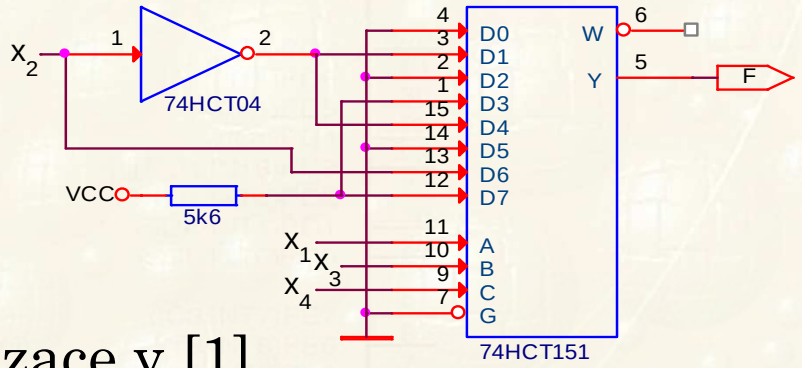
x_3 \ x_1 x_2				
	0	1	1	1
x_3	0	1	0	1
	I_0=0	I_2=1	I_3=$\overline{x_3}$	I_1=1

Pro A=x_2 a B=x_1

Pro funkci 4 proměnných

$$F(x_1, x_2, x_3, x_4) = x_1.x_3 + \overline{x}_1.x_2.x_3.x_4 + \overline{x}_1.\overline{x}_2.\overline{x}_3.x_4 + x_1.\overline{x}_2.\overline{x}_3.\overline{x}_4$$

x_2 \ 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=$\overline{x_2}$

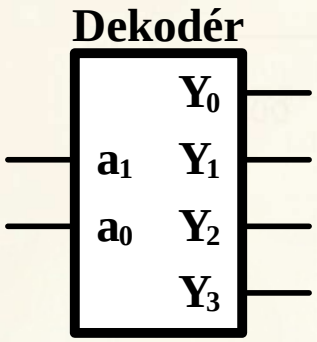


Další možnosti a dvoustupňové realizace v [1].

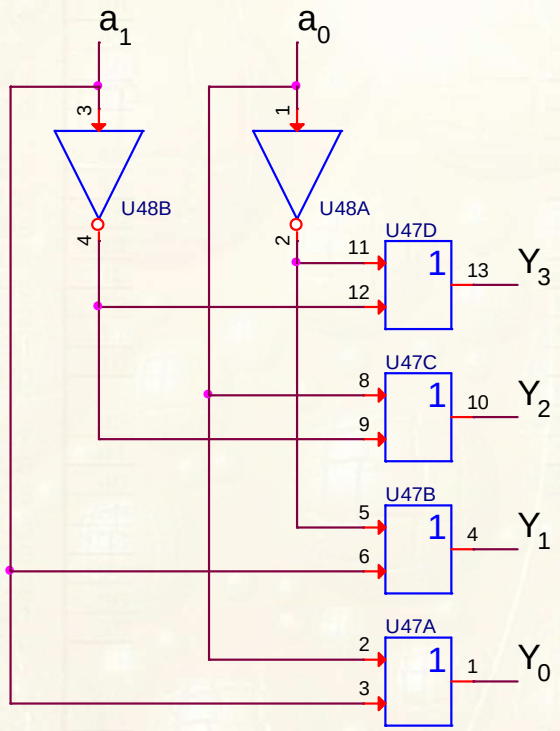
[1] J.Podlešák, P.Skalický : Spínací a číslicová technika, ČVUT, Praha 1994

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 – je dílčí částí paměťového obvodu, používá se jako adresový dekodér v mikroprocesorových systémech (aktivace pamětí a periferií), může ušetřit počet vodičů potřebných k řízení dynamického displeje (segmenty jsou postupně rozsvěcovány) nebo řádků klávesnice a indikovat určité stavy systému.

$$Y_3 = \overline{a}_1 + \overline{a}_0$$
$$Y_2 = \overline{a}_1 + a_0$$
$$Y_1 = a_1 + \overline{a}_0$$
$$Y_0 = a_1 + a_0$$

IMPLEMENTACE LKO DEKODÉRY A HRADLY AND NEBO OR

Obecně je dekodér převodník jednoho kódu na kód jiný (např. BIN→7segment, HEX →7segment). Nejznámějším je dvojkový dekodér realizující převod binárního čísla na kód 1 z 2^n a který se dá využít k realizaci LKO.

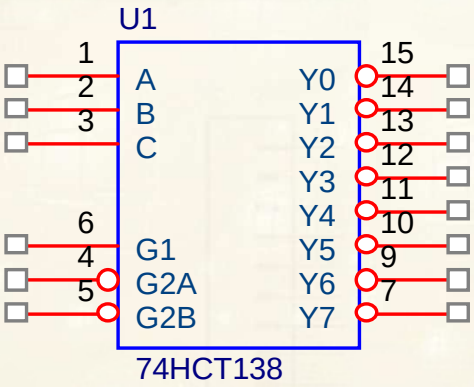
Nejznámějším obvodem je 74138, který lze popsat uvedenými rovnicemi. Používaný µP systémech realizaci aktivačních signálů pro adresové prostory pamětí a periferie.

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

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

…

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



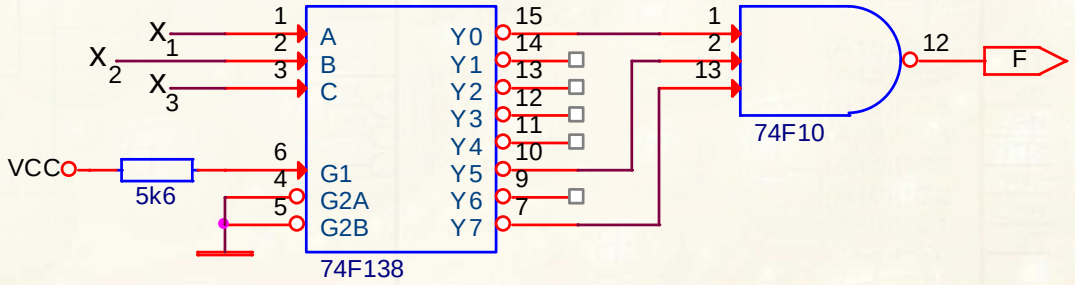
VSTUPY						VÝSTUPY							
C	B	A	G1	G2 A	G2 B	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	1	0	0	1	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	1	1	1	1	0	1
0	1	0	1	0	0	1	1	1	1	1	0	1	1
0	1	1	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	0	0	1	1	0	1	1	1	1	1
1	1	0	1	0	0	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	1	1	1	1	1	1	1
x	x	x	0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

Budou-li splněny podmínky aktivace obvodu tj. G1=1, G2A=G2B=0, potom výstupy indikují výskyt jednotlivých kombinací proměnných přivedených na vstupy C, B a A (např. bude-li C=1, B=1 a A=0 (binárně číslo 6), potom výstup Y_6=0 a zbývající Y_j budou v log.1. Například logickou funkci tří proměnných

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

pak můžeme zrealizovat následujícím obvodem.



Na aktivační vstupy můžeme přivést konkrétní aktivní úrovně nebo další vstupní proměnné pokud to v realizaci potřebujeme. Hlavní použití tohoto obvodu spočívá v realizaci **adresového dekodéru**, při umísťování externích pamětí do definovaných adresových prostorů nebo k vytváření většího počtu aktivačních signálů pomocí menšího počtu adresujících vývodů (šetření počtu vstupně/ výstupních vývodů (dále V/V) procesoru).

IMPLEMENTACE LKO PAMĚTMI (ROM NEBO RAM)

Paměť je součástka, která v aktivovaném stavu generuje na svých výstupech uloženou binární kombinaci z adresy určené kombinací vstupních proměnných připojených na její adresové vstupy. Pro paměť s 5 adresovými vstupy a 8 výstupy můžeme psát tyto rovnice

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

…

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

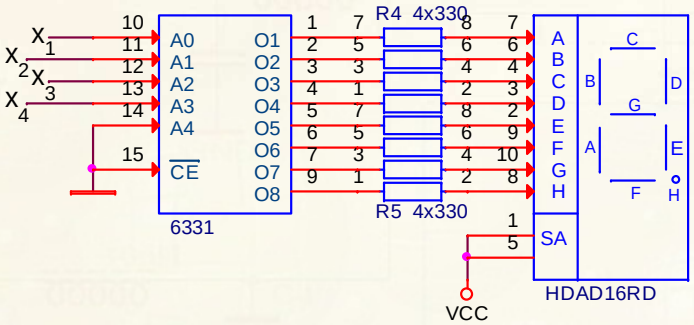
$$Q_0 = \overline{A_4}.\overline{A_3}.\overline{A_2}.\overline{A_1}.\overline{A_0}.f_0(0) + \overline{A_4}.\overline{A_3}.\overline{A_2}.\overline{A_1}.A_0.f_0(1) + \overline{A_4}.\overline{A_3}.\overline{A_2}.A_1.\overline{A_0}.f_0(2) +$$
$$\overline{A_4}.\overline{A_3}.\overline{A_2}.A_1.A_0.f_0(3)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

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

Teorém 4. Každou pamětí 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 proměnou x1 na vodič 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.



Obsah paměti se vypisujeme v hexadecimálním vyjádření např. takto

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

Stav Adresa	Proměnné x4,x3,x2,x1	Zobrazený znak	Segment G F E D C B A
0	0000	0	1 0 0 0 0 0 0
1	0001	1	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

Pro programátory obvykle potřebujeme tzv. INTEL HEX

:10000000406712032509086300017F7F7F7F7F60
:00000001FF

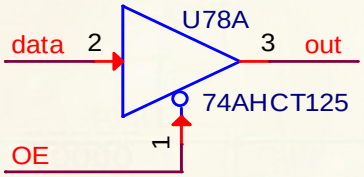
APLIKACE KOMBINAČNÍCH OBVODŮ

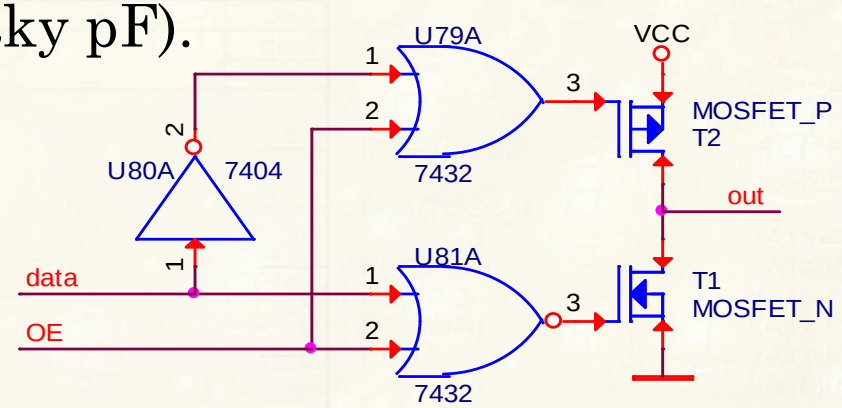
- **Aritmetické obvody** - obvody pro porovnání čísel, sčítačky, odčítačky, násobičky, děličky, ALU, obvody pro úpravu čísel (vytváření doplňků), „barelshifter“.
- **Kódovací obvody** – kódovací a dekódovací obvody, převodníky kódů.

  Například: BCD → 7 segment, Hex →7 segment, Bin → Gray, Grey → Bin, Teploměrový kód → Bin, Bin → Teploměrový kód, Grey+3 → BCD, Grey+3 → 7 segment, Prioritní kodér, atd.
- **Obvody zajišťující přenos informace** – samoopravné kódy, znáhodnění informace, atd. Přenos můžeme kontrolovat paritou, blokovou paritou v řádku i ve sloupci, kontrolním aritmetickým součtem, kontrolním polynomem např. CRC16 (LKO s LSO), kódy s větší vzdáleností (bezpečné a samoopravné) např. Hammingův kód.

OBVODY S TŘÍSTAVOVÝM VÝSTUPEM

Dosud byly popisovány obvody s dvoustavovým výstupem (0,1). Zavedeme obvody s třístavovým výstupem, které vyjma stavu 0 a 1 budou mít i třetí stav Z (vysoká impedance - oba tranzistory v koncovém stupni obvodu jsou v nevodivém stavu). Vývod **out** vůči Vcc a GND představuje vysoký odpor (stovky kΩ až MΩ) a malou kapacitu (jednotky pF).

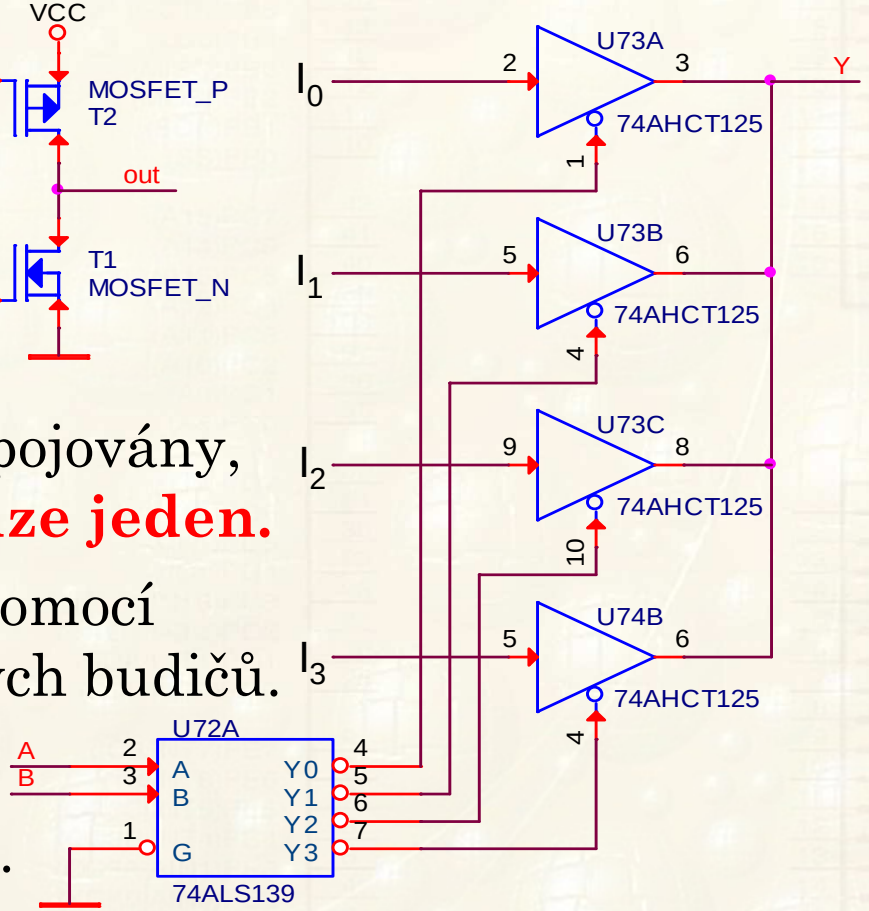




data	OE	Y
0	0	0
1	0	1
X	1	Z

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

Příklad MUX 1 ze 4 pomocí dekodéru a třístavových budičů.

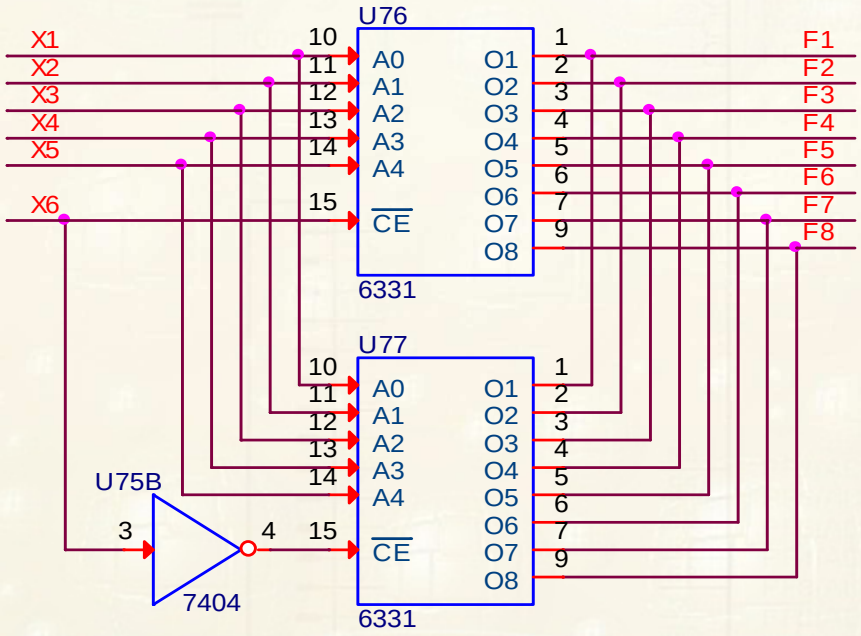


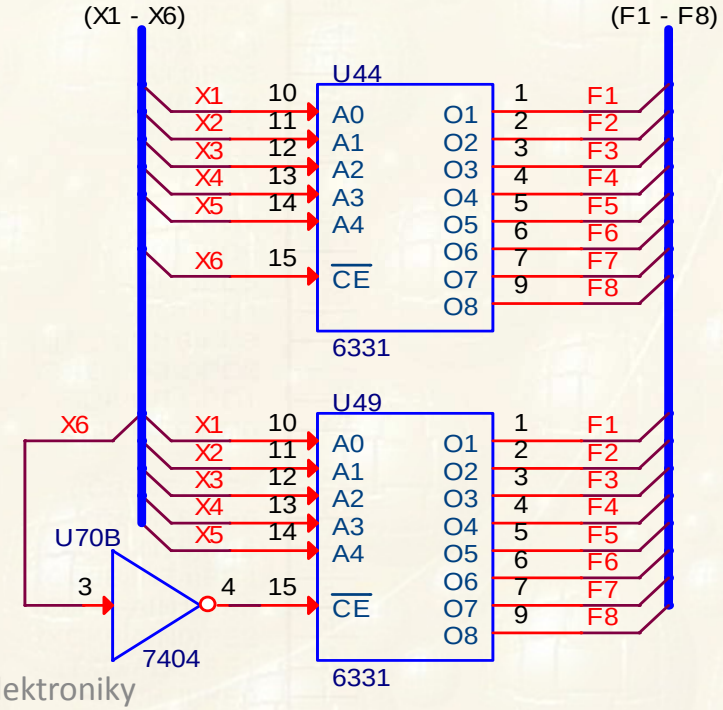
Aktivací jednoho budiče přeneseme hodnotu z jednoho vstupu I_n na výstup.

POUŽITÍ TŘÍSTAVOVÝCH OBVODŮ

Nejčastější použití obvodů s třístavovým výstupem je v systémech, kde přenášíme informace oběma směry mezi mnoha obvody po společných přenosových vodičích (**datová sběrnice**) (mikro-pocesory, přístrojové a komunikační sběrnice, propojení řady registrů, atd.). Lze s nimi realizovat i složitější kombinační obvody.

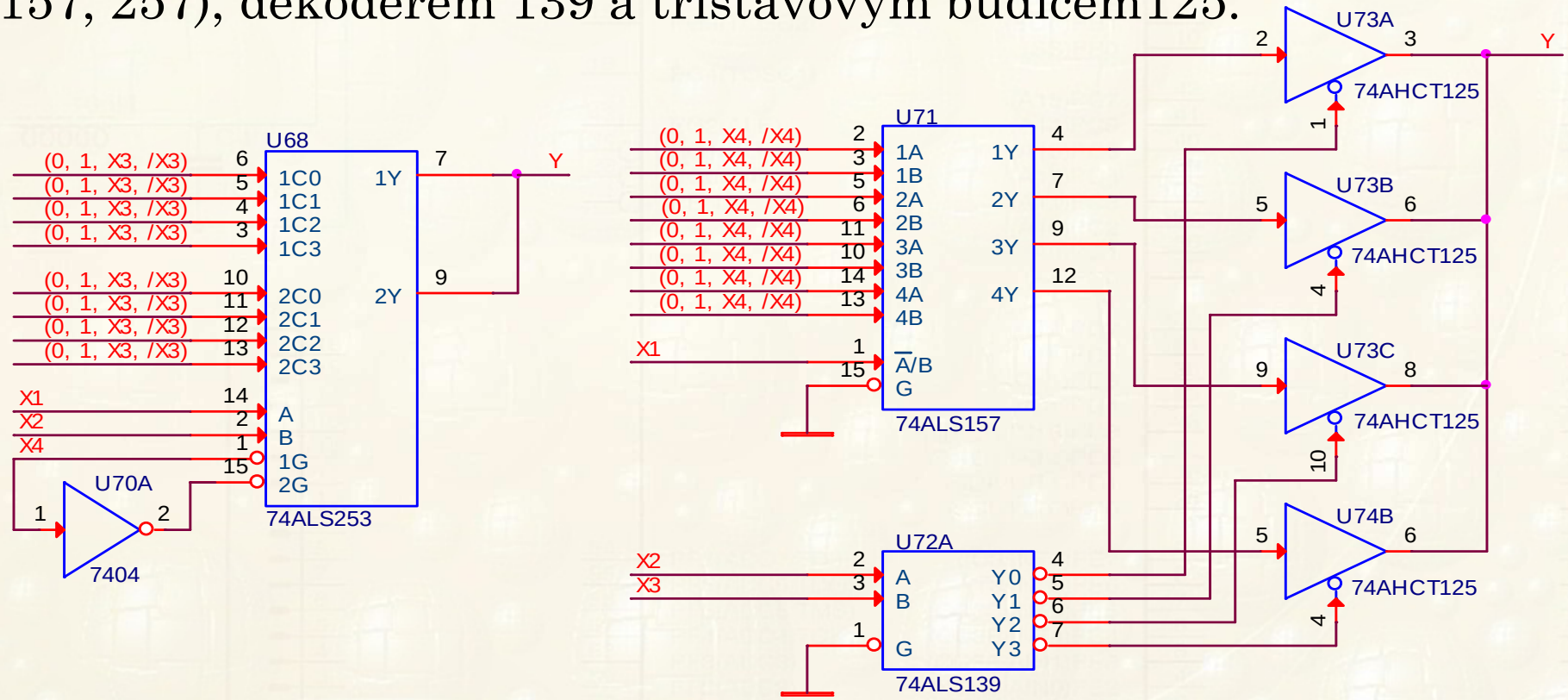
Příklad: **zvětšení kapacity** paměti pomocí dvou nebo více obvodů. K realizaci osm funkcí o 6 proměnných pomocí dvou pamětí PROM s kapacitou 32x8bitů. Pět proměnných spojíme s adresovacími vstupy, šestou na aktivační vstup paměti (řízení třístavového budiče). Každá paměť tak realizuje ½ kombinací požadovaných funkcí.





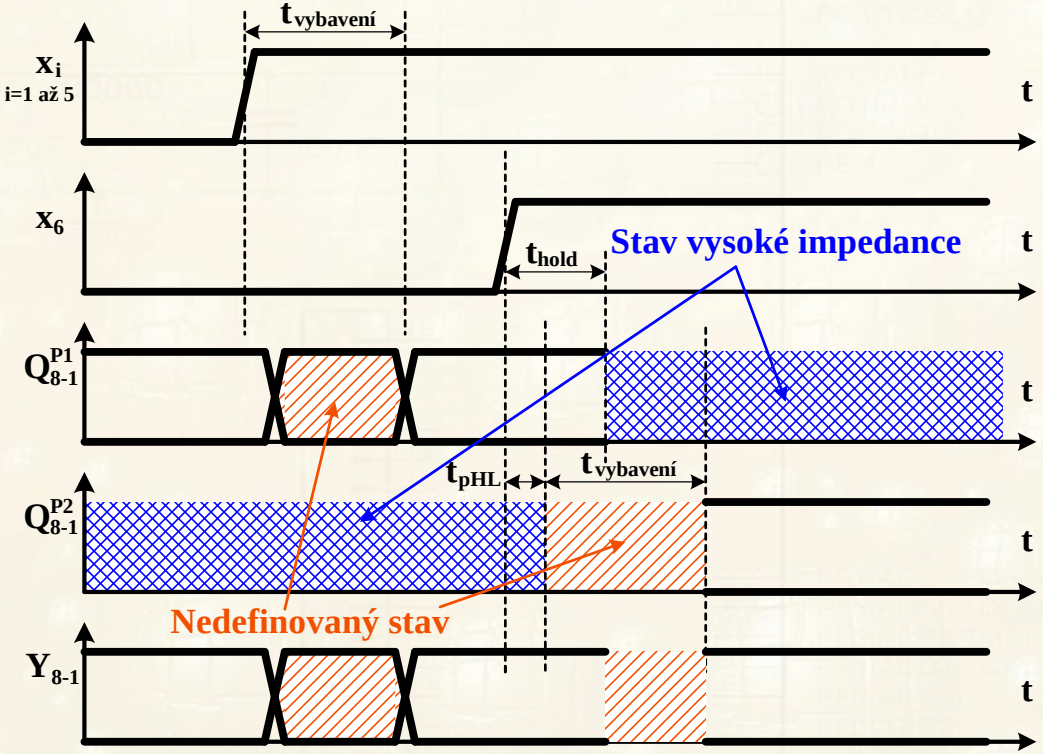
Uvedený **obvod nemá stejné vlastnosti** jako realizace obvody s dvoustavovým výstupem. Pro správnou činnost je potřeba zajistit, aby **současně nebyly aktivní obě paměti** (zkrat mezi výstupy s rozdílnou hodnotou výstupu).

Analogicky můžeme postupovat i s multiplexery s třístavovým výstupem (řada 25x). Na obrázcích jsou možné realizace logické funkce 4 proměnných s obvodem 253 a čtyř multiplexerů 1 ze 2 (157, 257), dekodérem 139 a třístavovým budičem125.



POUŽITÍ TŘÍSTAVOVÝCH OBVODŮ

Realizace LKO třístavovými obvody vykazuje odlišné vlastnosti výstupních signálů. Výstupní signály budou ovlivněny časovými parametry třístavových budičů (dobou aktivace a deaktivace budiče a vlastní tvorbou aktivačních signálů). U realizace pamětí, budou výstupy ovlivněny **dobou vybavení** (od platné adresy nebo aktivačního signálu paměti) dobou aktivace třístavového budiče (od aktivačního signálu třístavového budiče, je-li jím paměť vybavena) a vlastní tvorbou aktivačních signálů.



Obrázek popisuje situaci při změně na adresovém vodiči paměti a při přechodu na druhou paměť. Při řešení s multiplexerem je situace složitější, protože čas aktivace a deaktivace výstupu obvodu 253 je srovnatelný. Pro x_4 z 1→0, dojde ke zkratu mezi 1Y a 2Y, bude-li

$$t_{aktiv} < t_{pLH} + t_{deaktiv}$$