Návrh číslicových systémů (INC)

Jiří Matoušek, Otto Fučík

Vysoké učení technické v Brně
Fakulta informačních technologií
Božetěchova 2, 612 66 Brno

Použitá literatura

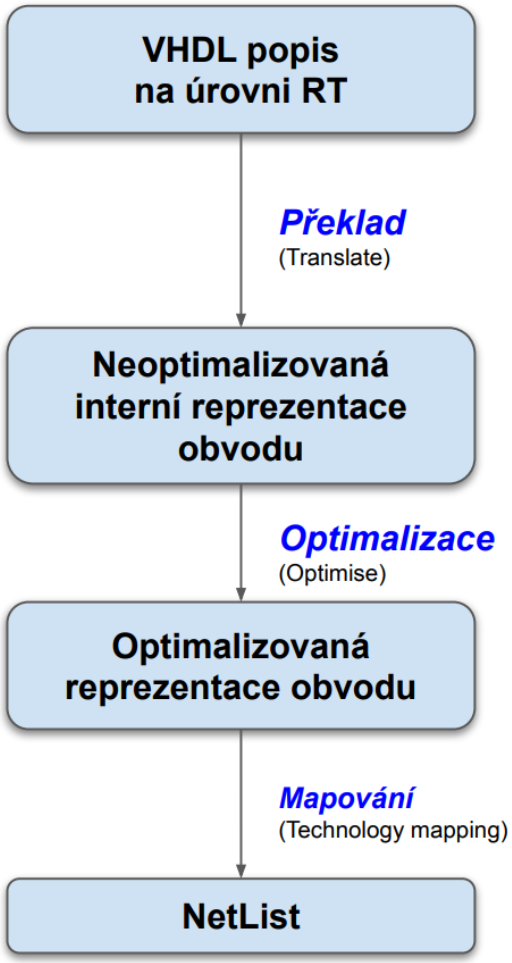
N. Frištacký, M. Kolesár, J. Kolenička a J. Hlavatý: „Logické systémy", SNTL Praha, 1986
M. Eysselt: „Logické systémy", SNTL Praha, skriptum VUT v Brně, 1985
J. F. Wakerly: „Digital Design. Principles and Practices", Prentice Hall, ISBN 0-13-769191-2, 2000
V. P. Nelson, H.T.Nagle, B.D.Carroll, J.D.Irwin: „Digital Logic Circuit Analysis & Design", ISBN 0-13-463894-8, 1995
T.L.Floyd: „Digital Fundamentals", Prentice Hall, ISBN 0-13-080850-4, 2000

Mapování logických funkcí do cílové technologie

Obsah

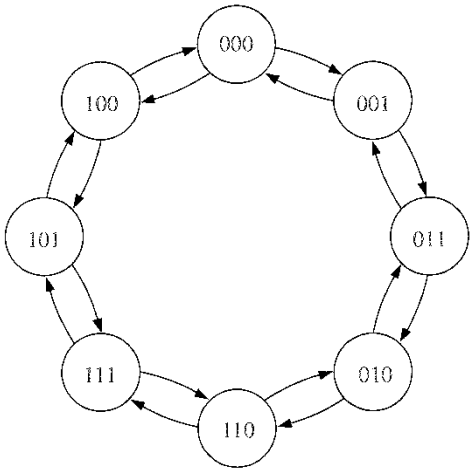
Fáze syntézy

1. Převod HDL popisu obvodu do interní reprezentace syntézního nástroje
2. Booleovské optimalizace interní reprezentace
3. **Mapování do cílové technologie**



Příklad mapování synchronního čítače

- Čítač v Grayově kódu
- Graf přechodů
  - Vstup Y=1 – přechody po směru hodinových ručiček – čítání nahoru
  - Vstup Y=0 – přechody proti směru hodinových ručiček – čítání dolů
- Tabulka přechodů a slovník přechodů
  - Implementace pomocí J-K KO

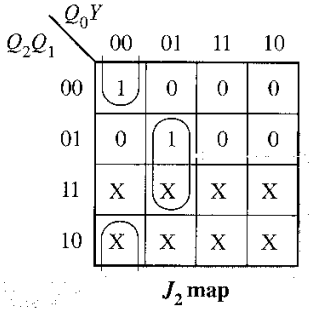


Present State			Next State Y = 0 (DOWN)			Next State Y = 1 (UP)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

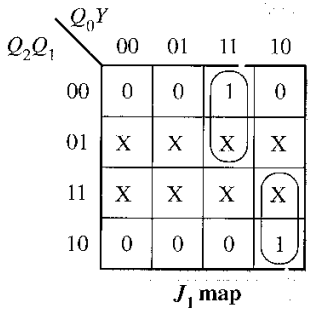
Q_i	Q_{i+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Příklad mapování synchronního čítače

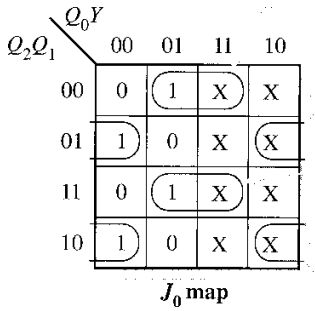
- Excitační Karnaughovy mapy



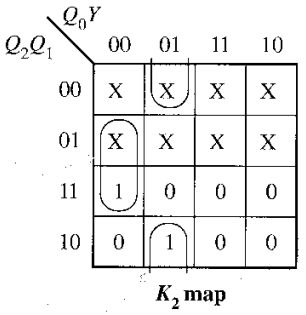
J_2 map



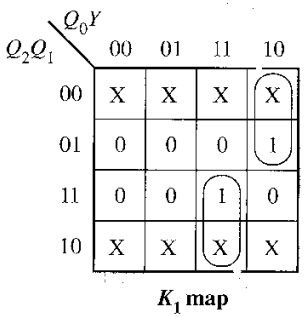
J_1 map



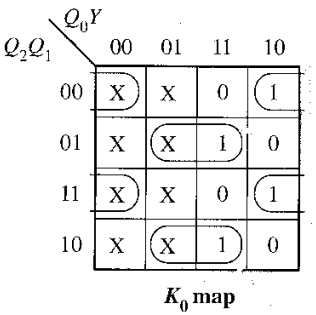
J_0 map



K_2 map



K_1 map



K_0 map

- Excitační výrazy pro budicí vstupy KO

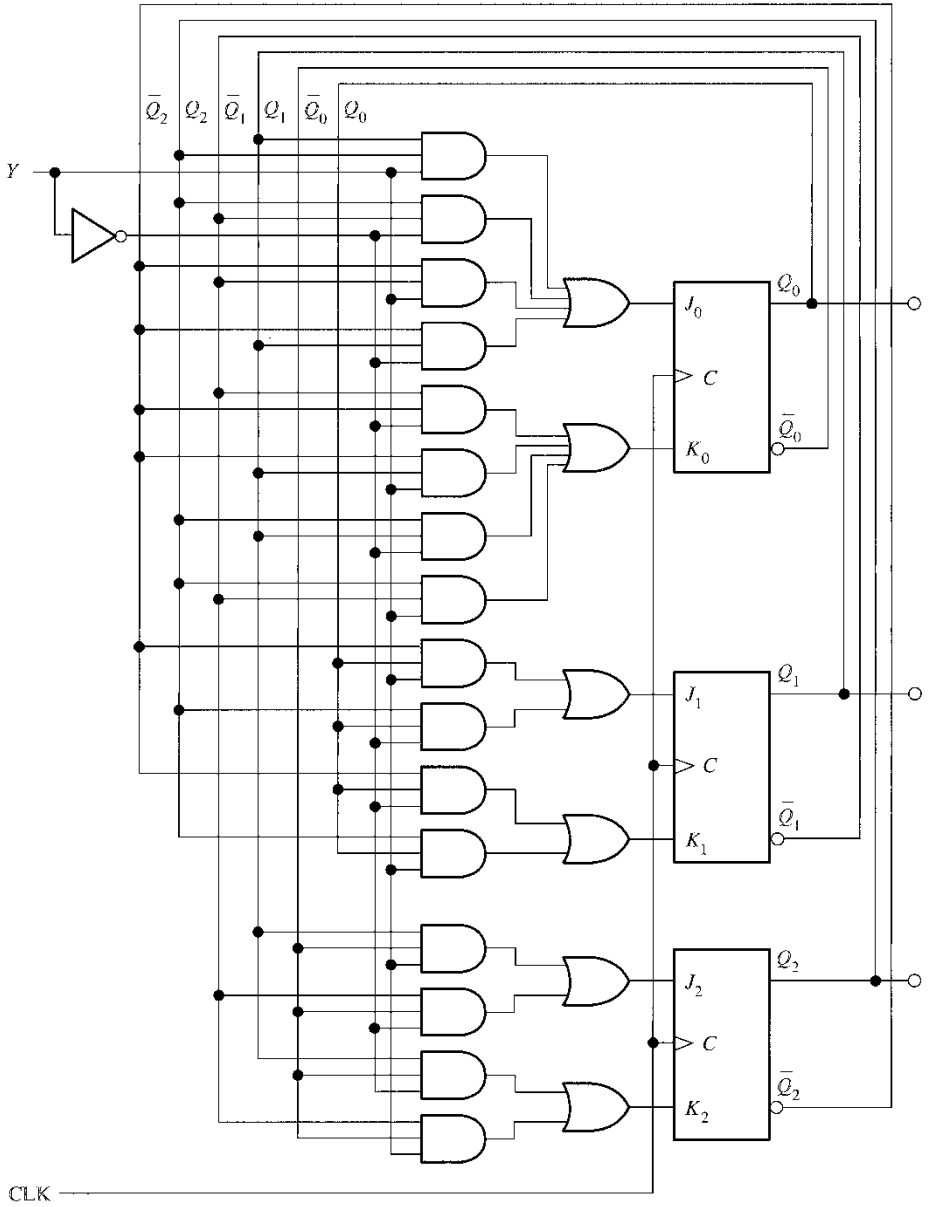
$$J_0 = Q_2Q_1Y + Q_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1\overline{Y} + \overline{Q}_2\overline{Q}_1Y \qquad K_0 = \overline{Q}_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1Y + Q_2Q_1\overline{Y} + Q_2\overline{Q}_1Y$$

$$J_1 = \overline{Q}_2Q_0Y + Q_2Q_0\overline{Y} \qquad K_1 = \overline{Q}_2Q_0\overline{Y} + Q_2Q_0Y$$

$$J_2 = Q_1\overline{Q}_0Y + \overline{Q}_1\overline{Q}_0\overline{Y} \qquad K_2 = Q_1\overline{Q}_0\overline{Y} + \overline{Q}_1\overline{Q}_0Y$$

Příklad mapování synchronního čítače

- Logické schéma
  - Y=0 – čítej dolů
  - Y=1 – čítej nahoru



Obsah

Shannonův teorém - rozklad logické funkce

- ÚNDF

$$f(x_1, x_2, \ldots, x_n) = [\bar{x}_k \cdot f(x_1, x_{k-1}, 0, x_{k+1}, \ldots, x_n)] + [x_k \cdot f(x_1, x_{k-1}, 1, x_{k+1}, \ldots, x_n)]$$

- ÚNKF

$$f(x_1, x_2, \ldots, x_n) = [\bar{x}_k + f(x_1, x_{k-1}, 1, x_{k+1}, \ldots, x_n] \cdot [x_k + f(x_1, x_{k-1}, 0, x_{k+1}, \ldots, x_n)]$$

- Např. podle proměnné x_1
  - ÚNDF $f(x_1, x_2, \ldots, x_n) = [\underline{x_1} \cdot f(0, x_2, \ldots, x_n)] + [x_1 \cdot f(1, x_2, \ldots, x_n)]$
  - ÚNKF $f(x_1, x_2, \ldots, x_n) = [x_1 + f(1, x_2, \ldots, x_n)] \cdot [x_1 + f(0, x_2, \ldots, x_n)]$
- Platí dualita funkcí

$$f(x_1, x_2, \ldots, x_n, 0, 1, +, \cdot) = f(x_1, x_2, \ldots, x_n, 1, 0, \cdot, +)$$

- **Důsledek**
  - Každou log. funkci libovolného počtu proměnných lze realizovat pomocí základních log. operací AND, OR a NOT dvou proměnných - superpozice logických funkcí

Shannonův teorém - příklad

- Rozklad funkce dle proměnné x_1

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

$$f(x_1 = 1) = x_2 \cdot \bar{x}_3$$
$$f(x_1 = 0) = x_2 + \bar{x}_2 \cdot x_3$$

- Rozklad funkce dle proměnné x_2

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

$$f(x_2 = 1) = x_1 \cdot \bar{x}_3 + \bar{x}_1$$
$$f(x_2 = 0) = \bar{x}_1 \cdot x_3$$

- Rozklad funkce dle proměnné x_3

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

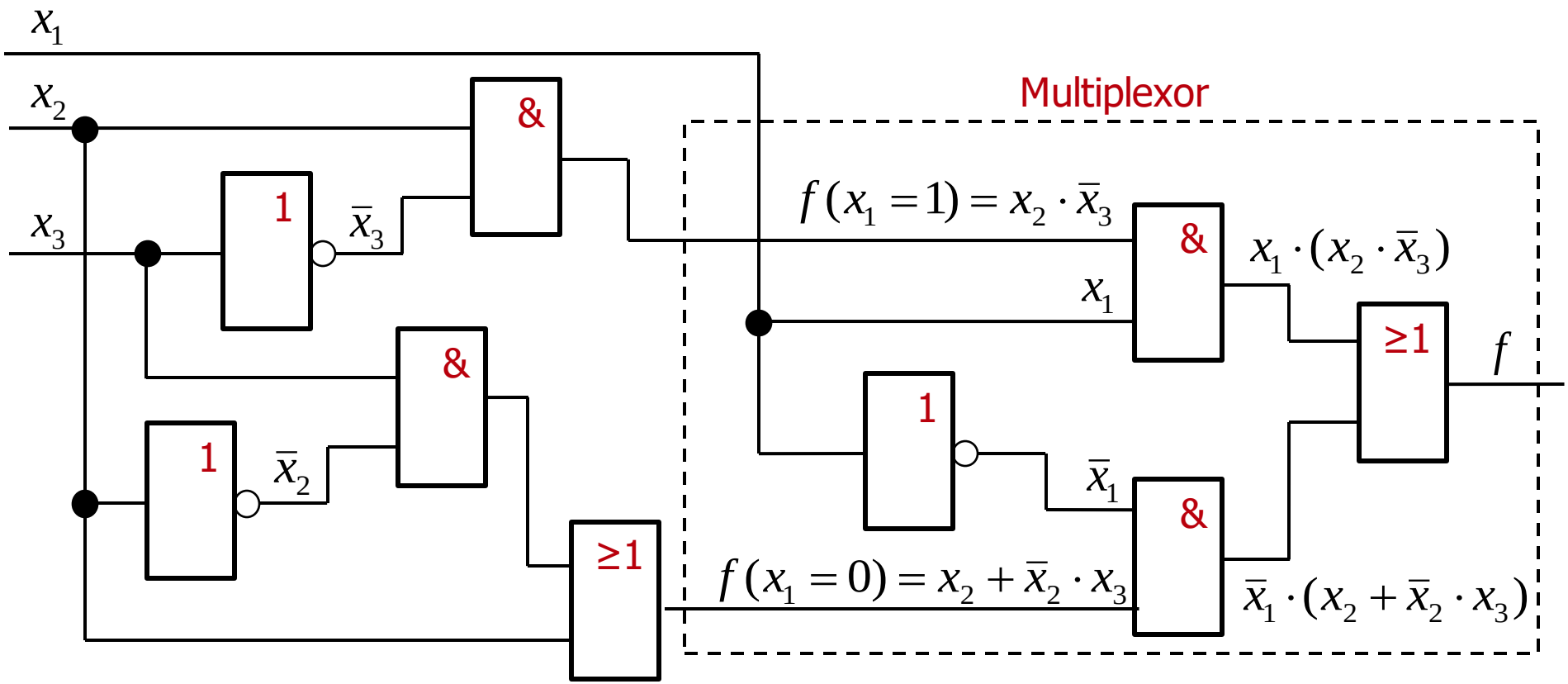
$$f(x_3 = 1) = \bar{x}_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2$$
$$f(x_3 = 0) = x_1 \cdot x_2 + \bar{x}_1 \cdot x_2$$

Shannonův teorém - příklad

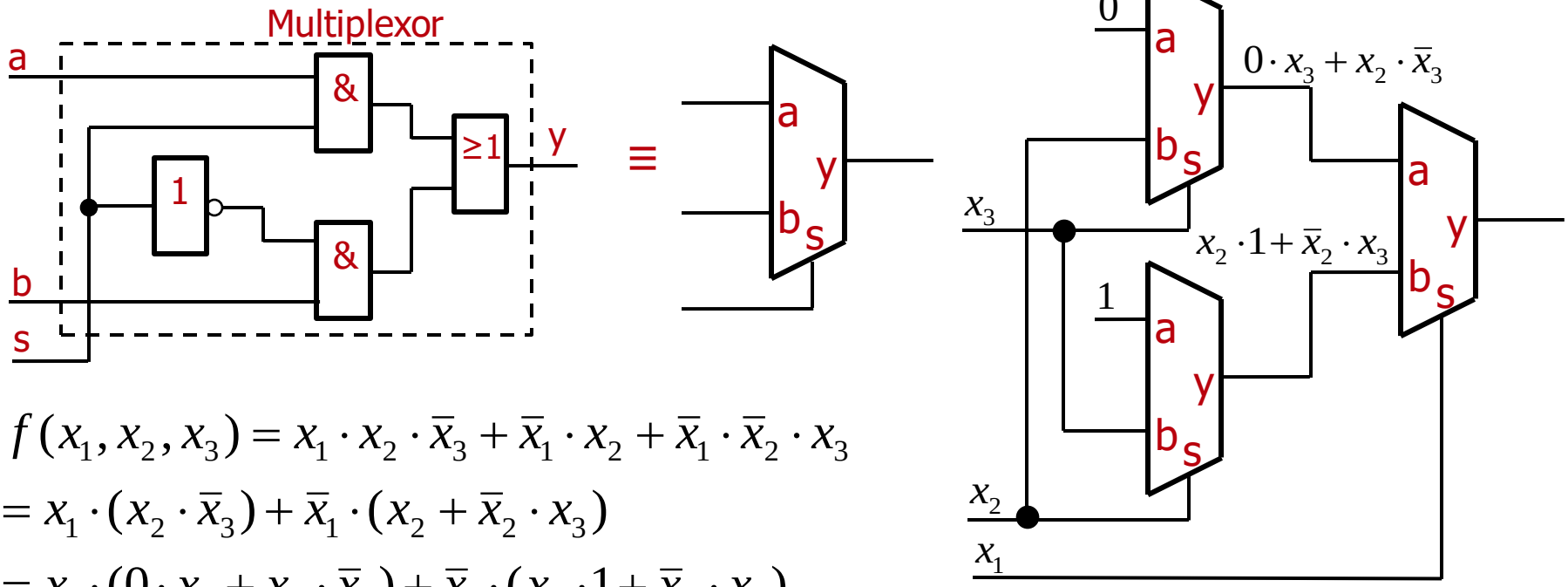
- Rozklad funkce dle proměnné x_1

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

$$f(x_1 = 1) = x_2 \cdot \bar{x}_3$$
$$f(x_1 = 0) = x_2 + \bar{x}_2 \cdot x_3$$



- **Důsledek**
  - Každá log. funkce může být implementována pouze pomocí multiplexorů
  - Poznámka: zvyšuje se zpoždění obvodu
- Příklad



$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot \bar{x}_3 + \bar{x}_1 \cdot x_2 + \bar{x}_1 \cdot \bar{x}_2 \cdot x_3$$
$$= x_1 \cdot (x_2 \cdot \bar{x}_3) + \bar{x}_1 \cdot (x_2 + \bar{x}_2 \cdot x_3)$$
$$= x_1 \cdot (0 \cdot x_3 + x_2 \cdot \bar{x}_3) + \bar{x}_1 \cdot (x_2 \cdot 1 + \bar{x}_2 \cdot x_3)$$

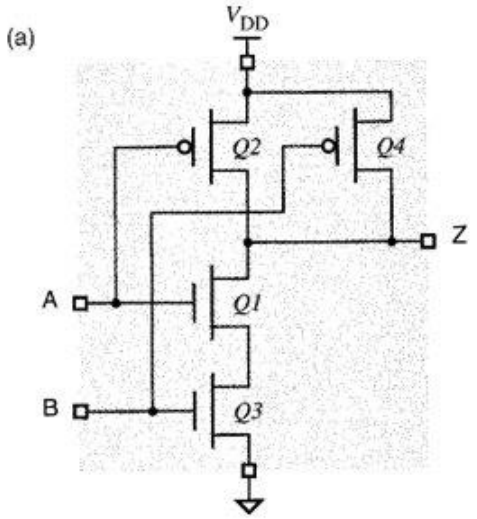
Shefferova algebra

- Využívá pouze log. funkci NAND – (a·b)', $a \uparrow b$, $\overline{a \cdot b}$
  - Lze ukázat, že pomocí funkce NAND lze realizovat základní log. členy a tedy i veškeré logické funkce
- Vede na efektivní realizaci v technologii CMOS

Hradlo NAND

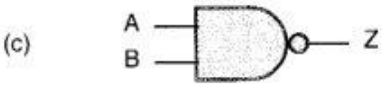
Hradlo AND

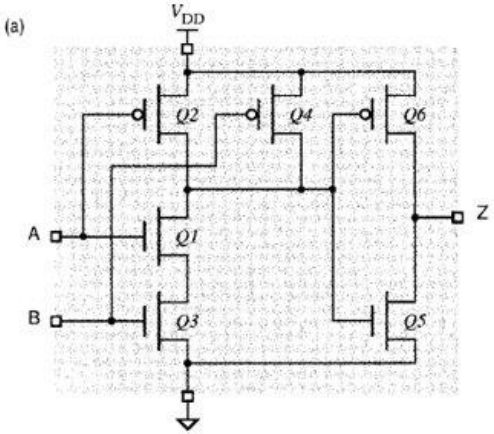
(a) Schéma, (b) Pravdivostní tabulka, (c) Symbol



(b)

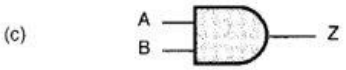
A	B	Q1	Q2	Q3	Q4	Z
L	L	off	on	off	on	H
L	H	off	on	on	off	H
H	L	on	off	off	on	H
H	H	on	off	on	off	L





(b)

A	B	Q1	Q2	Q3	Q4	Q5	Q6	Z
L	L	off	on	off	on	on	off	L
L	H	off	on	on	off	on	off	L
H	L	on	off	off	on	on	off	L
H	H	on	off	on	off	off	on	H



Shefferova algebra - pravidla

- Platí zákon komutativní $(a \cdot b)' = (b \cdot a)'$
- Neplatí zákon asociativní $((a \cdot b)' \cdot c)' \neq (a \cdot (b \cdot c)')'$

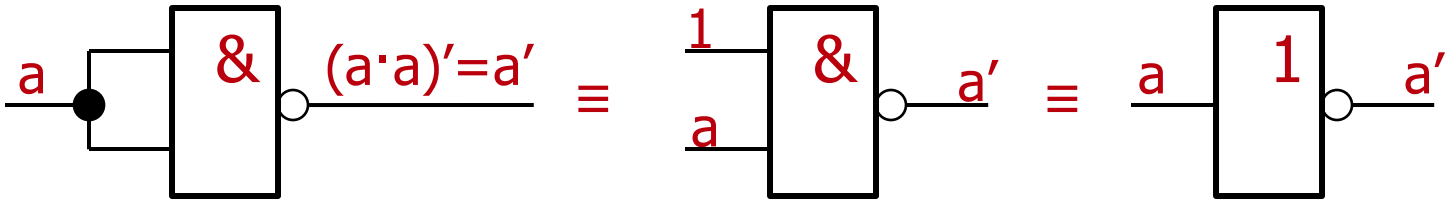
$(a \cdot a)' = a'$

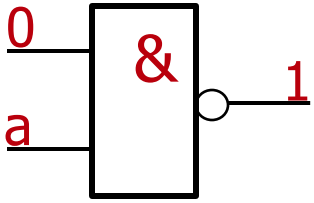
$(a \cdot 0)' = 1$

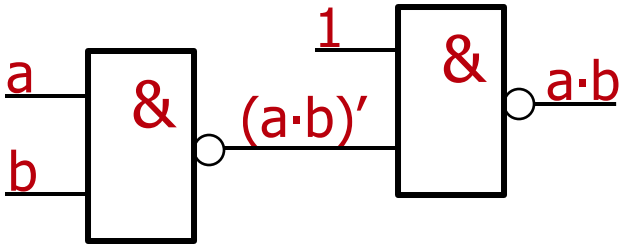
$(a \cdot 1)' = a'$

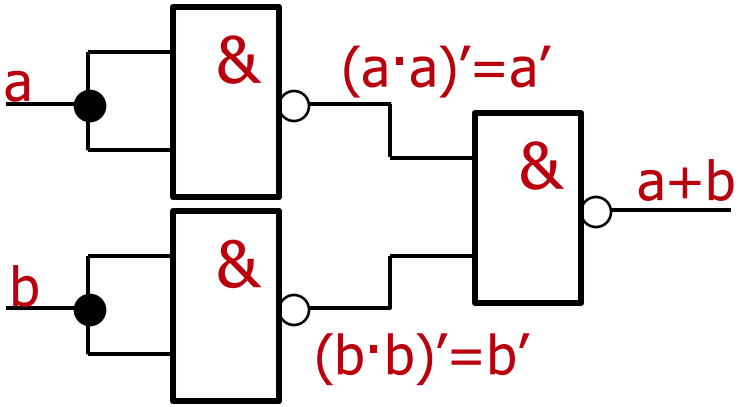
$(a \cdot b)' \cdot 1 = ((a \cdot b)')' = a \cdot b$

$((a \cdot a)' \cdot (b \cdot b)')' = (a' \cdot b')' = a + b$









Převod Booleovy algebry na Shefferovu algebru

- Využití teorémů:
  - Involuce (dvojí negace)
  - de Morganovy zákony
- Pro implementaci výrazu stačí pouze členy NAND
- Příklad

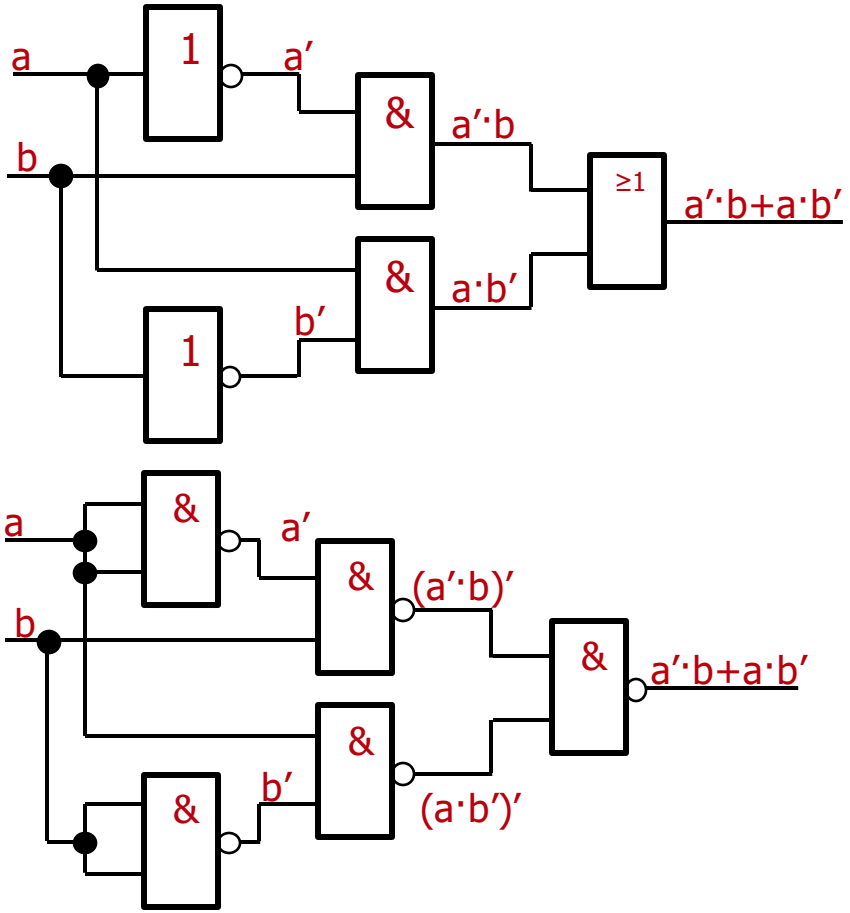
$$\overline{\overline{a}} = a$$

$$\overline{a+b} = \overline{a} \cdot \overline{b}$$

$$\overline{a} \cdot b + a \cdot \overline{b}$$

$$= \overline{\overline{\overline{a} \cdot b + a \cdot \overline{b}}}$$

$$= \overline{\overline{\overline{a} \cdot b} \cdot \overline{a \cdot \overline{b}}}$$



Piercova algebra

- Využívá pouze log. funkci NOR – (a+b)', $a \downarrow b$
  - Lze ukázat, že pomocí funkce NOR lze realizovat základní log. členy a tedy i veškeré logické funkce
- Vede na efektivní realizaci v technologii CMOS
  - Hradlo NOR vs. hradlo OR viz hradlo NAND vs. hradlo AND

Piercova algebra - pravidla

- Platí zákon komutativní $(a+b)'=(b+a)'$
- Neplatí zákon asociativní $((a+b)'+c)'\neq(a+(b+c)')'$

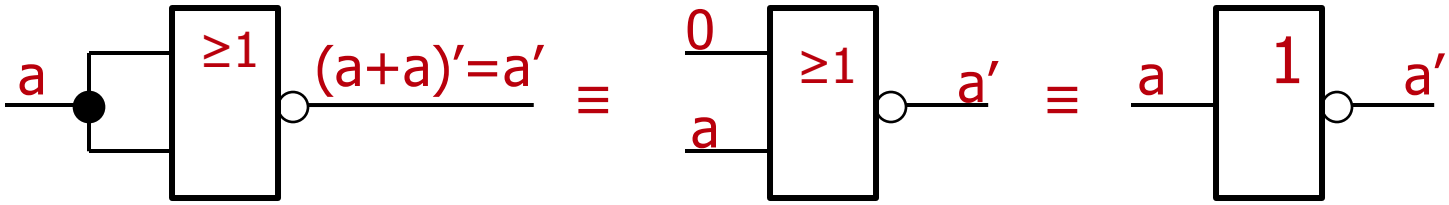
$(a+a)'=a'$

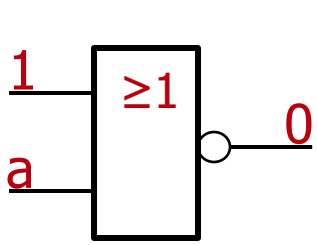
$(a+0)'=a'$

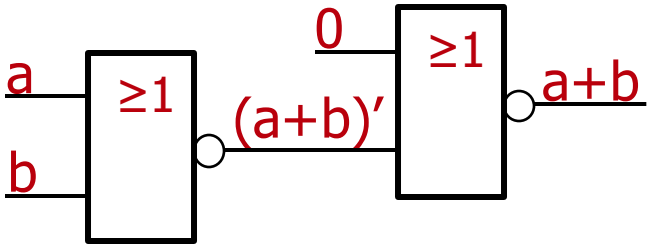
$(a+1)'=0$

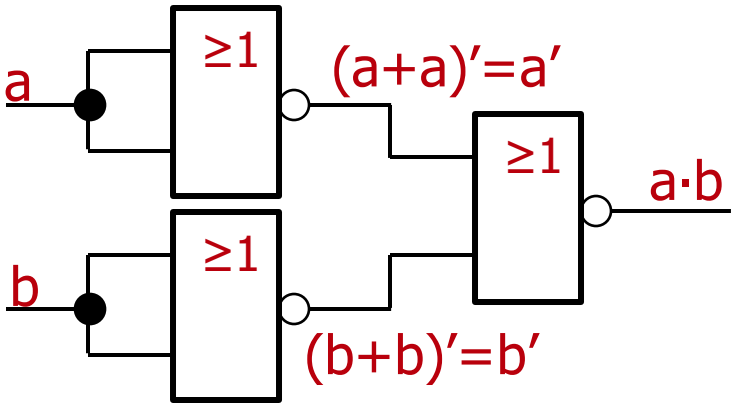
$(a+b)'+0=((a+b)')'=a+b$

$((a+a)'+(b+b)')'=(a'+b')'=a\cdot b$









Převod Booleovy algebry na Piercovu algebru

- Využití teorémů:
  - Involuce (dvojí negace)
  - de Morganovy zákony
- Pro implementaci výrazu stačí pouze člen NOR
- Příklad

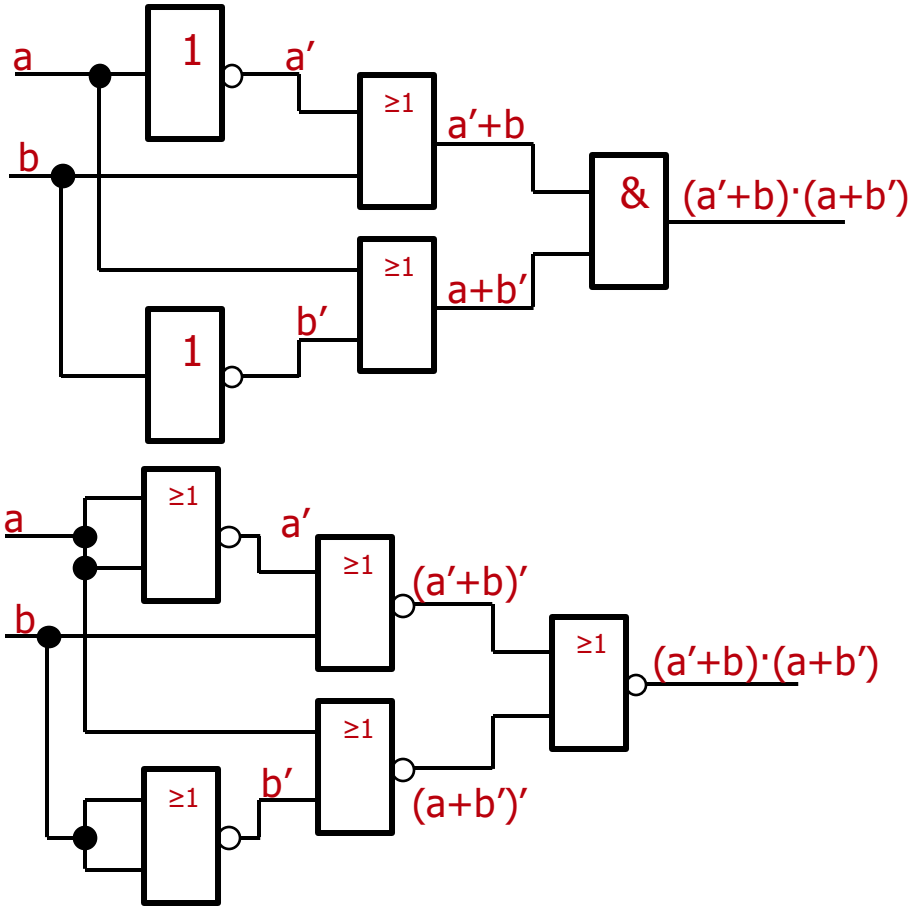
$$\overline{\overline{a}} = a$$

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

$$(\overline{a} + b) \cdot (a + \overline{b})$$

$$= \overline{\overline{(\overline{a} + b) \cdot (a + \overline{b})}}$$

$$= \overline{\overline{\overline{a} + b} + \overline{a + \overline{b}}}$$

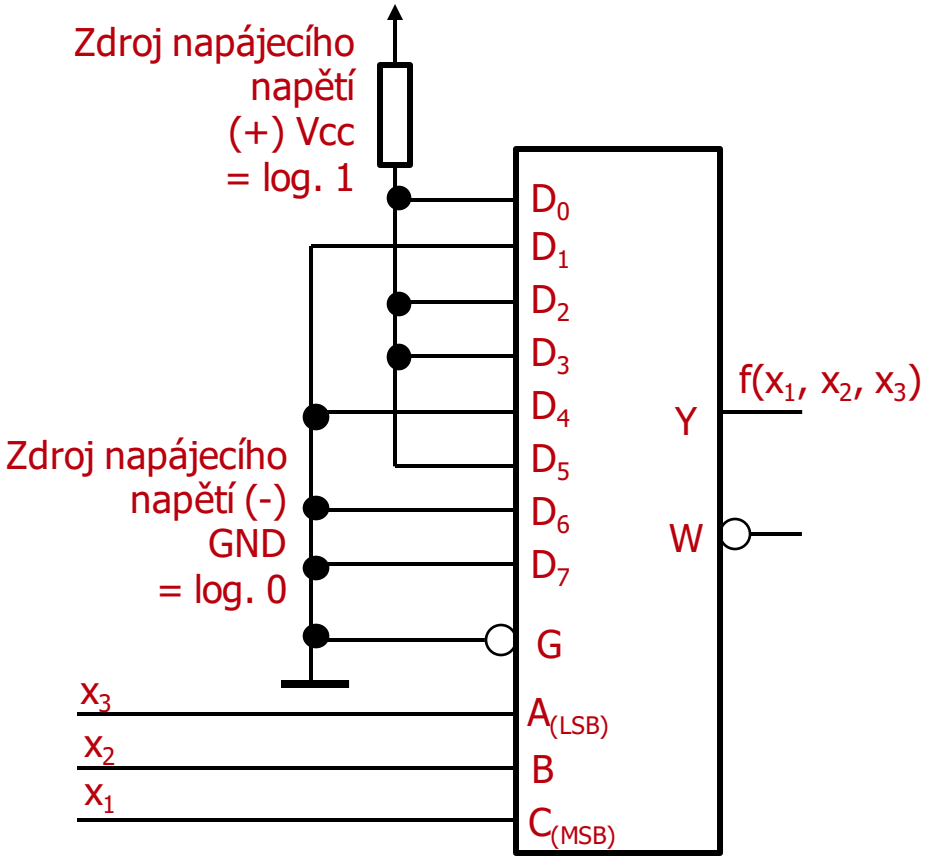


Obsah

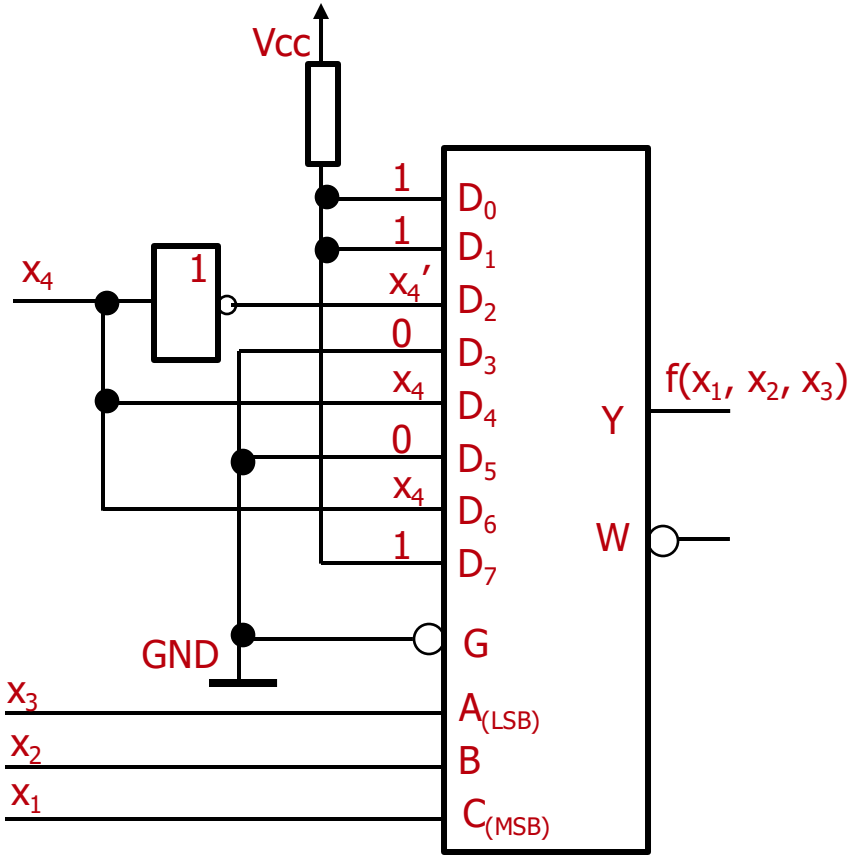
Implementace log. funkce pomocí multiplexoru

- Příklad: Implementace funkce $f(x_1, x_2, x_3) = \sum m(0,2,3,5)$
  - Pravdivostní tabulka
  - Realizace multiplexorem 8-1

Vstupy		Výstup
C, B, A	Data	Y
x_1, x_2, x_3		$f(x_1, x_2, x_3)$
000	$D_0=1$	1
001	$D_1=0$	0
010	$D_2=1$	1
011	$D_3=1$	1
100	$D_4=0$	0
101	$D_5=1$	1
110	$D_6=0$	0
111	$D_7=0$	0



- Příklad
  - Implementace funkce $f(x_1, x_2, x_3, x_4) = \sum m(0,1,2,3,4,9,13,14,15)$
  - Realizace multiplexorem 8-1 a invertorem



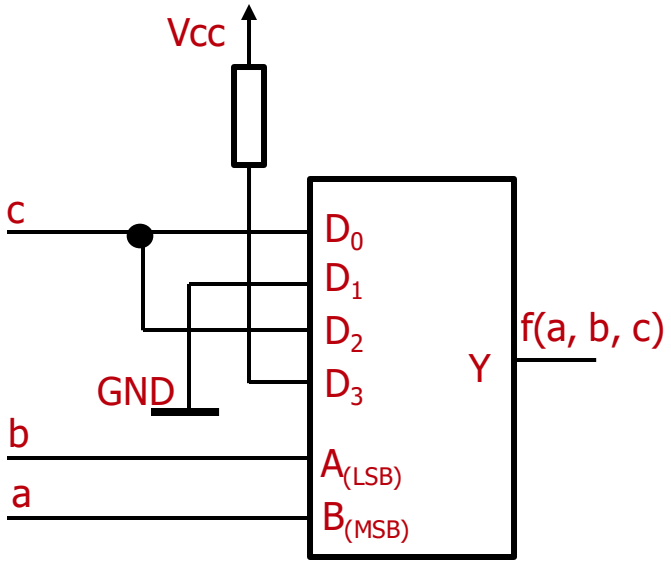
	C, B, A	x_4	f	=	Y
	x_1, x_2, x_3				$f(x_1, x_2, x_3, x_4)$
0 1	000 000	0 1	1 1	1	$D_0=1$
2 3	001 001	0 1	1 1	1	$D_1=1$
4 5	010 010	0 1	1 0	x_4'	$D_2=x_4'$
6 7	011 011	0 1	0 0	0	$D_3=0$
8 9	100 100	0 1	0 1	x_4	$D_4=x_4$
10 11	101 101	0 1	0 0	0	$D_5=0$
12 13	110 110	0 1	0 1	x_4	$D_6=x_4$
14 15	111 111	0 1	1 1	1	$D_7=1$

Implementace log. funkce pomocí multiplexoru

- Příklad – verze 1
  - Implementace funkce
  - Pravdivostní tabulka
  - Realizace multiplexorem 4-1

$$f(a,b,c) = a \cdot b + \bar{b} \cdot c$$
$$= a \cdot b \cdot 1 + 1 \cdot \bar{b} \cdot c$$
$$= a \cdot b \cdot (c + \bar{c}) + (a + \bar{a}) \cdot \bar{b} \cdot c$$
$$= a \cdot b \cdot c + a \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot c + \bar{a} \cdot \bar{b} \cdot c$$

Vstupy		**Výstup**
B, A	**Data**	**Y**
a, b		**f(a, b, c)**
00	D_0=c	c
01	D_1=0	0
10	D_2=c	c
11	D_3=1	1

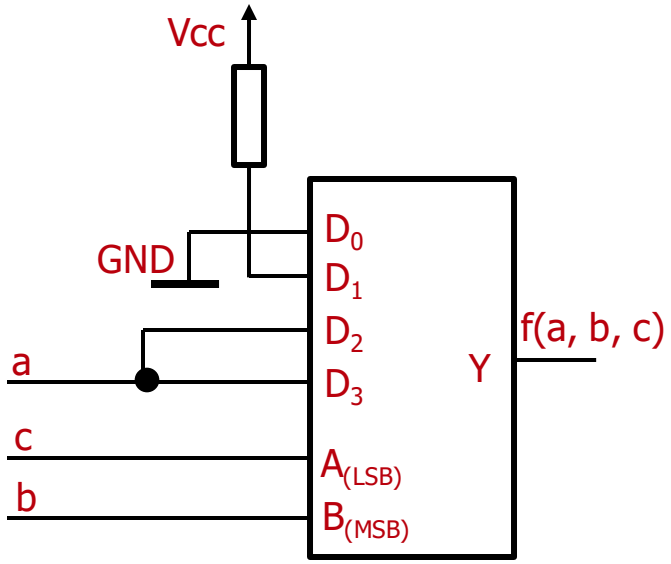


Implementace log. funkce pomocí multiplexoru

- Příklad – verze 2
  - Implementace funkce
  - Pravdivostní tabulka
  - Realizace multiplexorem 4-1

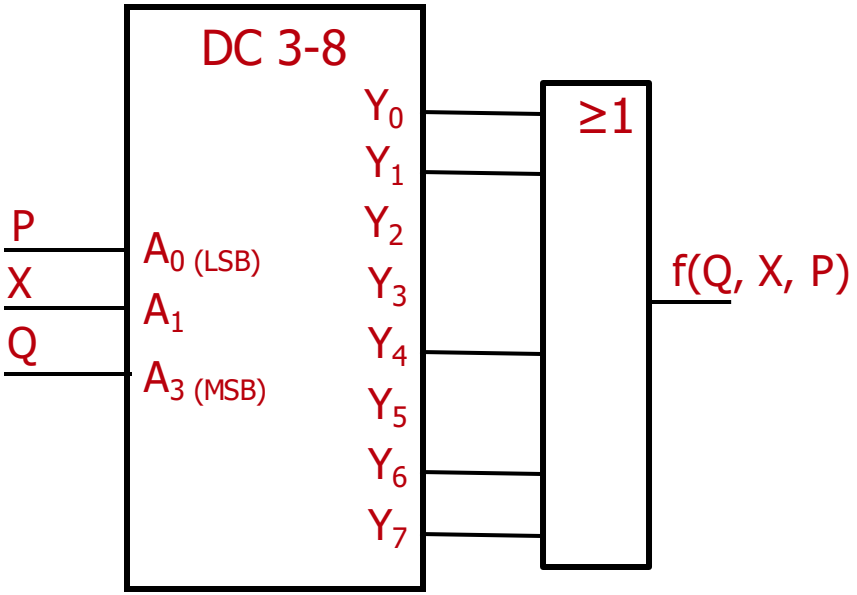
$$f(a,b,c) = a \cdot b + \overline{b} \cdot c$$
$$= a \cdot b \cdot 1 + 1 \cdot \overline{b} \cdot c$$
$$= a \cdot b \cdot (c + \overline{c}) + (a + \overline{a}) \cdot \overline{b} \cdot c$$
$$= a \cdot b \cdot c + a \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot c + \overline{a} \cdot \overline{b} \cdot c$$

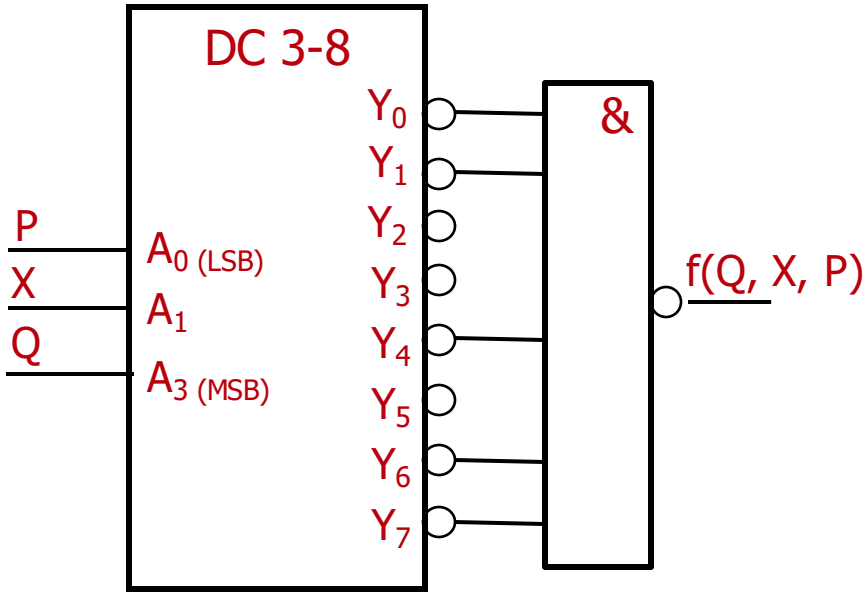
Vstupy		**Výstup**
B, A	**Data**	**Y**
b, c		**f(a, b, c)**
00	D_0=0	0
01	D_1=1	1
10	D_2=a	a
11	D_3=a	a



Implementace log. funkce pomocí dekodéru

- Implementace funkce $f(Q, X, P) = \sum m(0,1,4,6,7)$
  - S výstupem aktivním v jedničce pomocí tříbitového dekodéru s výstupy aktivními v log. 1 a log. členu OR
  - S výstupem aktivním v nule pomocí tříbitového dekodéru s výstupy aktivními v nule a log. členu NAND (DeMorganův zákon)

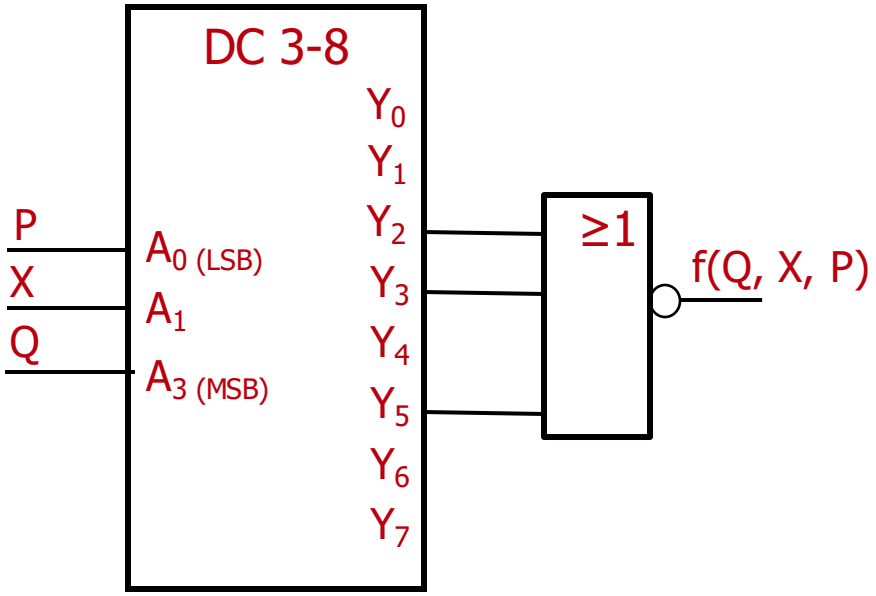


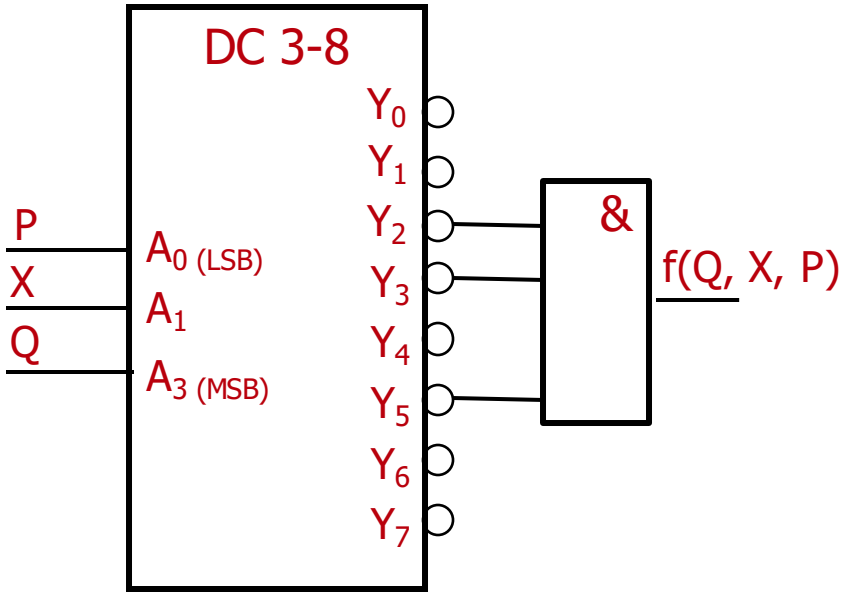


Implementace log. funkce pomocí dekodéru

- Implementace funkce $f(Q, X, P) = \sum m(0,1,4,6,7)$
  - S výstupem aktivním v jedničce pomocí tříbitového dekodéru s výstupy aktivními v nule a log. členu NOR
  - S výstupem aktivním v nule pomocí tříbitového dekodéru s výstupy aktivními v nule a log. členu AND (de Morganův zákon)

$$f(Q, X, P) = \sum m(0,1,4,6,7) = \prod M(2,3,5)$$



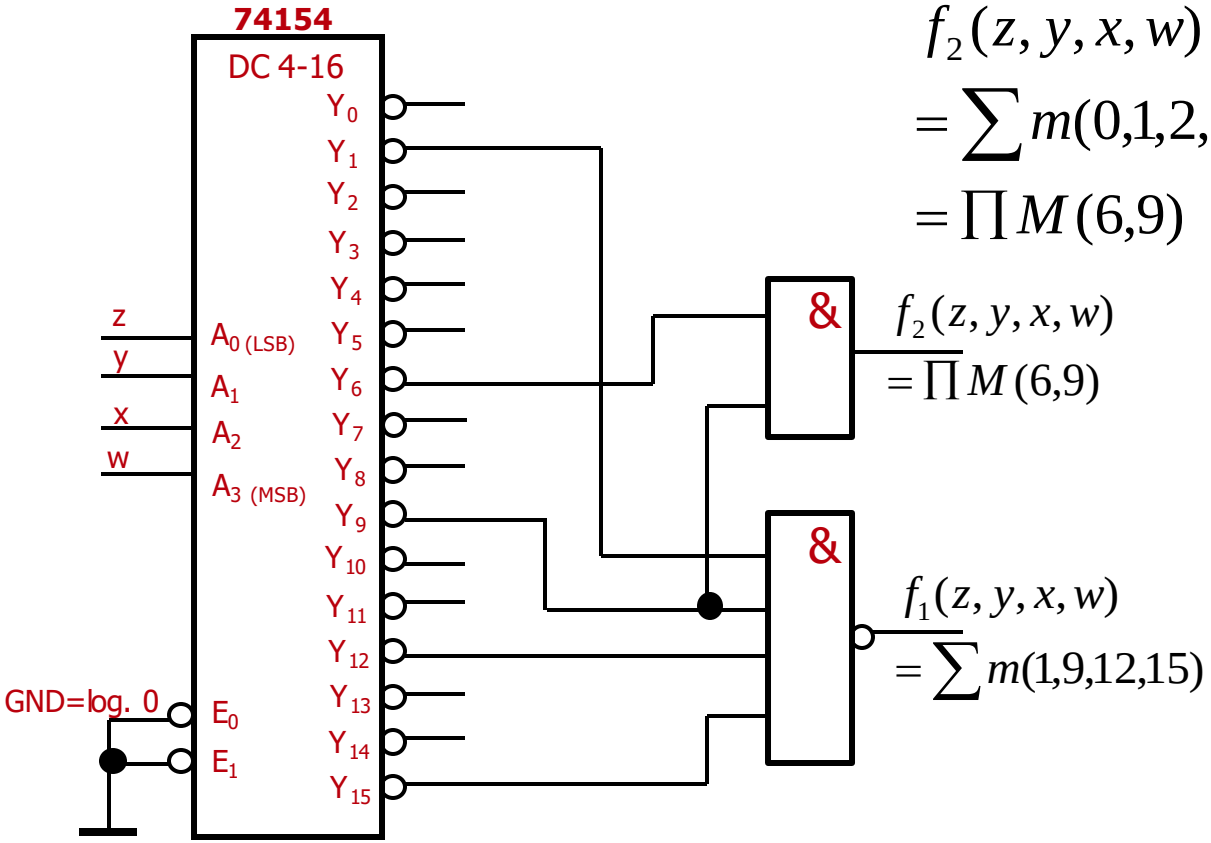


Implementace log. funkce pomocí dekodéru

- Příklad
  - Implementace funkcí pomocí dekodéru DC 4-16 s povolovacími vstupy aktivními v nule

$$f_1(z,y,x,w) = \sum m(1,9,12,15)$$

$$f_2(z,y,x,w) = \sum m(0,1,2,3,4,5,7,8,10,11,13,14) = \prod M(6,9)$$



Majoritní funkce

- Definice
  - n≥3…počet logických proměnných x_0, …, x_n
  - Všem logickým proměnným je přiřazena váha $w_i=1$
  - Prahová funkce nabývá hodnoty 1, pokud alespoň nadpoloviční většina vstupních proměnných je rovna 1
  - = symetrická prahová funkce lichého řádu s prahem $T = \frac{n+1}{2}$

$$M_n = 1 \Leftrightarrow \sum_{i=1}^{n} x_i \geq \frac{n+1}{2}$$

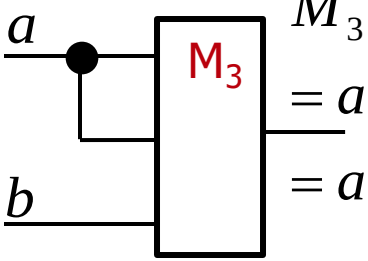
  - Řád majority - počet proměnných
- Příklad
  - $w_a=1$, $w_b=1$, $w_c=1$
  - N=1+1+1=3
  - T=2

$$M_3(a,b,c)$$
$$= \bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c} + a \cdot b \cdot c$$
$$= a \cdot b + a \cdot c + b \cdot c$$

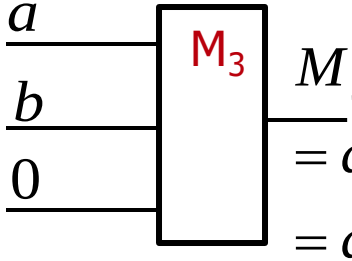
a	b	c	Σx_i	$M_3=1 \Leftrightarrow \Sigma(w_i \cdot x_i) \geq 2$
0	0	0	1·0+1·0+1·0=0	0
0	0	1	1·0+1·0+1·1=1	0
0	1	0	1·0+1·1+1·0=1	0
0	1	1	1·0+1·1+1·1=2	1
1	0	0	1·1+1·0+1·0=1	0
1	0	1	1·1+1·0+1·1=2	1
1	1	0	1·1+1·1+1·0=2	1
1	1	1	1·1+1·1+1·1=3	1

Implementace log. funkce pomocí majority

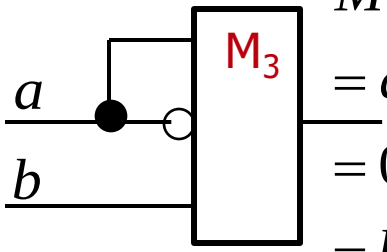
- Příklady



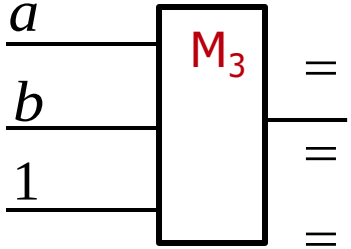
$$M_3(a,a,b) = a\cdot a + a\cdot b + a\cdot b = a + a\cdot b = a$$



$$M_3(a,b,0) = a\cdot b + a\cdot 0 + b\cdot 0 = a\cdot b$$



$$M_3(a,\bar{a},b) = a\cdot\bar{a} + a\cdot b + \bar{a}\cdot b = 0 + b\cdot(a+\bar{a}) = b\cdot 1 = b$$



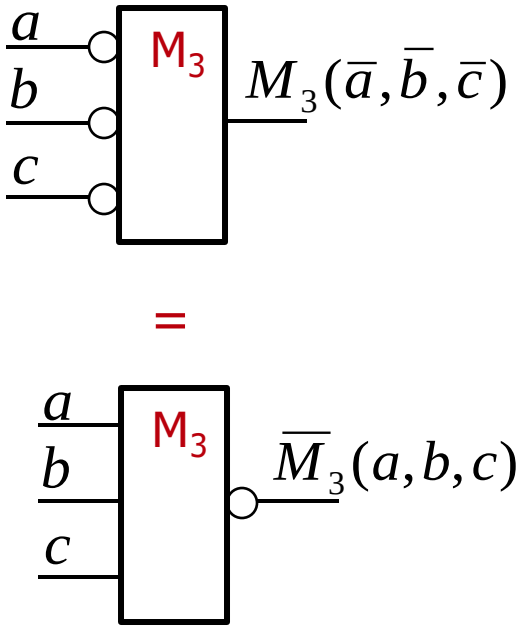
$$M_3(a,b,1) = a\cdot b + a\cdot 1 + b\cdot 1 = a\cdot b + a + b = a\cdot(b+1) + b = a + b$$

Implementace log. funkce pomocí majority

- Příklad

$$M_3(a,b,c) = \bar{a}\cdot b\cdot c + a\cdot\bar{b}\cdot c + a\cdot b\cdot\bar{c} + a\cdot b\cdot c$$

$$M_3(\bar{a},\bar{b},\bar{c}) = a\cdot\bar{b}\cdot\bar{c} + \bar{a}\cdot b\cdot\bar{c} + \bar{a}\cdot\bar{b}\cdot c + \bar{a}\cdot\bar{b}\cdot\bar{c} = \overline{M}_3(a,b,c)$$



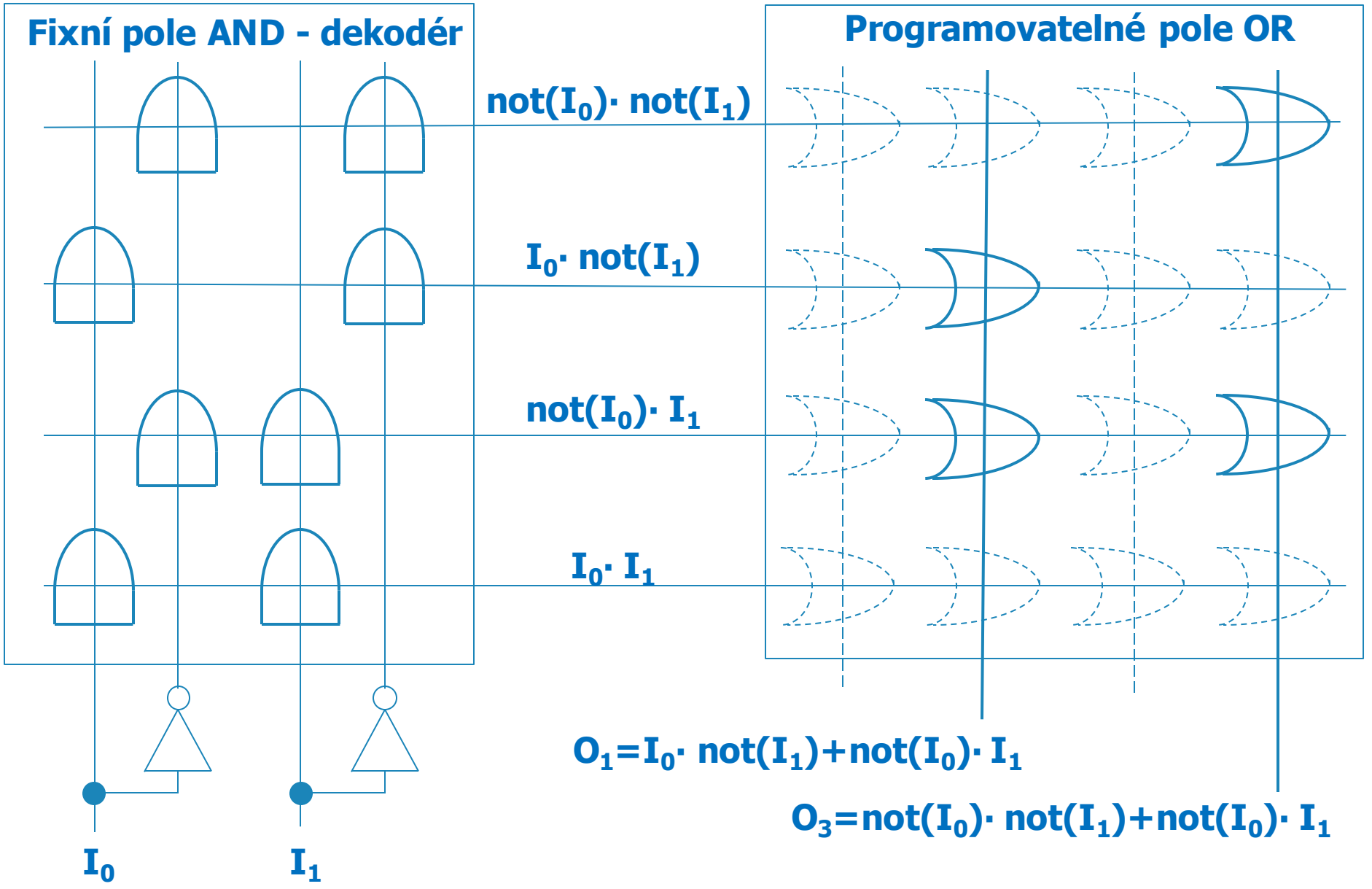
$$\sum m(3,5,6,7) = \prod M(0,1,2,4)$$

a	b	c	Σx_i	$M_3=1 \Leftrightarrow \Sigma(w_i\cdot x_i)\geq 2$
0	0	0	1·0+1·0+1·0=0	0
0	0	1	1·0+1·0+1·1=1	0
0	1	0	1·0+1·1+1·0=1	0
0	1	1	1·0+1·1+1·1=2	1
1	0	0	1·1+1·0+1·0=1	0
1	0	1	1·1+1·0+1·1=2	1
1	1	0	1·1+1·1+1·0=2	1
1	1	1	1·1+1·1+1·1=3	1

Obsah

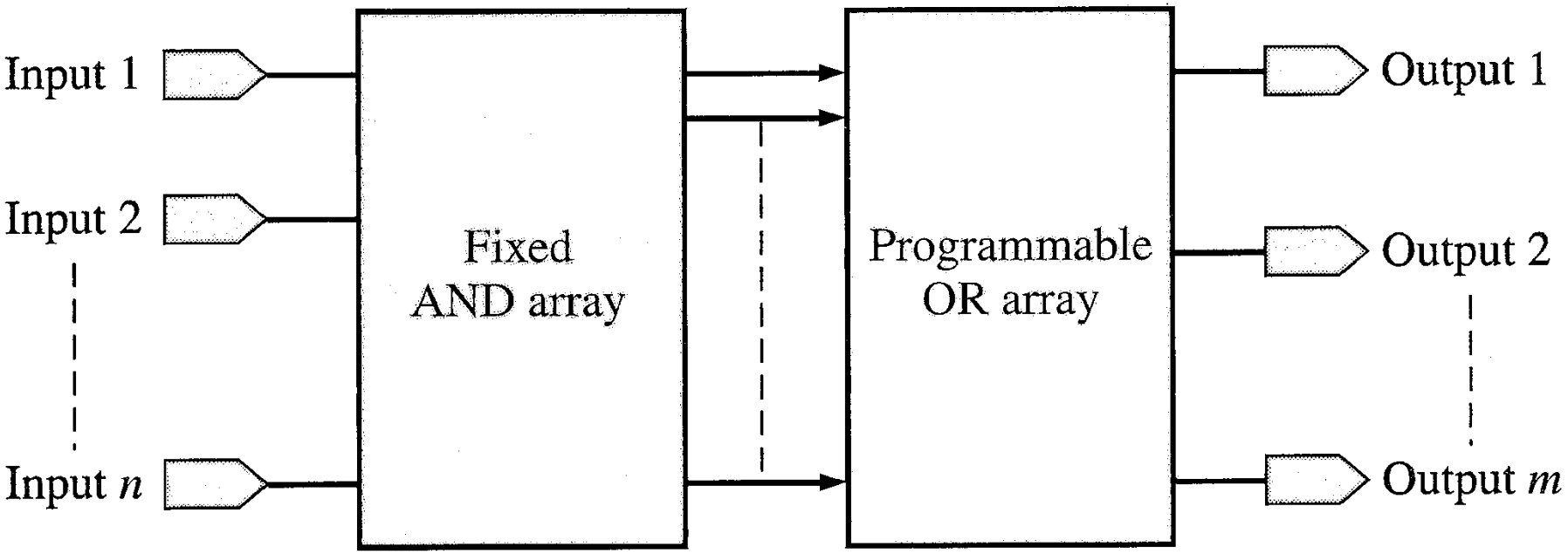
- Úvod
- Mapování na logické členy
- Mapování na kombinační moduly
- Mapování na programovatelná zařízení
- Syntéza sekvenčních obvodů

Implementace log. funkce pomocí ROM

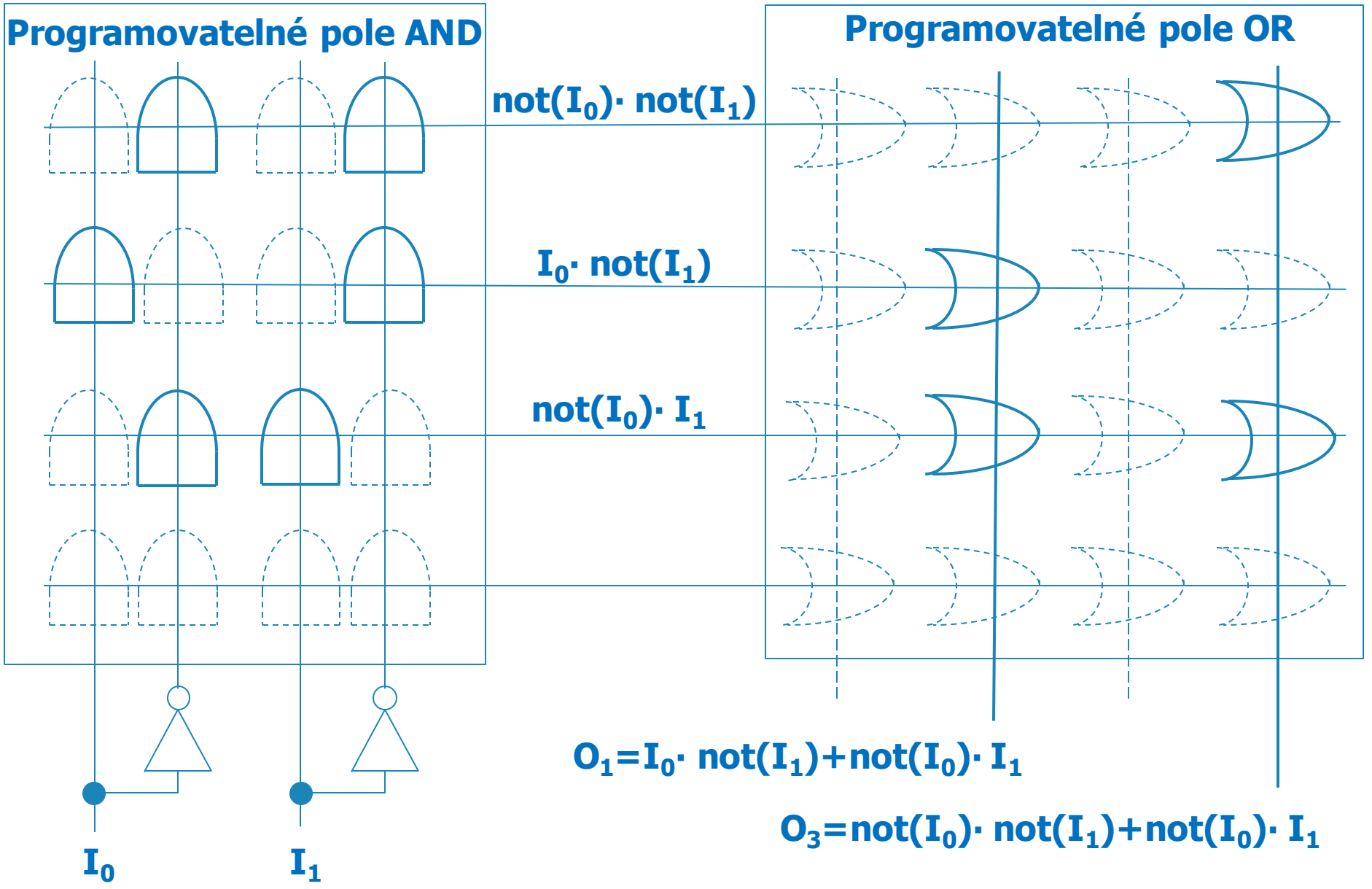


Implementace log. funkce pomocí ROM

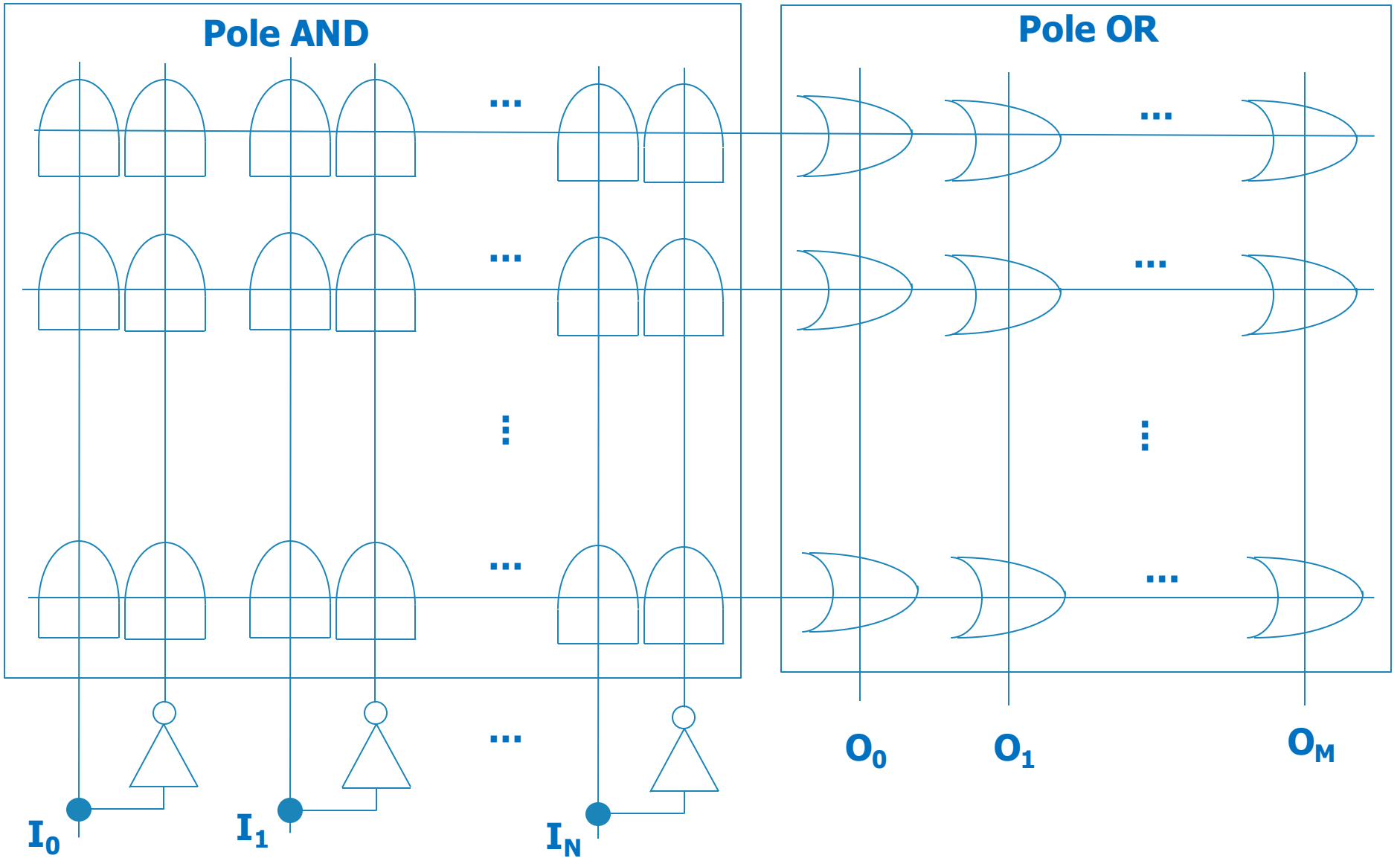
- Použití paměti ROM (PROM) pro realizaci log. funkcí
  - AND pole – dekodér adres
  - OR pole – data uložena v paměti
  - Díky fixnímu AND poli je využití omezeno, neboť pro funkce s více vstupy je třeba velkých kapacit paměti, což je drahé



Implementace log. funkce pomocí PLA



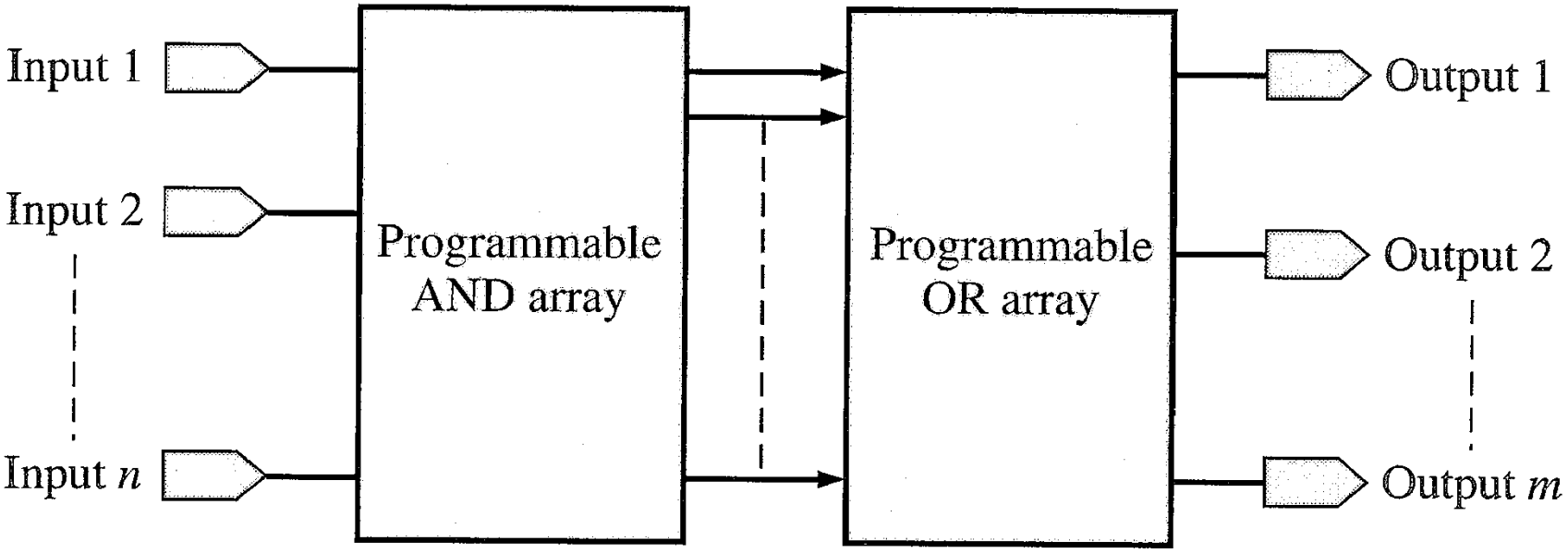
Implementace ÚNDF log. funkce pomocí PLA



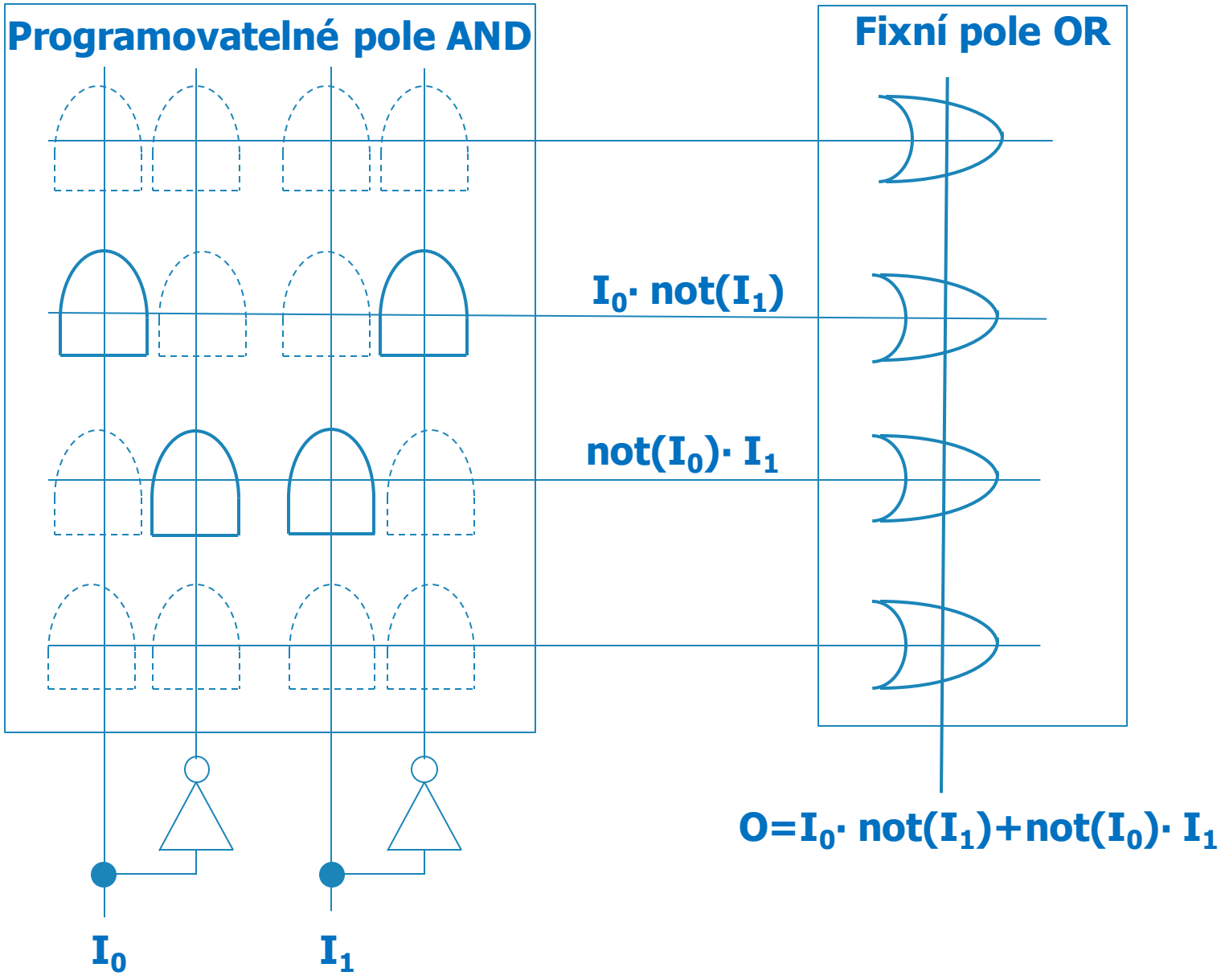
Duálně platí, že pole OR-AND implementuje součin sum (ÚNKF, POS)

Implementace log. funkce pomocí PLA

- Programmable Logic Array (PLA)
  - Výhodou je skutečnost, že jsou obě pole plně programovatelná
  - Nevýhodou je větší zpoždění kvůli programovacím propojkám („pojistkám") v obou polích, které mají větší zpoždění než vodiče a log. členy díky přechodovému odporu a parazitním kapacitám
  - Pomalé, drahé



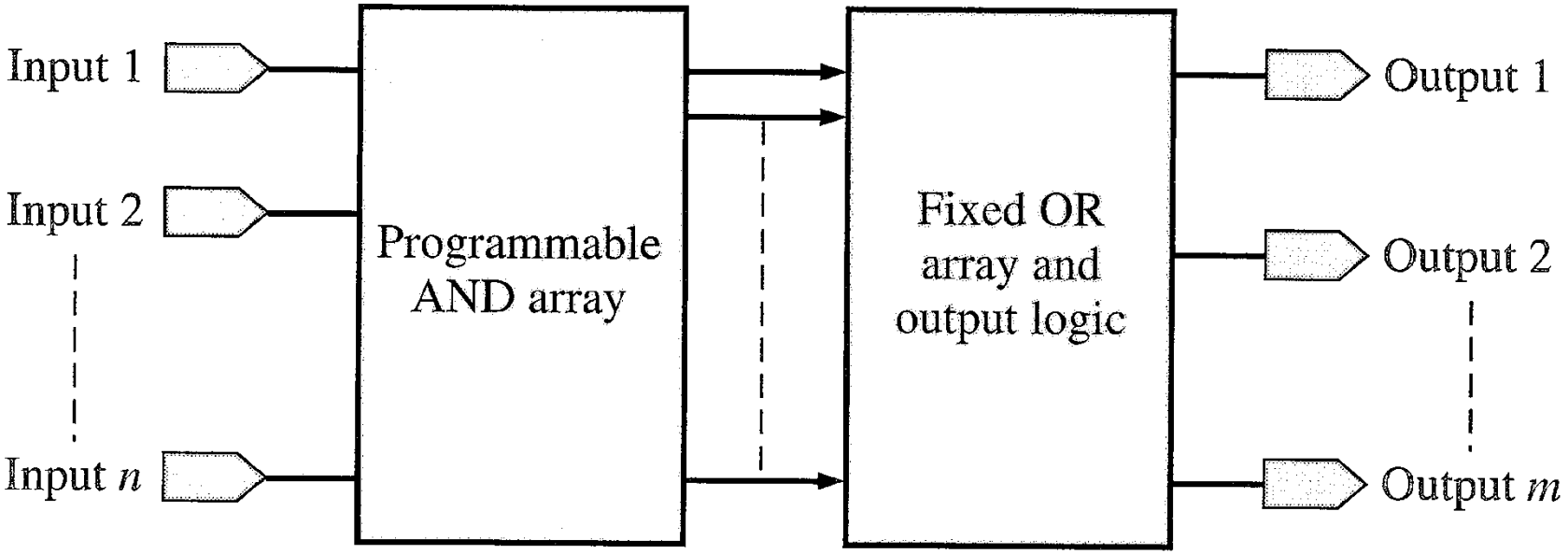
Implementace log. funkce pomocí PAL



$$O = I_0 \cdot \text{not}(I_1) + \text{not}(I_0) \cdot I_1$$

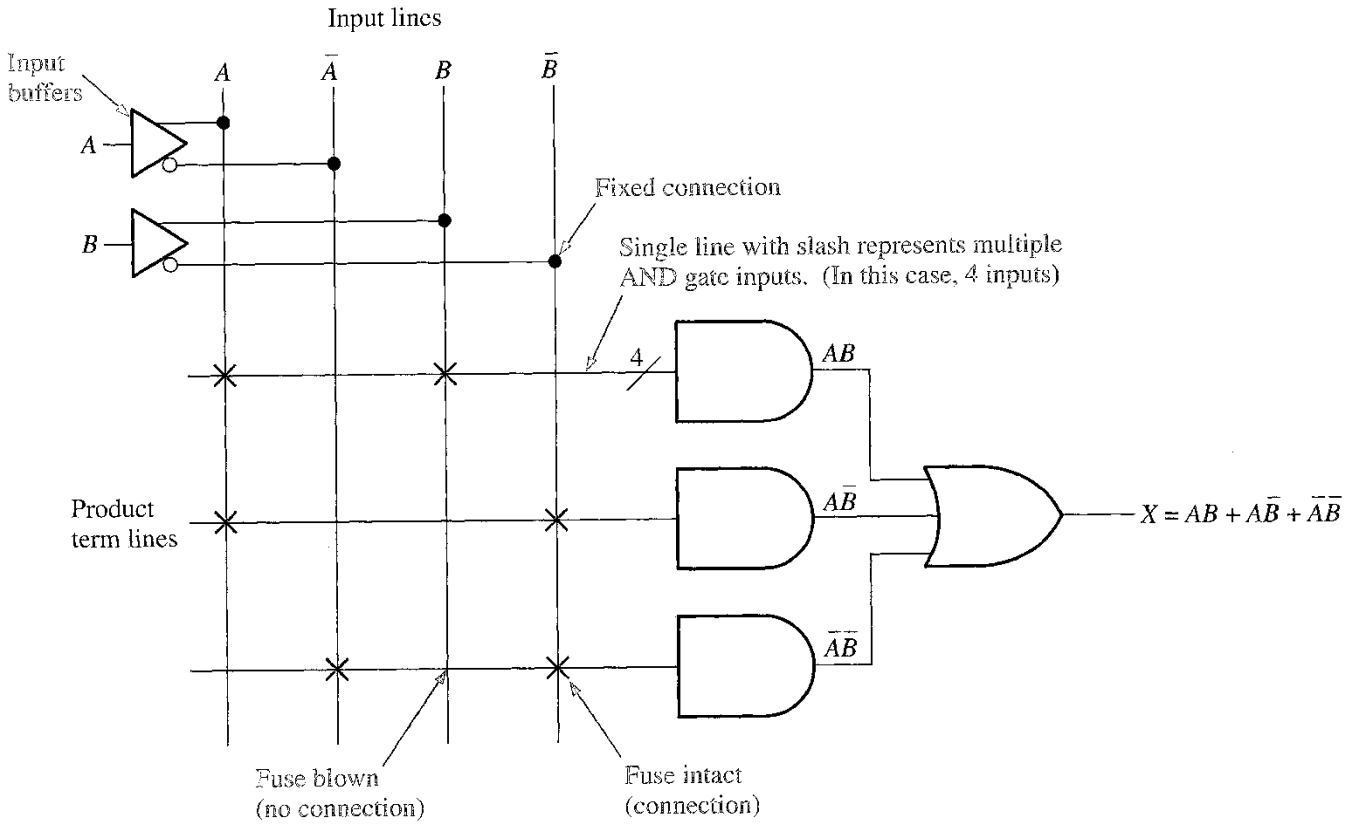
Implementace log. funkce pomocí PAL

- Programmable Array Logic (PAL)
  - Pouze pole AND je programovatelné
  - Nemají omezení pamětí PROM (díky programovatelnému poli AND)
  - Jsou rychlejší díky fixnímu poli OR bez propojek
  - Jsou levnější než PLA
  - Realizují disjunktní formu



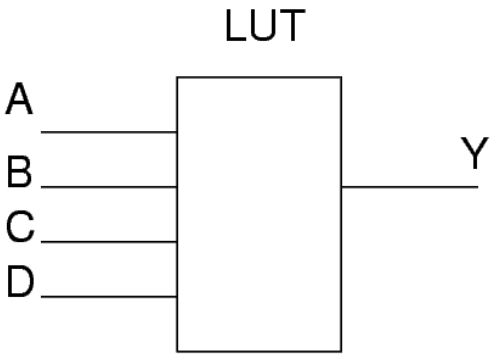
Implementace log. funkce pomocí PAL

- Vnitřní struktura
  - Obsahuje též invertory pro každou vstupní proměnnou
  - Pro naše účely budeme místa, ve kterých propojky zůstanou, značit křížkem



Implementace log. funkce v FPGA

- Look-Up Table (LUT)
  - Základní logické hradlo: *N*-bitový vstup, 1-bitový výstup
  - Realizuje libovolnou binární funkci *N* proměnných
- Technologie
  - Spartan3: N=4
  - Virtex UltraScale : N=6
- Příklad:
  - 4-vstupé hradlo

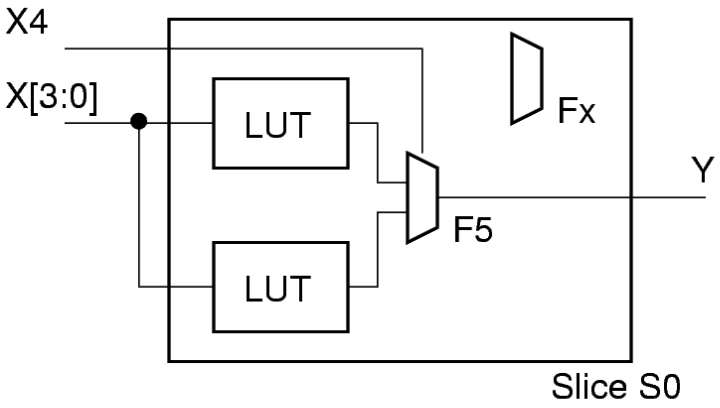


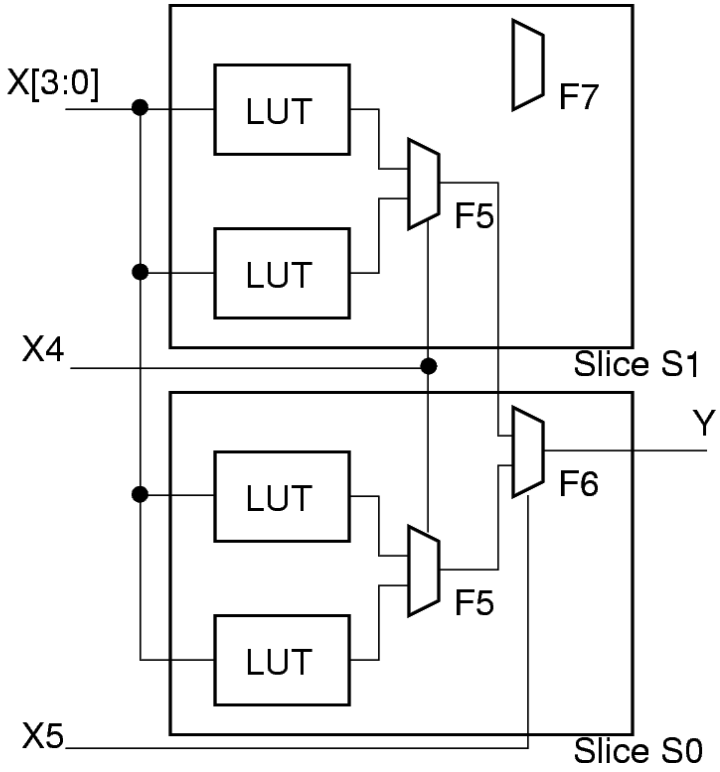
$F(A,B,C,D) = \bar{A}\bar{B}\bar{C}D \text{ or } A\bar{B}\bar{C}\bar{D} \text{ or } \bar{A}\bar{B}C\bar{D} \text{ or } ABCD$

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Implementace log. funkce v FPGA

- Pomocí multiplexerů MUXFx lze jednoduše vytvářet složitější funkce
- MUXFx je potom označován jako MUXF5, MUXF6, …
- Příklad:
  - 4-vstupá hradla LUT





Obsah

- Úvod
- Mapování na logické členy
- Mapování na kombinační moduly
- Mapování na programovatelná zařízení
- Syntéza sekvenčních obvodů

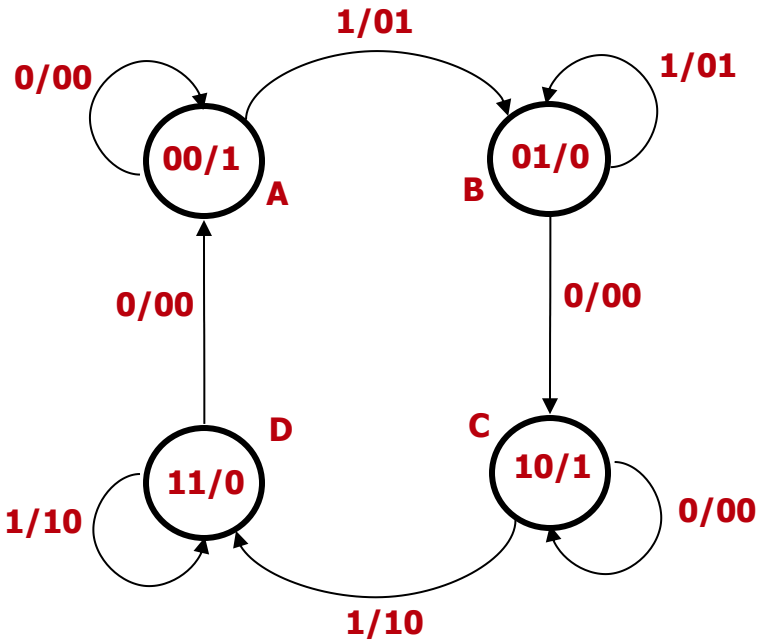
Syntéza automatu robota

- Navrhněme řídicí obvod (konečný automat) robota, který bude fungovat následovně
  - Robot má dvě kolečka a umí se otáčet vlevo a vpravo
    - Výstup z1=1 – zatoč vlevo
    - Výstup z2=1 – zatoč vpravo
    - Výstupy z1=0 a z2=0 – robot jede dopředu
  - Robot má vpředu dotykový senzor, který generuje signál x následovně
    - Vstup x=1 – je detekována překážka
    - Vstup x=0 – žádná překážka není detekována
  - Robot má na střeše majáček
    - Výstup m=1 – svítí, pokud robot jede dopředu
    - Výstup m=0 – nesvítí, pokud robot zatáčí vlevo či vpravo
  - Jedno z možných „inteligentních" chování robota – definice stavů řídícího automatu
    - Stav A – překážka není detekována, poslední zatočení bylo vlevo
    - Stav B – překážka detekována, zatoč vpravo
    - Stav C – překážka není detekována, poslední zatočení bylo vpravo
    - Stav D – překážka detekována, zatoč vlevo

Syntéza automatu robota

- Graf přechodů
  - Lze nakreslit např. dle slovního popisu či tabulky přechodů
  - Ve stavech jsou uvedeny kódy stavu/hodnota Mooreova výstupu M - kód/M
  - Na hranách je uvedena hodnota vstupu X/hodnota Mealyho výstupů Z1 a Z2 - X/Z_1Z_2

Současný stav		Vstup X	Následující stav		Mealyho výstupy Z1 Z2	Mooreův výstup M
Název	Kód		Název	Kód		
A	00	0	A	00	00	1
A	00	1	B	01	01	1
B	01	0	C	10	00	0
B	01	1	B	01	01	0
C	10	0	C	10	00	1
C	10	1	D	11	10	1
D	11	0	A	00	00	0
D	11	1	D	11	10	0



Slovník přechodů klopných obvodů

- Opakování
  - Určuje požadované hodnoty na excitačních vstupech (S a R, J a K, D či T) klopného obvodu, které je třeba aplikovat, aby automat provedl požadovaný přechod ze současného stavu Q_i do následujícího stavu Q_{i+1}
  - Jedná se o naopak zapsané tabulky přechodů – pro dané hodnoty výstupů definujeme potřebné hodnoty vstupů

Q_i	Q_{i+1}	S	R	J	K	D	T
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	0	1
1	1	X	0	X	0	1	0

- Při návrhu (syntéze) sekvenčních automatů používáme excitační tabulky KO, neboť nás zajímají přechody mezi stavy automatu, na základě kterých definujeme hodnoty vstupů KO

Syntéza automatu robota

- Určení rovnic pro excitační vstupy jednotlivých KO
  - Současný stav – Present State (PS)
  - Hodnota vstupu – X
  - Následující stav – Next State (NS)
- Doplníme hodnoty pro všechny excitační vstupy jednotlivých KO

Q_i	Q_{i+1}	S	R	J	K	D	T
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	0	1
1	1	X	0	X	0	1	0

souč. stav		vstup	násl. stav		vstupy S-R				vstupy J-K				vstupy D		vstupy T		výstupy		
PS1	PS0	X	NS1	NS0	S1	R1	S0	R0	J1	K1	J0	K0	D1	D0	T1	T0	M	Z1	Z2
0	0	0	0	0	0	X	0	X	0	X	0	X	0	0	0	0	1	0	0
0	0	1	0	1	0	X	1	0	0	X	1	X	0	1	0	1	1	0	1
0	1	0	1	0	1	0	0	1	1	X	X	1	1	0	1	1	0	0	0
0	1	1	0	1	0	X	X	0	0	X	X	0	0	1	0	0	0	0	1
1	0	0	1	0	X	0	0	X	X	0	0	X	1	0	0	0	1	0	0
1	0	1	1	1	X	0	1	0	X	0	1	X	1	1	0	1	1	1	0
1	1	0	1	1	X	0	X	0	X	0	X	0	1	1	0	0	0	0	0
1	1	1	0	0	0	1	0	1	X	1	X	1	0	0	1	1	0	1	0

Syntéza automatu robota

- Nalezneme rovnice pro excitační vstupy jednotlivých KO, např. pomocí Karnaughovy mapy
  - Pro jednotlivé excitační vstupy D0 a D1 KO
  - Pro jednotlivé výstupy Z1, Z2 a M

$$M = \overline{PS0}$$

- Řešení pro KO D

D0

x \ PS	PS0	PS1
0	0	0
1	1	0

$$D0 = X \cdot \overline{PS0} + \overline{X} \cdot PS0 \cdot PS1 + X \cdot PS0 \cdot \overline{PS1}$$

Z1

x \ PS	PS0	PS1
0	0	0
0	1	1

$$Z1 = X \cdot PS1$$

D1

x \ PS	PS0	PS1
0	1	1
0	1	0

$$D1 = \overline{PS0} \cdot PS1 + \overline{X} \cdot PS0$$

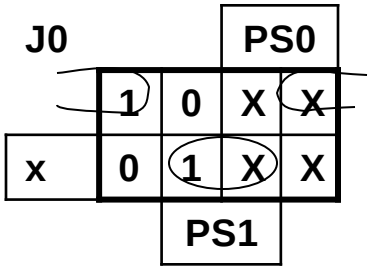
Z2

x \ PS	PS0	PS1
0	0	0
1	0	0

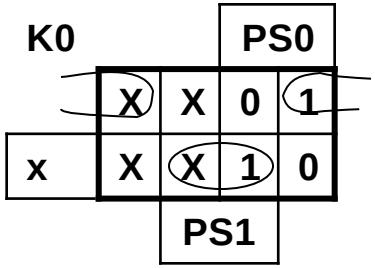
$$Z2 = \overline{PS1} \cdot X$$

Syntéza automatu robota

- Řešení pro KO J-K



$$J0 = X \cdot PS1 + \overline{X} \cdot \overline{PS1}$$

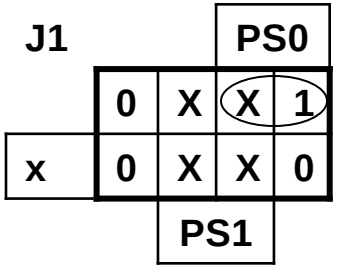


$$K0 = X \cdot PS1 + \overline{X} \cdot \overline{PS1}$$

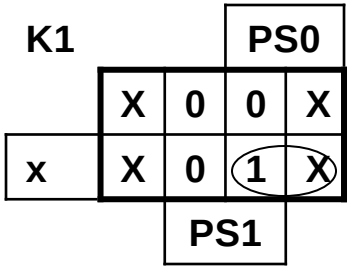
$$M = \overline{PS0}$$

$$Z1 = X \cdot PS1$$

$$Z2 = \overline{PS1} \cdot X$$



$$J1 = \overline{X} \cdot PS0$$

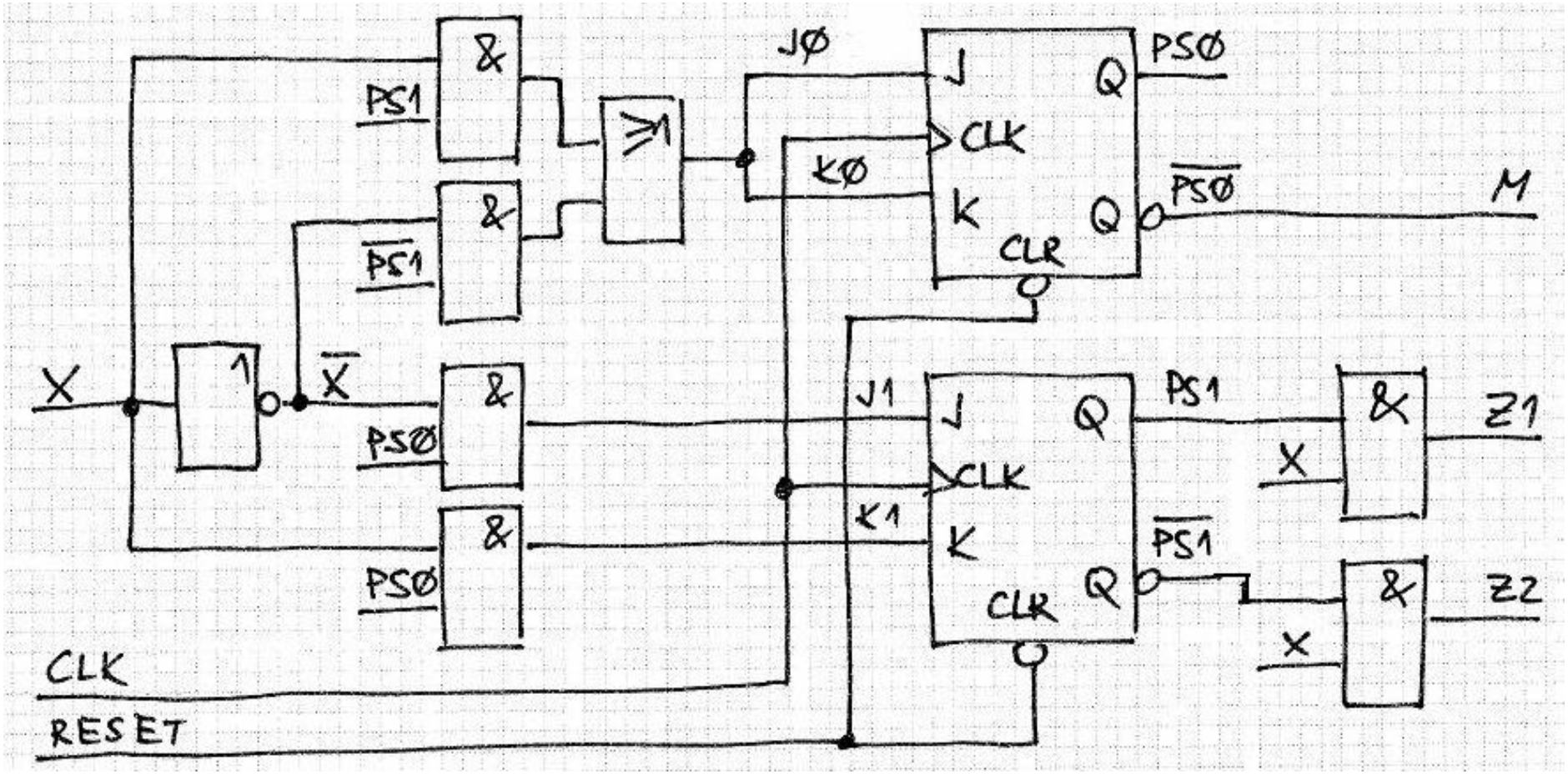


$$K1 = X \cdot PS0$$

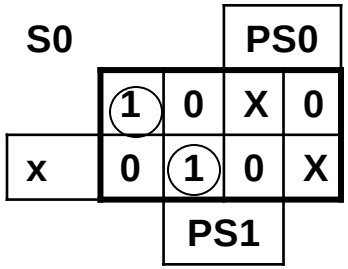
- Řešení pro KO J-K – schéma

$$J0 = X \cdot PS1 + \overline{X} \cdot \overline{PS1} \qquad K0 = X \cdot PS1 + \overline{X} \cdot \overline{PS1} \qquad M = \overline{PS0} \qquad Z1 = X \cdot PS1$$

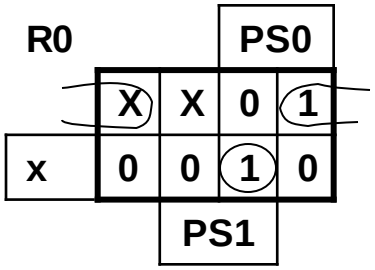
$$J1 = \overline{X} \cdot PS0 \qquad K1 = X \cdot PS0 \qquad Z2 = \overline{PS1} \cdot X$$



- Řešení pro KO R-S



$$S0 = X \cdot \overline{PS0} \cdot PS1 + \overline{X} \cdot \overline{PS0} \cdot \overline{PS1}$$

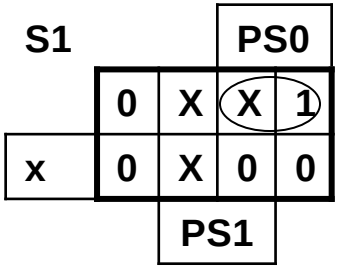


$$R0 = \overline{X} \cdot \overline{PS1} + X \cdot PS0 \cdot PS1$$

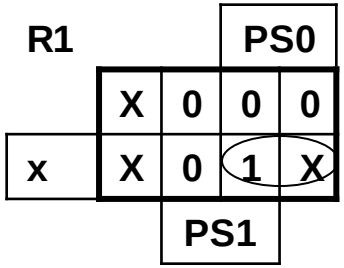
$$M = \overline{PS0}$$

$$Z1 = X \cdot PS1$$

$$Z2 = \overline{PS1} \cdot X$$



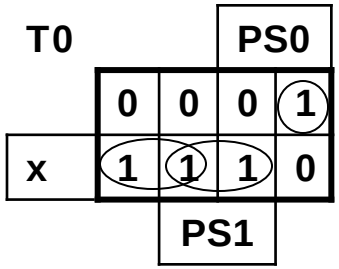
$$S1 = \overline{X} \cdot PS0$$



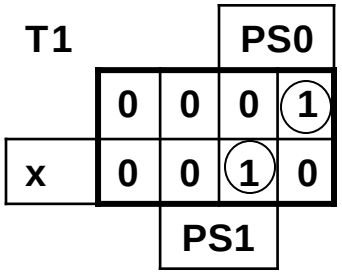
$$R1 = X \cdot PS0$$

Syntéza automatu robota

- Řešení pro KO T



$$T0 = X \cdot \overline{PS0} + X \cdot PS1 + \overline{X} \cdot PS0 \cdot \overline{PS1}$$



$$T1 = X \cdot PS0 \cdot PS1 + \overline{X} \cdot PS0 \cdot \overline{PS1}$$

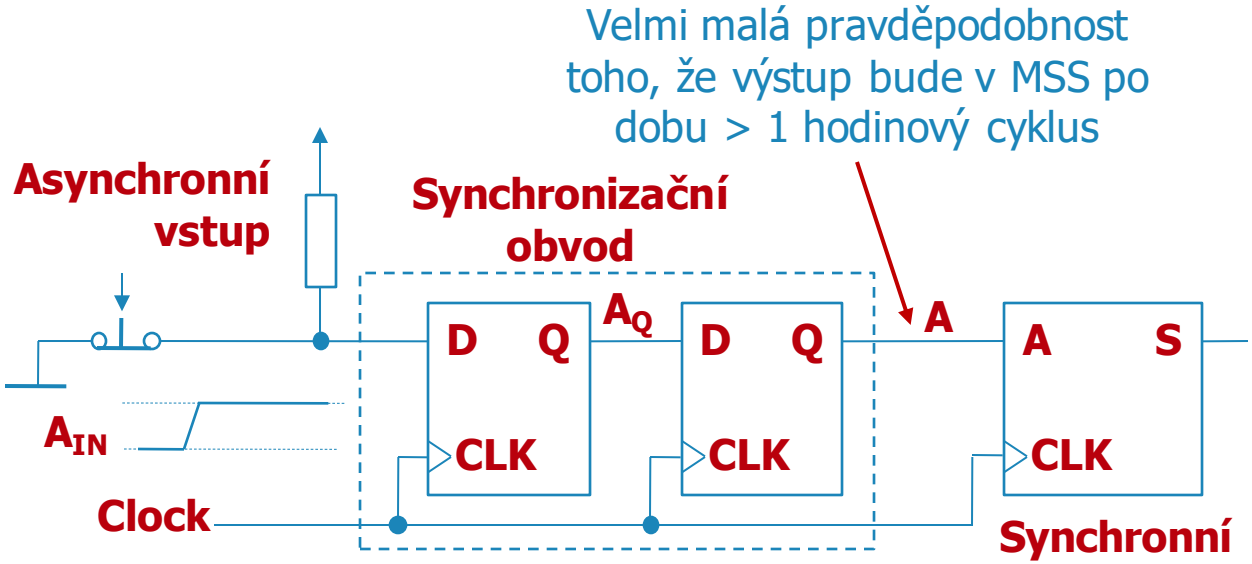
$$M = \overline{PS0}$$

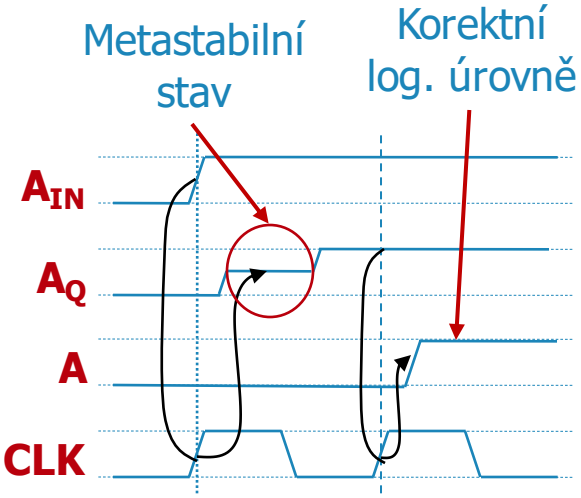
$$Z1 = X \cdot PS1$$

$$Z2 = \overline{PS1} \cdot X$$

Syntéza detektoru náběžné hrany

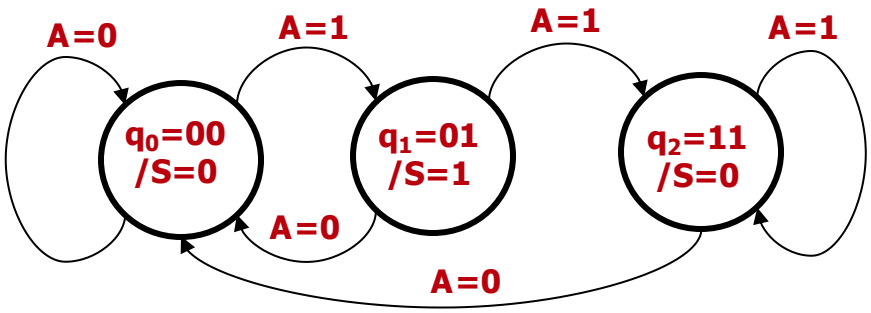
- Synchronizační obvod
  - Omezuje pravděpodobnost toho, že se metastabilní stav projeví na vstupu synchronního automatu
  - Výstupem je zasynchronizovaná vstupní asynchronní událost
  - Vstupní událost je zpožděna o 2 takty hodinového signálu
- Synchronní obvod
  - Konečný automat detekující kladnou hranu
  - Zkusíme implementovat jako Mooreův i Mealyho konečný automat





Syntéza detektoru náběžné hrany - Moore

- Varianta 1: Mooreův konečný automat
  - Kódování stavů je výhodné volit tak, aby byla implementace co nejjednodušší



Současný stav PS		Vstup A	Následující stav NS		Výstup Mealy	Výstup Moore S
Název	Kód Q1,Q0		Název	Kód D1,D0		
q_0	00	0	q_0	00	-	0
q_0	00	1	q_1	01	-	0
q_1	01	0	q_0	00	-	1
q_1	01	1	q_2	11	-	1
q_2	11	0	q_0	00	-	0
q_2	11	1	q_2	11	-	0

Následující stav D1		Současný stav Q1,Q0			
		00	01	11	10
A	0	0	0	0	X
	1	0	1	1	X

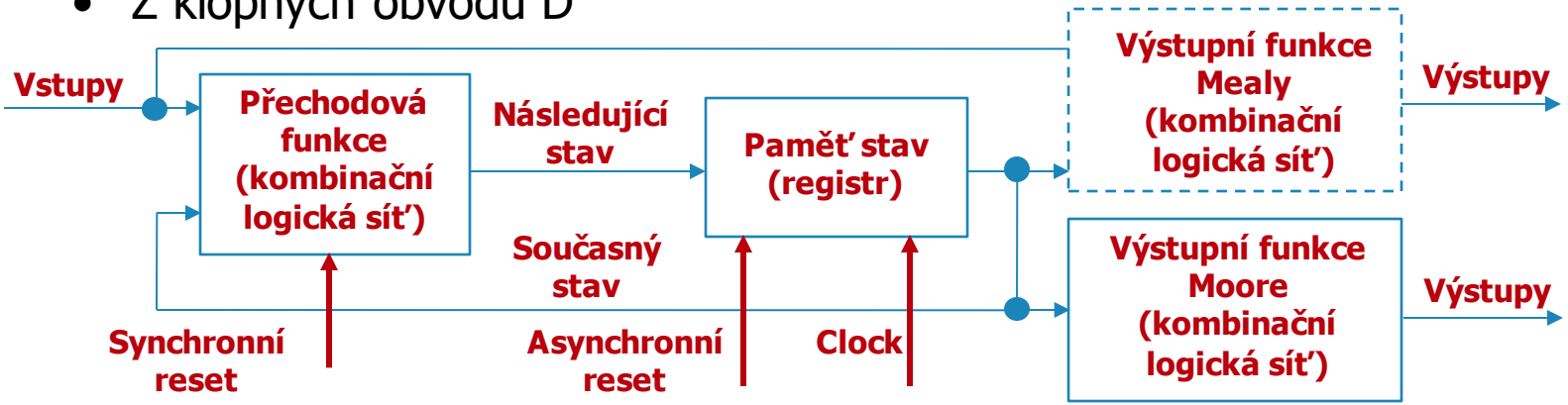
Následující stav D0		Současný stav Q1,Q0			
		00	01	11	10
A	0	0	0	0	X
	1	1	1	1	X

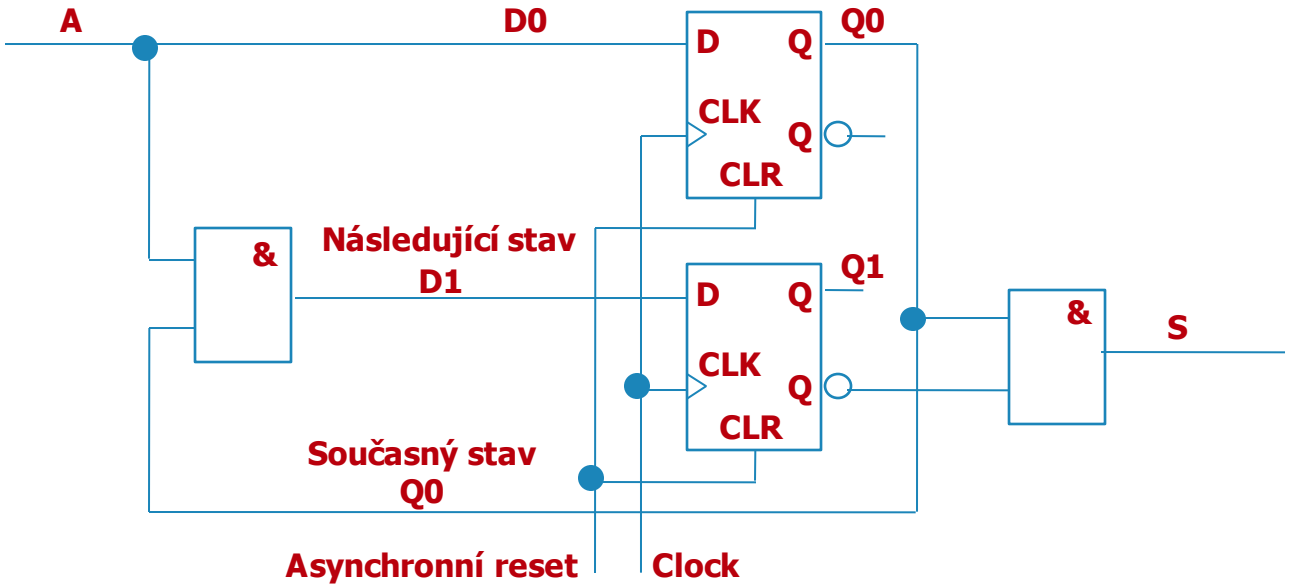
Mooreův výstup S		Současný stav Q1	
		0	1
Současný stav Q0	0	0	X
	1	1	0

$$D1 = A \cdot Q0 \qquad D0 = A \qquad S = \overline{Q1} \cdot Q0$$

Syntéza detektoru náběžné hrany - Moore

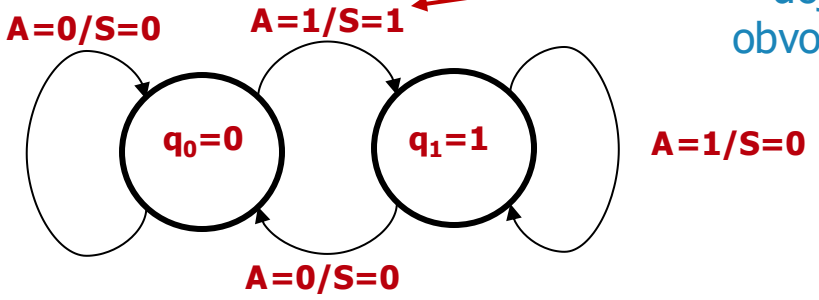
- Synchronní sekvenční obvod
  - Z klopných obvodů D



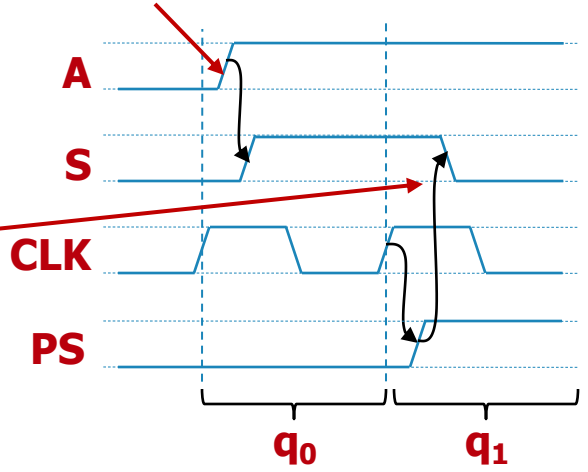


Syntéza detektoru náběžné hrany - Mealy

- Varianta 2: Mealyho konečný automat
  - Mealyho automat může mít méně stavů než Mooreův



Bezprostředně po změně A=0->1 (při PS=q_0), dojde ke změně výstupu S=0->1, ve které obvod setrvá až do provedení přechodu PS=q_1



Po přechodu do stavu PS=q_1 dojde ke změně výstupu S=1->0 i když vstup setrvá v A=1

Současný stav PS		Vstup A	Následující stav NS		Výstup Mealy S	Výstup Moore
Název	Kód Q0		Název	Kód D0		
q_0	0	0	q_0	0	0	-
q_0	0	1	q_1	1	1	-
q_1	1	0	q_0	0	0	-
q_1	1	1	q_1	1	0	-

Mealyho výstup S		Vstup A	
		0	1
Současný stav Q0	0	0	1
	1	0	0

$$D0 = A \qquad S = A \cdot \overline{Q0}$$

Syntéza detektoru náběžné hrany - Mealy

- Synchronní sekvenční obvod
  - Z klopných obvodů D