Univerzita J. E. Purkyně v Ústí nad Labem
Přírodovědecká fakulta

Základy elektroniky
(opora)

Petr Haberzettl

Ústí nad Labem 2020

Analogová část

Výchozí předpoklady:

- obvody se soustředěnými parametry – všechny sledované veličiny jsou pouze funkcemi času,
- obvody jsou v ustáleném stejnosměrném stavu – obvodové veličiny (napětí a proud) jsou v závislosti na čase konstantní,
- obvody jsou tvořeny ideálními lineárními prvky.

1. Pasivní ideální prvky:

- rezistor (ideální odporník),
- induktor (ideální cívka),
- kapacitor (ideální kondenzátor).

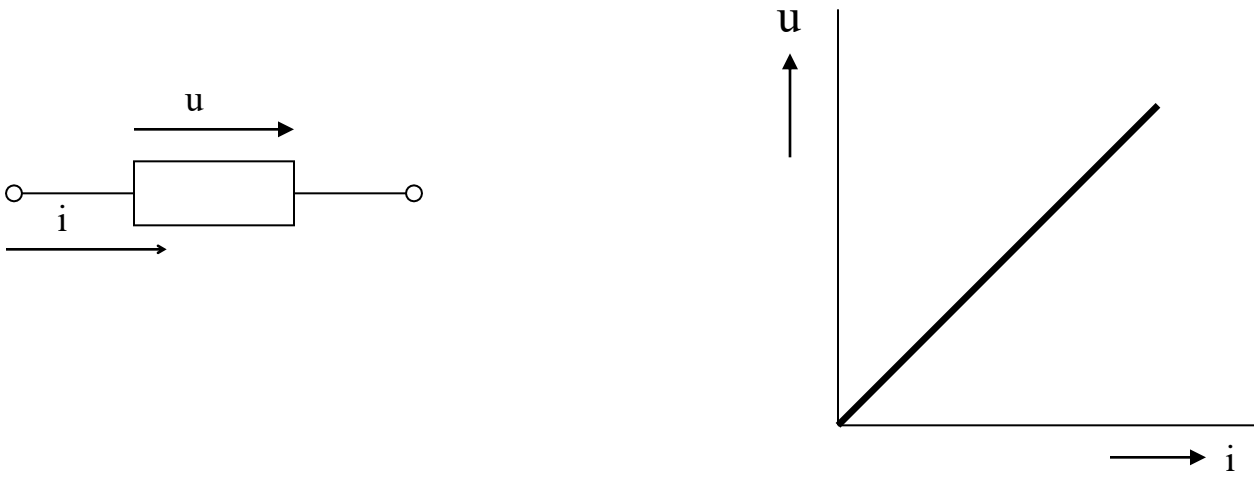
2. Aktivní ideální prvky:

- ideální zdroj napětí,
- ideální zdroj proudu.

Rezistor:

Je takový prvek, v němž dochází pouze k přeměně elektrické energie na tepelnou a jehož jediným a konstantním parametrem je **elektrický odpor.**

Definiční rovnice je Ohmův zákon.

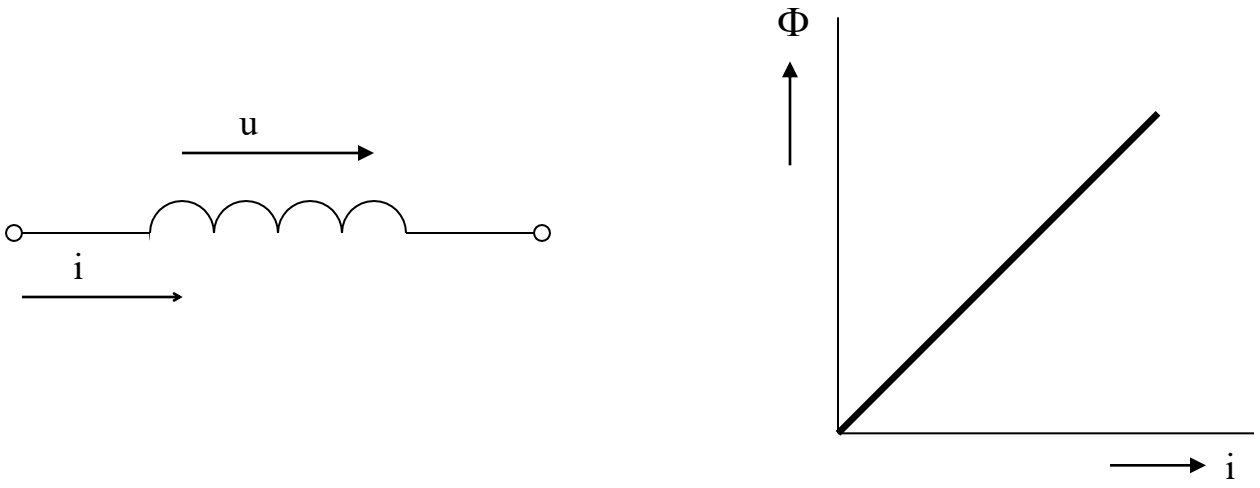


Obr.1. Schématická značka a voltampérová charakteristika rezistoru

Induktor:

Je prvek, v němž se akumuluje a vydává jen energie magnetického pole, přičemž nevznikají tepelné ztráty. Jediným a konstantním parametrem je **indukčnost**.

Pro magnetický tok **Φ** jednoho závitu platí: $\Phi = L \bullet i$, kde **L** je vlastní indukčnost a **i** je proud. Z indukčního zákona platí pro napětí: $u = L(d\mathbf{i}/d\mathbf{t})$.

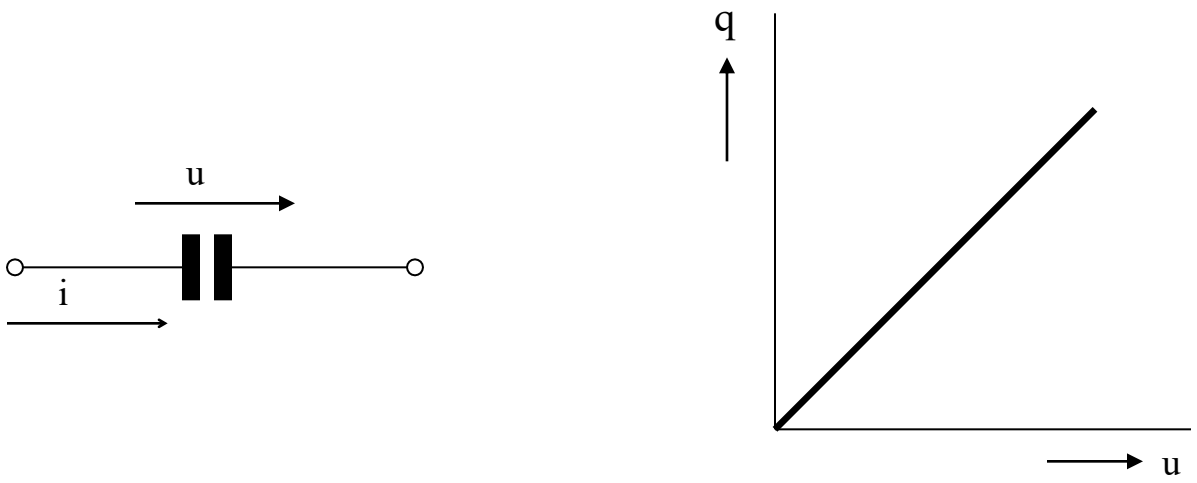


Obr.2. Schématická značka a weberampérová charakteristika induktoru

Kapacitor:

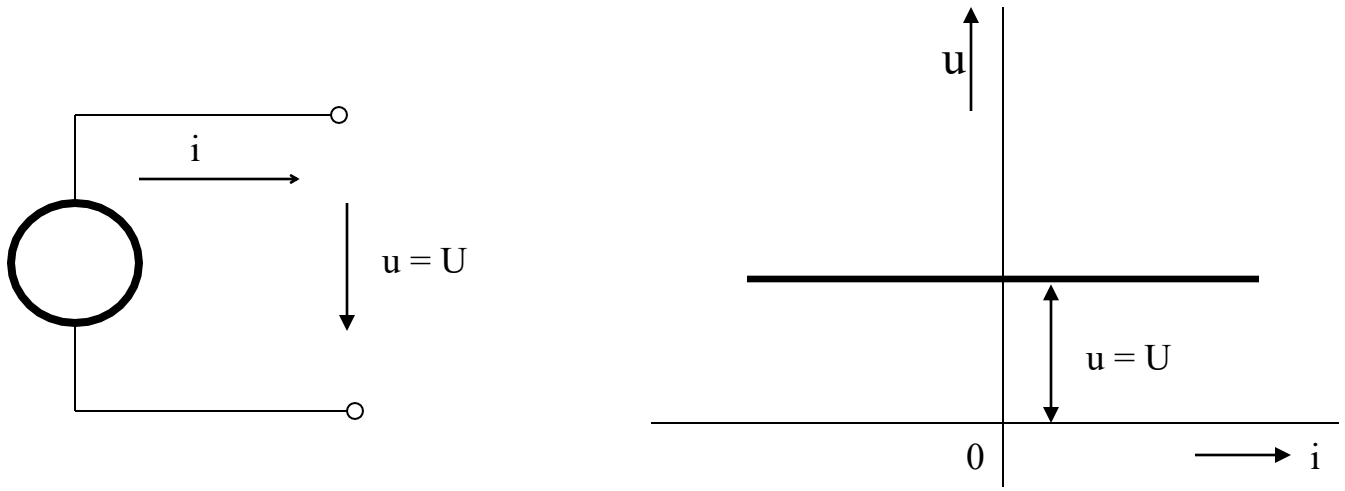
Je prvek, v němž se akumuluje jen energie elektrického pole, přičemž nevznikají tepelné ztráty. Jediným a konstantním parametrem je **kapacita.**

Pro elektrický náboj **q** platí: q = C • u , kde **C** je kapacita a **u** je napětí.
Ze zákona zachování elektrického náboje platí pro proud: i = **C**(d**i**/d**t**).



Obr.3. Schématická značka a coulombvoltová charakteristika kapacitoru

Ideální zdroj napětí:
Je takový zdroj, jehož napětí nezávisí na velikosti odebíraného proudu.



Obr.4. Schématická značka a voltampérová charakteristika

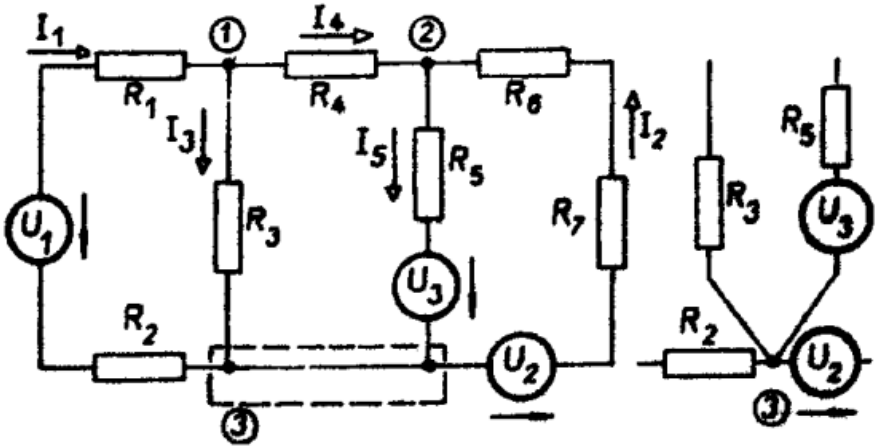
Ideální zdroj proudu:
Je takový zdroj, jehož proud nezávisí na odebíraném napětí.

Obr.5. Schématická značka a voltampérová charakteristika

Topologie elektrických obvodů:

Elektrické obvody se zpravidla skládají z většího počtu aktivních a pasivních dvojpólů:

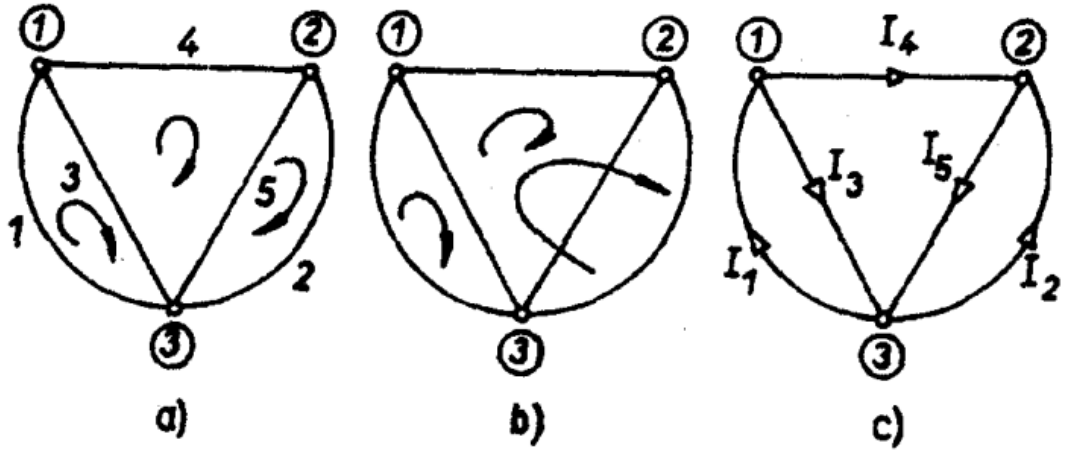
- každý jednotlivý dvojpól je do obvodu zapojen dvěma svorkami – **póly**,
- svorka dvojpólu se nazývá **uzel prvního řádu,**
- spoj dvou dvojpólů tvoří **uzel druhého řádu**,
- spoj tří dvojpólů tvoří **uzel třetího řádu** atd.,
- uzly jsou v elektrickém obvodu spojeny **větvemi**,
- libovolný uzavřený okruh v elektrickém obvodu se nazývá **smyčka**.



Obr.6. Složitější elektrický obvod

Čárový graf:

Značky dvojpólů jsou vynechány a větve jsou zakresleny jednoduchými čarami spojující příslušné uzly. Zakreslením zvolených směrů proudů vznikne **orientovaný čárový graf**.



Obr.7. Čárový graf obvodu a volba smyček

Metody řešení lineárních obvodů:

Řešení (analýza) obvodu spočívá v tom, že pro daný obvod a zadané parametry zdrojů hledáme ostatní obvodové veličiny.

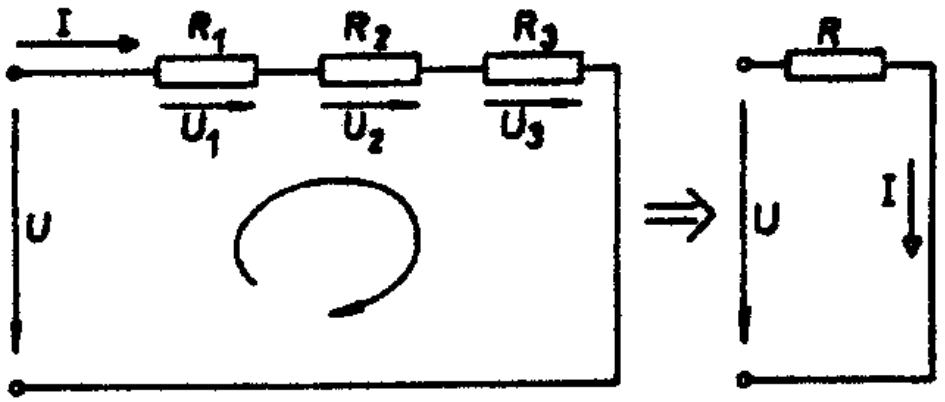
Metody řešení lineárních odporových obvodů můžeme rozdělit na:

- metody postupného zjednodušování obvodu,
- řešení obvodu Kirchhoffovými zákony,
- ostatní metody.

Metoda postupného zjednodušování obvodu:

Metoda je vhodná pro obvody s jedním zdrojem nebo pro skupiny pasivních dvojpólů ve složitějších obvodech. Její podstatou je nahrazování sériových a paralelních skupin pasivních dvojpólů ekvivalentními pasivními dvojpóly podle těchto zásad:

1. **rezistory v sérii**: je-li několik pasivních dvojpólů zapojeno v sérii, protéká jimi jediný společný proud. Odpor $R = R_1 + R_2 + R_3$.
   Obecně je výsledný odpor sériového spojení rezistorů : $R = \sum_{x=1}^{n} R_x$

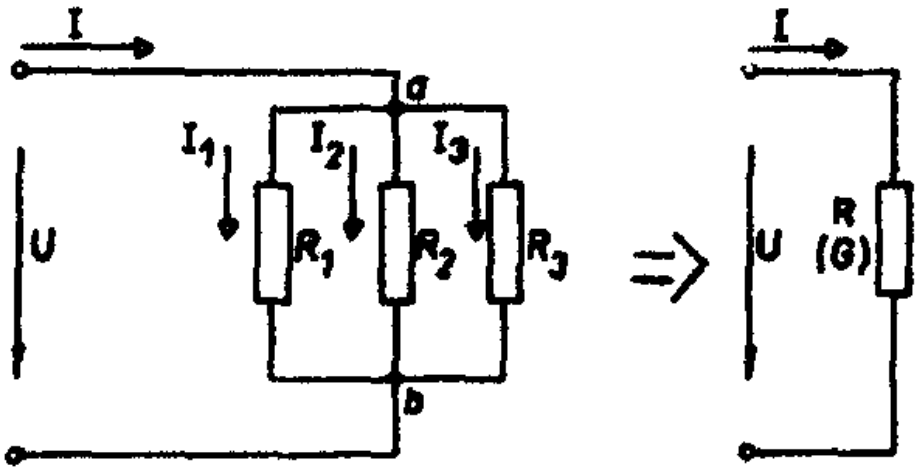


Obr.8. Rezistory v sérii a ekvivalentní náhradní schéma

Metoda postupného zjednodušování obvodu:

2. **rezistory paralelně**: na rezistorech spojených paralelně je jedno společné napětí U.
Celkový proud se dělí: $I = I_1 + I_2 + I_3 = U \bullet (G_1 + G_2 + G_3) = U \bullet G = U/R$.

Při paralelním spojení rezistorů se sčítají jejich vodivosti: $G = \sum_{x=1}^{n} G_x$, $\frac{1}{R} = \sum_{x=1}^{n} \frac{1}{R_x}$

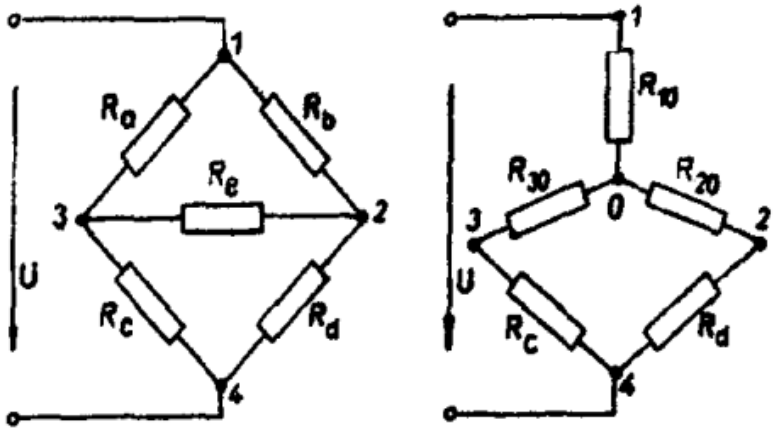


Obr.8. Rezistory paralelně a ekvivalentní náhradní schéma

Metoda postupného zjednodušování obvodu:

3. **transfigurace trojúhelníka na hvězdu**: používá se v případech, kdy se vykytují skupiny tří dvojpólů, které nejsou zapojeny ani sériově ani paralelně. Transfigurací se nesmí změnit poměry vně transfigurované oblasti.

$$R_{10} = \frac{R_a \cdot R_b}{R_a + R_b + R_e}, \quad R_{20} = \frac{R_b \cdot R_e}{R_a + R_b + R_e}, \quad R_{30} = \frac{R_a \cdot R_e}{R_a + R_b + R_e}$$



Obr.9. Můstkové zapojení a jeho úprava transfigurací

Řešení obvodu Kirchhoffovými zákony:

I KZ: Algebraický součet proudů ve kterémkoliv uzlu se rovná nule, $\sum_{x=1}^{n} I_n = 0.$

II KZ: Součet svorkových napětí zdrojů a všech úbytků napětí na rezistorech je v uzavřeném obvodě roven nule,

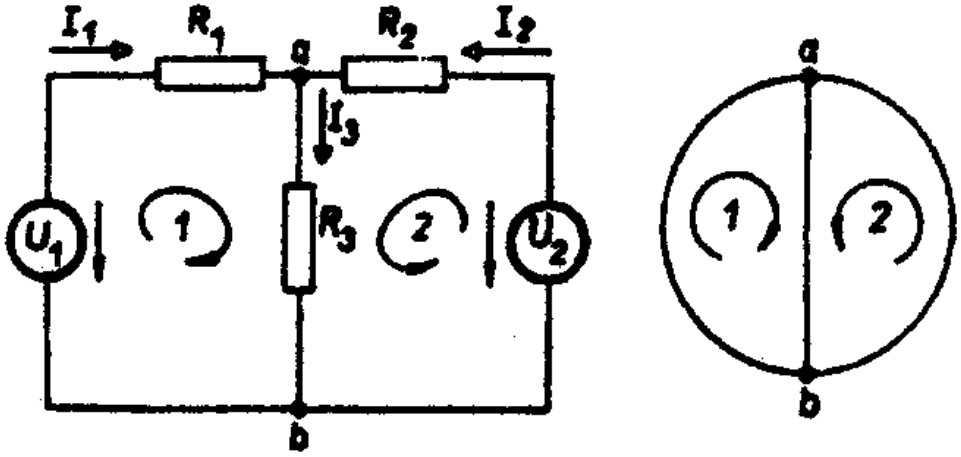
$$\sum_{x=1}^{n} U_x = 0.$$

Pro I KZ: $-I_1 - I_2 + I_3 = 0,$

pro II KZ: $R_1 I_1 + R_2 I_2 - U_1 = 0$ (1.smyčka),

$R_2 I_2 + R_3 I_3 - U_2 = 0$ (2.smyčka).

Řešení: soustava tří nezávislých lineárních rovnic.



Obr.10. Řešení obvodu Kirchhoffovými zákony

Další metody řešení lineárních obvodů:

- **princip superpozice**: účinek všech zdrojů v lineárním obvodu je roven součtu účinků jednotlivých zdrojů působících samostatně; skutečné proudy ve větvích obvodu jsou dány superpozicí dílčích proudů,

- **metoda uzlových napětí**: na nezávislé uzly obvodu aplikujeme I KZ, přičemž dvojpóly ve větvích s proudy vyjadřujeme uzlovými napětími,

- **metoda smyčkových proudů**: pro smyčkové proudy aplikujeme jen II KZ.

digitální část

Přehled

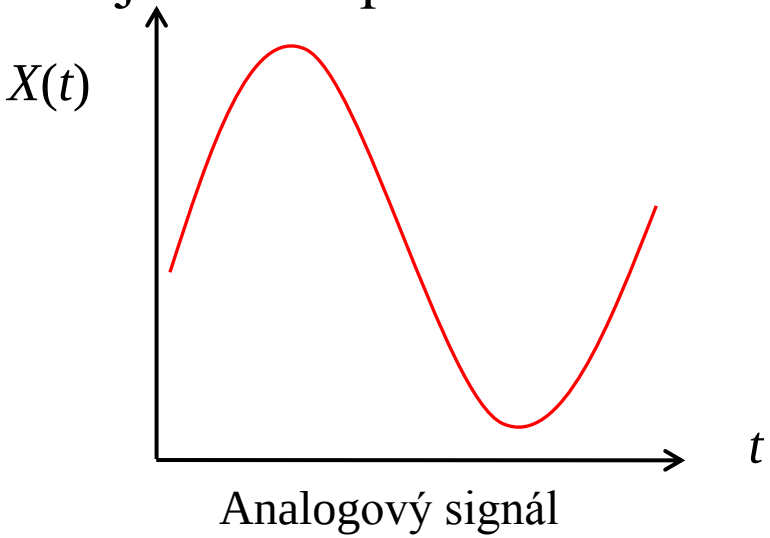
- 1.1 Digitální systémy
- 1.2 Binární čísla
- 1.3 Převod soustav
- 1.4 Oktalová a hexadecimální čísla
- 1.5 Doplňky
- 1.6 Znaménková binární čísla
- 1.7 Binární kódy
- 1.8 Binární uložiště a registry
- 1.9 Binární logika

Digitální systémy a binární čísla

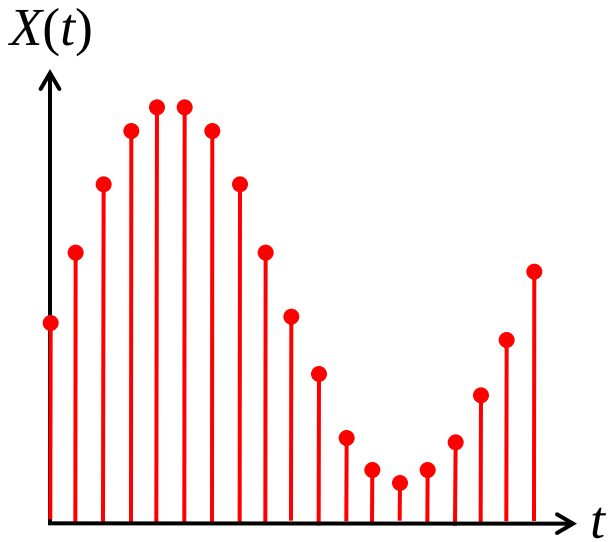
- Digitální počítače
  - Hlavní účel
    - Vědecké, industriální a reklamní aplikace
- Digitální systémy
  - Telefonní spínací ústředny
  - Digitální kamery
  - Elektronické kalkulátory
  - Digitální televizory
- Systém diskrétního zpracování informací
  - Manipulace s diskrétními informacemi
  - Například: {1, 2, 3, …} a {A, B, C, …}…

Analogový a digitální signál

- Analogový systém
  - Fyzické veličiny nebo signály se mohou průběžně měnit v předepsaném rozsahu
- Digitální systémy
  - Fyzické veličiny nebo signály mohou nabývat pouze konkrétních hodnot
  - Mají velkou přesnost



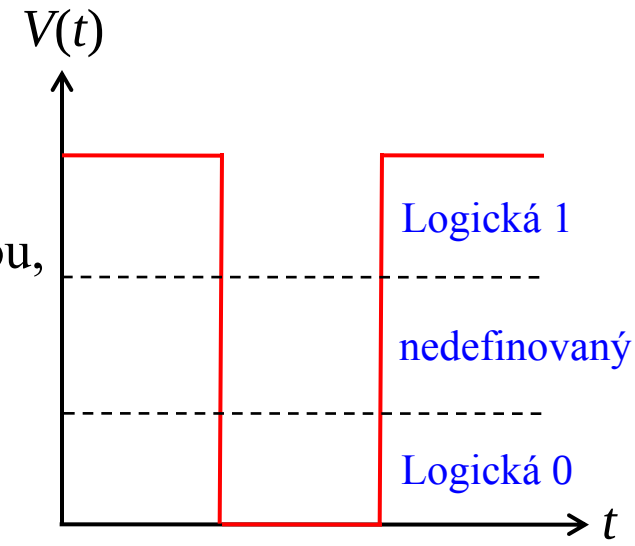
Analogový signál



Digitální signál

Binární digitální signál

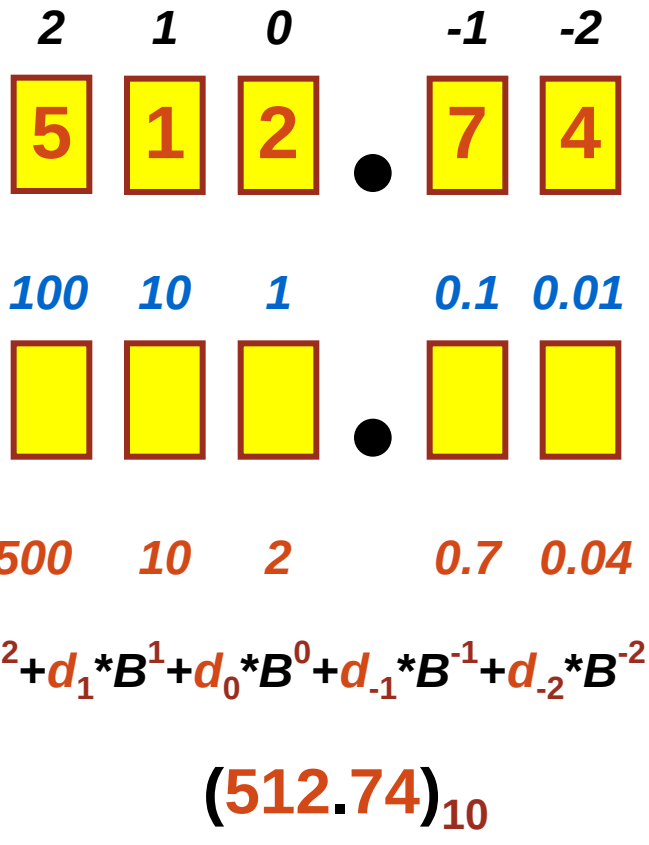
- Informace je reprezentována fyzickou hodnotou
- U digitálních systémů hodnota nabývá konkrétních hodnot
  - Binární soustava je nejpoužívanější
- Binární hodnoty jsou reprezentovány:
  - Logickou 0 a 1
  - Slovy (symboly) True (T) a False (F)
  - Slovy (symboly) High (H) a Low (L)
  - A slovy On nebo Off
- Binární hodnoty jsou reprezentovány hodnotou, nebo rozsahem fyzické veličiny.



Binární digitální signál

Decimální numerický systém

- Základ (nebo také radix)
  - 10 číslic { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
- Pozice
  - Celá čísla nebo zlomky
- Váhy
  - Váha = (*základ*) ^{pozice}
- Důležitost
  - Součet "*číslice* x *váha*"
- Formální notace:

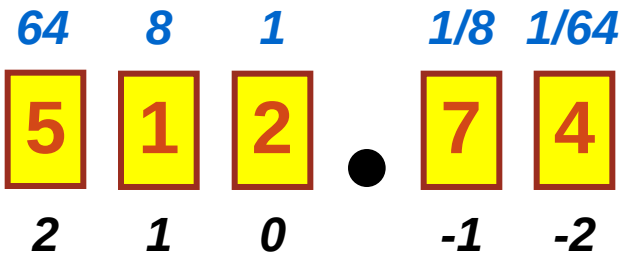


$$d_2*B^2+d_1*B^1+d_0*B^0+d_{-1}*B^{-1}+d_{-2}*B^{-2}$$

$$(512.74)_{10}$$

Oktalový numerický systém

- Základ = 8
  - 8 číslic { 0, 1, 2, 3, 4, 5, 6, 7 }
- Váhy
  - Váha = (*základ*) ^{pozice}
- Důležitost
  - Součet "*číslice* x *váhat*"
- Formální notace:



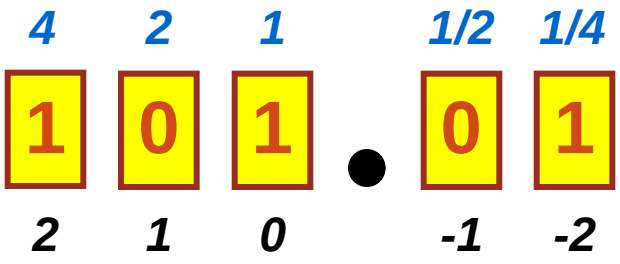
$5_2 * 8^2 + 1 * 8^1 + 2 * 8^0 + 7 * 8^{-1} + 4 * 8^{-}$

$= (330.9375)_{10}$

$(512.74)_8$

Binární numerický systém

- Základ = 2
  - 2 číslice { 0, 1 }, zvané "*bity*"
- Váhy
  - Váha = (*základ*) ^{pozice}
- Důležitost
  - Součet "*bitu* x *váhy*"
- Formální notace:
- Skupina bitů: 8 bitů = *Byte*



$1*2^2+0*2^1+1*2^0+0*2^{-1}+1*2^{-2}$

$=(5.25)_{10}$

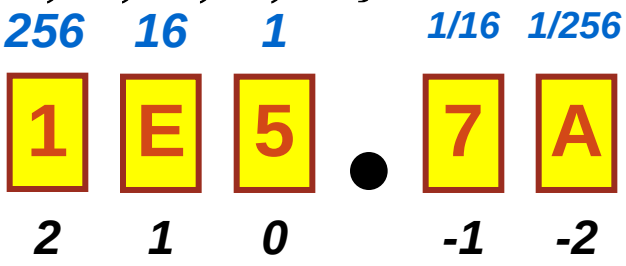
$(101.01)_2$

1 0 1 1

1 1 0 0 0 1 0 1

Hexadecimální numerický systém

- Základ = 16
  - 16 číslic { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }
- Váhy
  - Váha = (*základ*) ^{pozice}
- Důležitost
  - Součet "*číslic* x *váha*"
- Formální notace:



$1*16^{2}+14*16^{1}+5*16^{0}+7*16^{-1}+10*16^{-2}$

$=(485.4765625)_{10}$

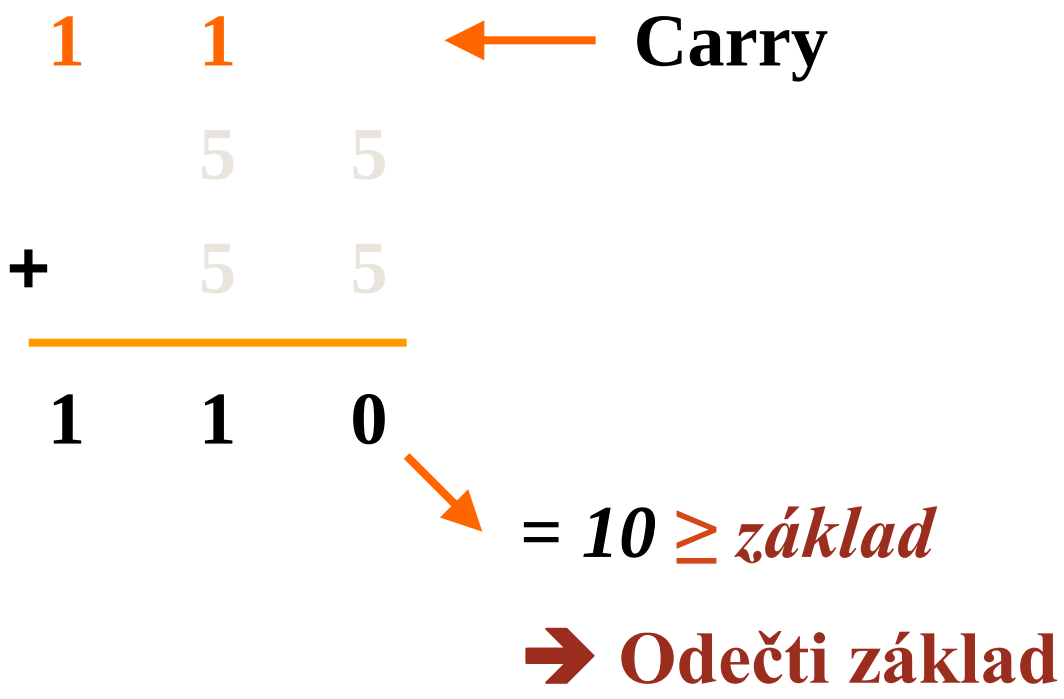
$(1E5.7A)_{16}$

n	2^n
0	$2^0=1$
1	$2^1=2$
2	$2^2=4$
3	$2^3=8$
4	$2^4=16$
5	$2^5=32$
6	$2^6=64$
7	$2^7=128$

n	2^n	
8	$2^8=256$	
9	$2^9=512$	
10	$2^{10}=1024$	Kilo
11	$2^{11}=2048$	
12	$2^{12}=4096$	
20	$2^{20}=1M$	Mega
30	$2^{30}=1G$	Giga
40	$2^{40}=1T$	Tera

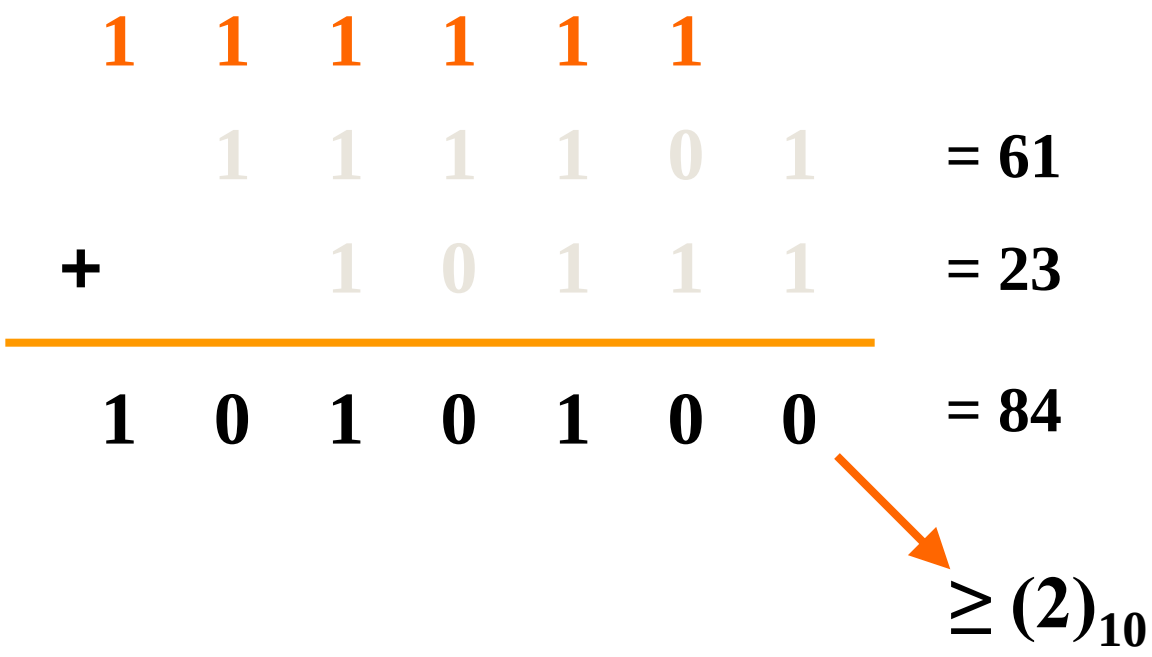
Sčítání

- Decimální sčítání



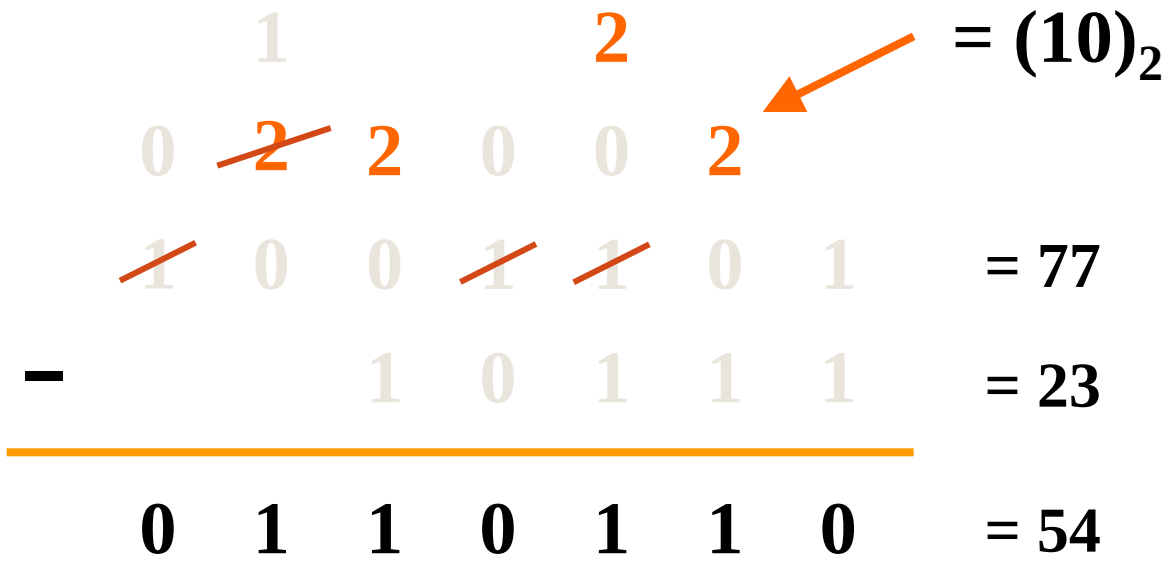
Binární sčítání

- Sčítání sloupců



Binární odčítání

- Zapůjčit si základ když je třeba

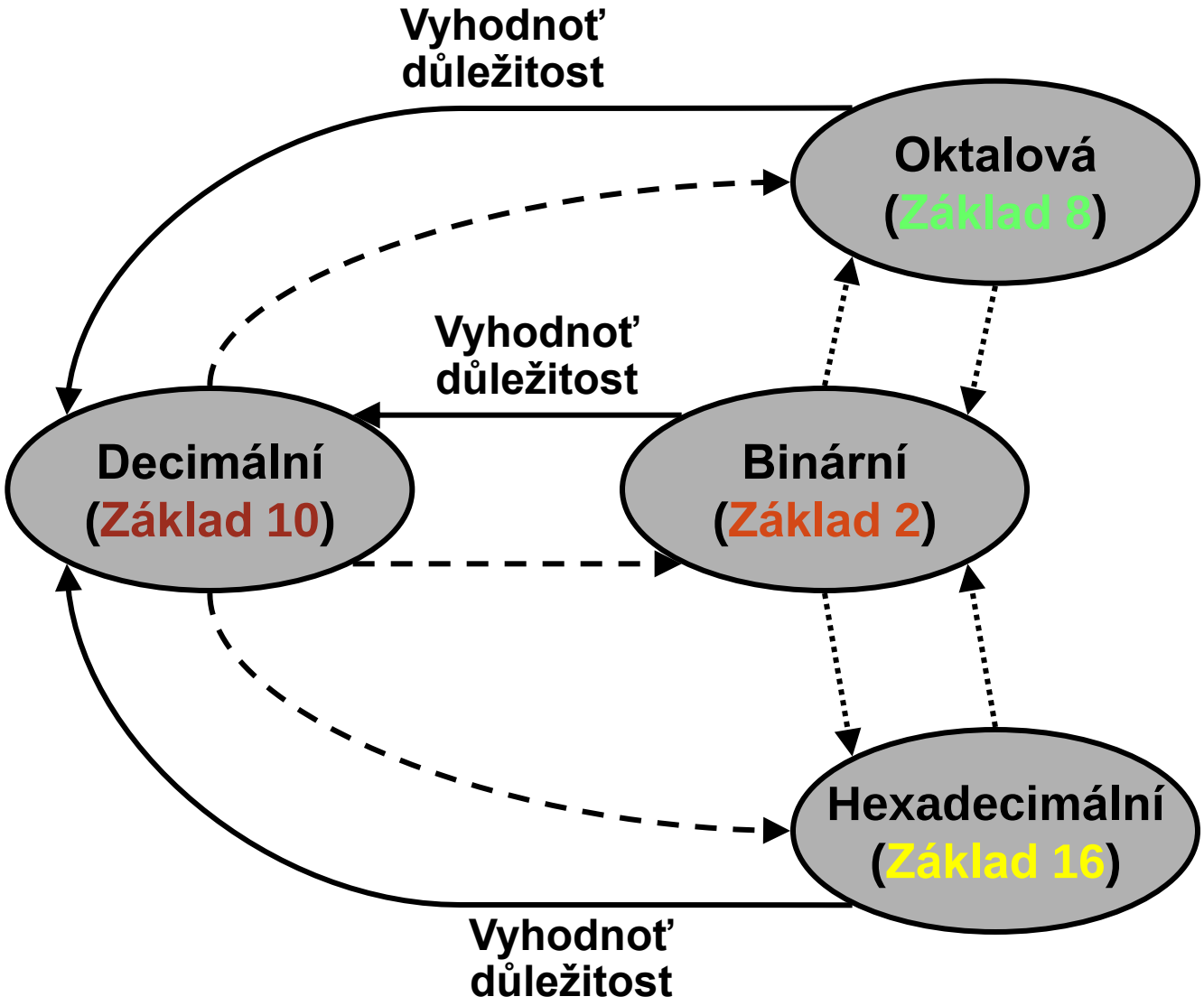


Binární násobení

- Bit po bitu

			1	0	1	1	1
x				1	0	1	0
			0	0	0	0	0
		1	0	1	1	1	
	0	0	0	0	0		
1	0	1	1	1			
1	1	1	0	0	1	1	0

Převody soustav (numerických systémů)



Z decimální (celé číslo) do binární soustavy

- Vyděl číslo základem (=2)
- Zapíšeme zbytek
- Vezmeme si pozůstatek a opakujeme krok

Příklad: $(13)_{10}$

	Podíl	Zbytek	Výsledná hodnota
13 / 2 =	6	1	$a_0 = 1$
6 / 2 =	3	0	$a_1 = 0$
3 / 2 =	1	1	$a_2 = 1$
1 / 2 =	0	1	$a_3 = 1$

Výsledek: $(13)_{10} = (a_3\, a_2\, a_1\, a_0)_2 = (1101)_2$

Z decimální (celé číslo) do binární soustavy

- Vynásobíme číslo základem (=2)
- Vezmeme celé číslo (buď 0 nebo 1) jako koeficient
- Vezmeme výsledný zlomek a opakujeme

Příklad: $(0.625)_{10}$

		Celé číslo		Zlomek	Výsledná hodnota
0.625	* 2 =	1	.	25	$a_{-1} = 1$
0.25	* 2 =	0	.	5	$a_{-2} = 0$
0.5	* 2 =	1	.	0	$a_{-3} = 1$

Výsledek: $(0.625)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_2 = (0.101)_2$

Z decimální do osmičkové soustavy

Příklad: $(175)_{10}$

	Podíl	Zbytek	Výsledná hodnota
175 / 8 =	21	7	$a_0 = 7$
21 / 8 =	2	5	$a_1 = 5$
2 / 8 =	0	2	$a_2 = 2$

Výsledek: $(175)_{10} = (a_2\, a_1\, a_0)_8 = (257)_8$

Příklad: $(0.3125)_{10}$

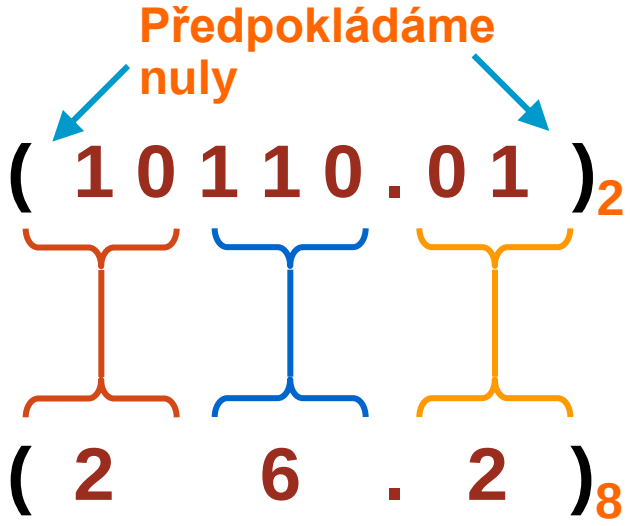
	Celé číslo		Zlomek	Výsledná hodnota
0.3125 * 8 =	2	.	5	$a_{-1} = 2$
0.5 * 8 =	4	.	0	$a_{-2} = 4$

Výsledek: $(0.3125)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_8 = (0.24)_8$

Z binární do oktalové soustavy

- $8 = 2^3$
- Každá skupina tří bitů reprezentuje oktalové číslo

Příklad:



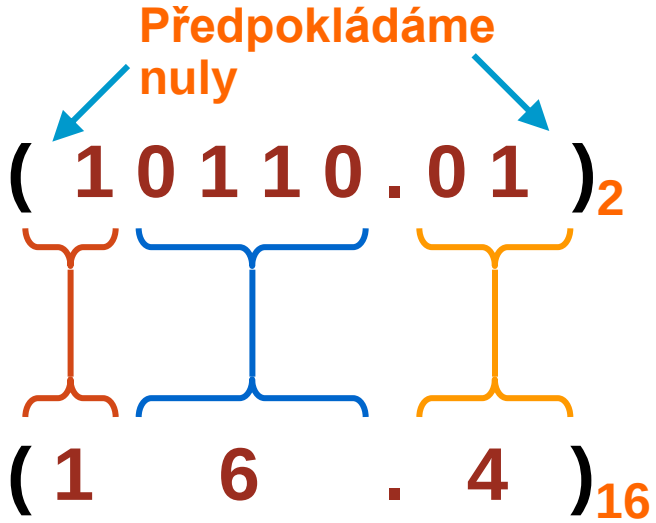
Octalová	Binární
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Funguje oběma směry (*Z binární do oktalové a naopak)*

Z binární do hexadecimální soustavy

- $16 = 2^4$
- Každá skupina čtyř bitů reprezentuje jednu hexadecimální hodnotu

Příklad:

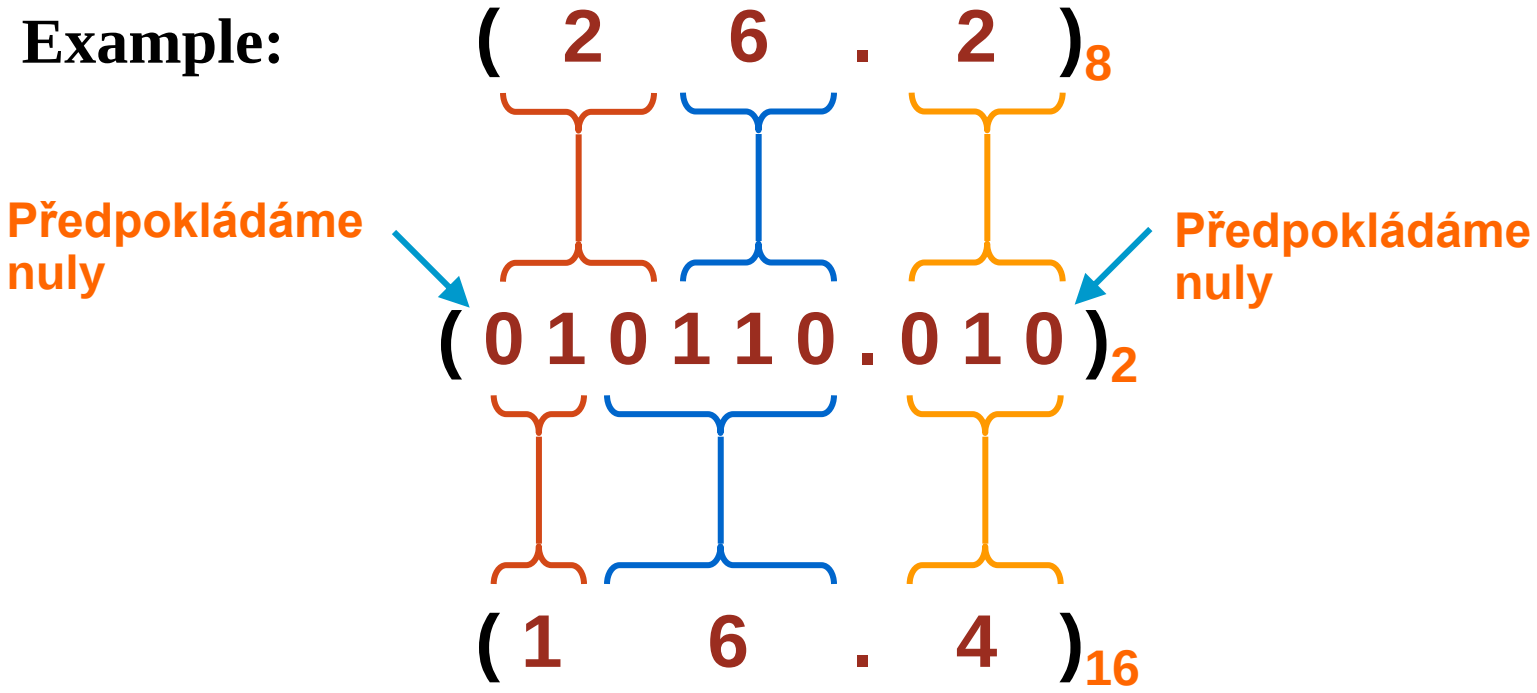


Hexadec.	Binární
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

Funguje opět oběma směry

Z oktalové do hexadecimální soustavy

- Nejdříve převod do binární soustavy



Funguje opět oběma směry

Decimální	Binární	Oktalová	Hexadec.
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.5 Doplňky

- Jsou dva typy doplňků pro každou soustavu-r
- **Doplněk - (r-1)**
  - Danému číslu *N* o základu *r* obsahující *n* číslic, (*r–1*) *doplněk čísla N* je definován jako:

$$(r^n - 1) - N$$

- **Příklad pro 6-místné decimální číslo**:
  - $(r^n - 1) - N = (10^6 - 1) - N = 999999 - N$
  - 546700 je 999999–546700 = 453299
- **Příklad pro 7-místné decimální číslo**:
  - $(r^n - 1) - N = (2^7 - 1) - N = 1111111 - N$
  - 1011000 je 1111111–1011000 = 0100111

1.6 Znaménková binární čísla

- K reprezentaci negativních čísel potřebujeme notaci pro negativní hodnoty
- Nejobvyklejší způsob je umístění znaménka na první pozici z leva
- Je domluveno, že když tzv. sign bit 0 je pro pozitivní a 1 pro negativní.

decimální	2. doplněk	1. doplněk	znaménková hodnota
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Znaménková binární čísla

- Aritmetický součet
  - Součet dvou čísel se řídí pravidly normální aritmetiky. **Pokud jsou znaménka stejná**, sečteme tyto hodnoty a přiřadíme toto znaménko. **Pokud jsou rozdílná**, odečteme od sebe hodnoty a zapíšeme znaménko převahující hodnoty.
- Příklad:

+ 6	00000110	– 6	11111010
+13	00001101	+13	00001101
+ 19	00010011	+ 7	00000111
+ 6	00000110	– 6	11111010
–13	11110011	–13	11110011
– 7	11111001	– 19	11101101

Znaménková binární čísla

- Aritmetické odčítání
  - Formou 2. doplňku:

1. Vezmeme druhý doplněk menšitele (včetně sign bitu) a přičteme ho k menšenci (včetně sign bitu).
2. Carry bit vypustíme.

$$(\pm A) - (+B) = (\pm A) + (-B)$$
$$(\pm A) - (-B) = (\pm A) + (+B)$$

- Příklad:

(– 6) – (– 13) → (11111010 – 11110011)

→ (11111010 + 00001101)

→ 00000111 (+ 7)

1.7 Birnání kódy

- BCD kód
  - Číslo s k decimálními číslicemi bude potřebovat 4k bitů v BCD.
  - Decimálně 396 je reprezentováno v BCD 12-cti bity jako 0011 1001 0110, kde každá skupina bitů reprezentuje jednu decimální číslici.
  - Decimální číslo v BCD je stejné jako jeho ekvivalent v binární soustavě pokud je v rozsahu 0-9.
  - Binární kombinace1010 k 1111 nemá smysl v BCD.

Table 1.4
Binary-Coded Decimal (BCD)

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Binární kódy

- Příklad:
  - Decimální číslo185 a jeho hodnota v BCD a dvojkové sosutavě:

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

- BCD sčítání

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	+1001
9	1001	12	1100	17	10001
			+0110		+0110
			10010		10111

Binární kódy

- Příklad:
  - Součet184 + 576 = 760 v BCD:

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binární součet	0111	10000	1010	
+6		0110	0110	
Součet BCD	0111	0110	0000	760

- (+375) + (-240) = +135

0	375
+9	760
0	135

Binární kódy

- (ASCII) tabulka znaků

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Binární kódy

- Error-Detecting kód
  - Redundance (nadbytečná informace), ve formě bitu navíc, můžu být vložena do binárního kódu pro detekci erroru.
  - Jednoduchá forma redundance je parita, bit navíc přiložený do kódu k označení sudého a lichého čísla.
  - Kód má sudou paritu pokud číslo v kódu je sudé
  - Kód má lichou paritu pokud číslo v kódu je liché
  - Příklad:

Zpráva A:	100010011	(sudá parita)
Zpráva B:	100010010	(lichá parita)

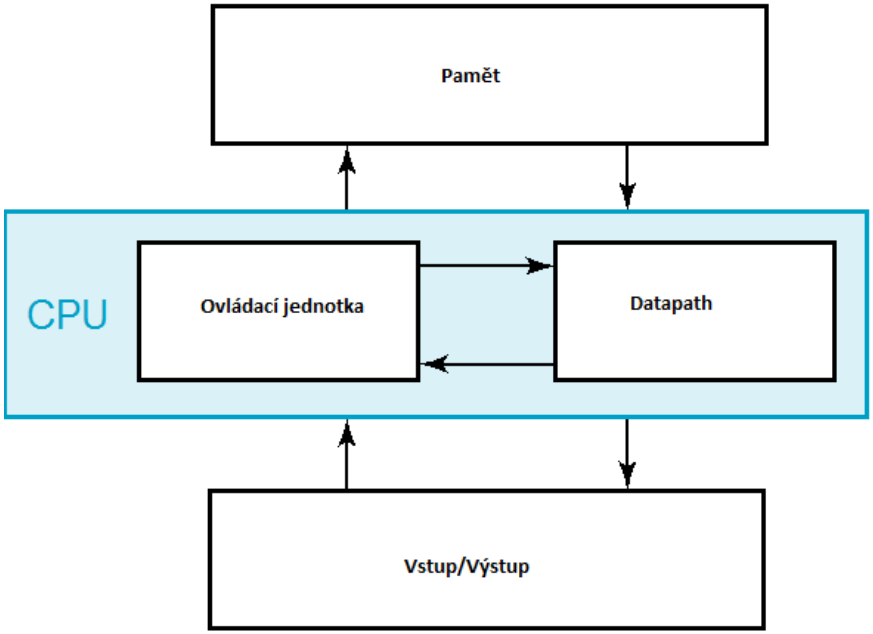
1.8 Binární uložiště a registry

- Registry
  - Binární buňka je zařízení schopné uchovat vždy jeden stav ze dvou.
  - Registr je skupina těchto buněk. Registru o n-binárních jednotech může uchovat informaci o velikosti n-bitů.

n buněk  2^n možných stavů

- Binární buňka
  - Dva stabilní stavy
  - Skladuje jeden bit informace
- Registr
  - Skupina binárních buněk
- Přenos registrů
  - Přenos informací z jednoho registru do druhého

Příklad digitálního počítače

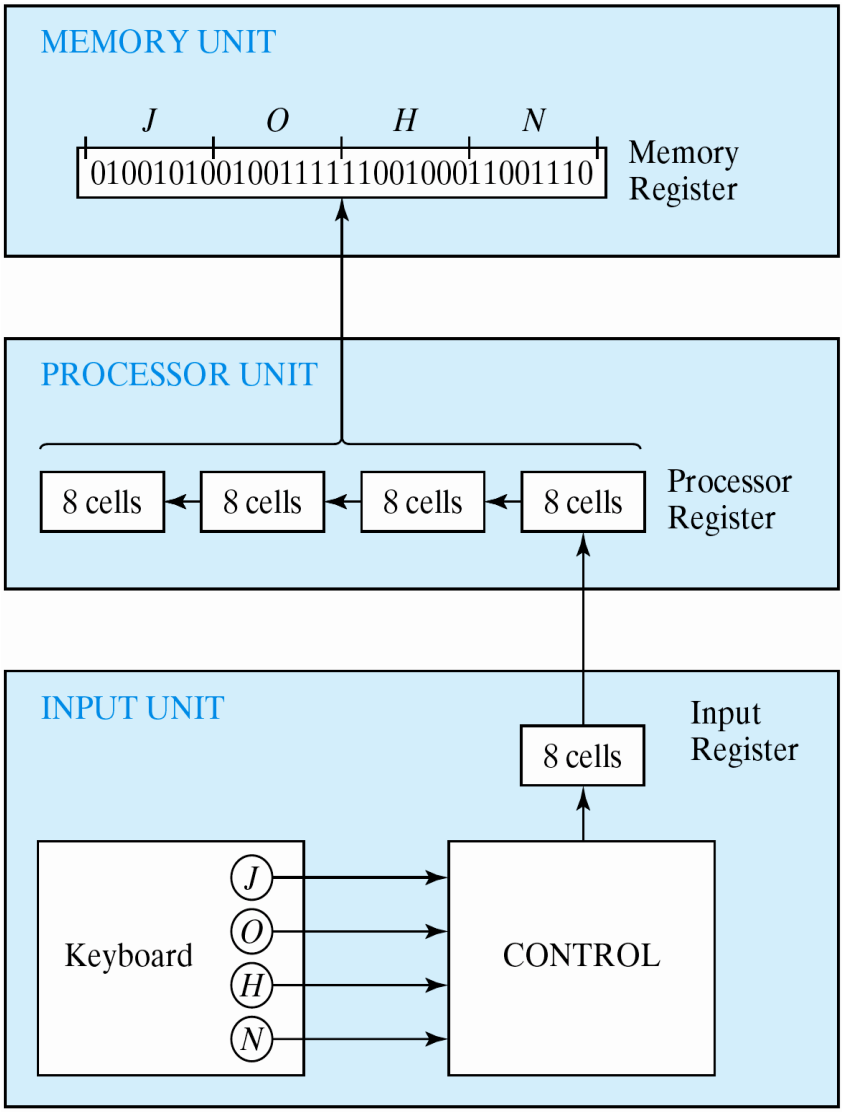


Vstup: Klávesnice, myš, modem, mikrofón

Výstup: CRT, LCD, modem, reproduktory

Synchronní nebo Asynchronní?

Přenos informací



1.9 Binární logika

- Definice binární logiky
  - Binární logika se skládá z binárních proměnných a setu binárních operací.
  - Proměné jsou označený písmeny abecedy, jako *A*, *B*, *C*, *x*, *y*, *z*, atd., každá proměná může nabývat pouze dvou hodnot: 1 a 0,
  - Jsou tři základní logické operace: AND, OR, a NOT.

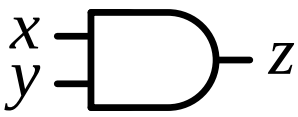
Binární logika

- Pravdivostní tabulky, Boolenovský tvar a logická hradla

AND

x	*y*	*z*
0	0	0
0	1	0
1	0	0
1	1	1

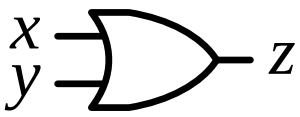
$z = x \bullet y = x\,y$



OR

x	*y*	*z*
0	0	0
0	1	1
1	0	1
1	1	1

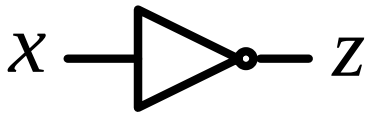
$z = x + y$



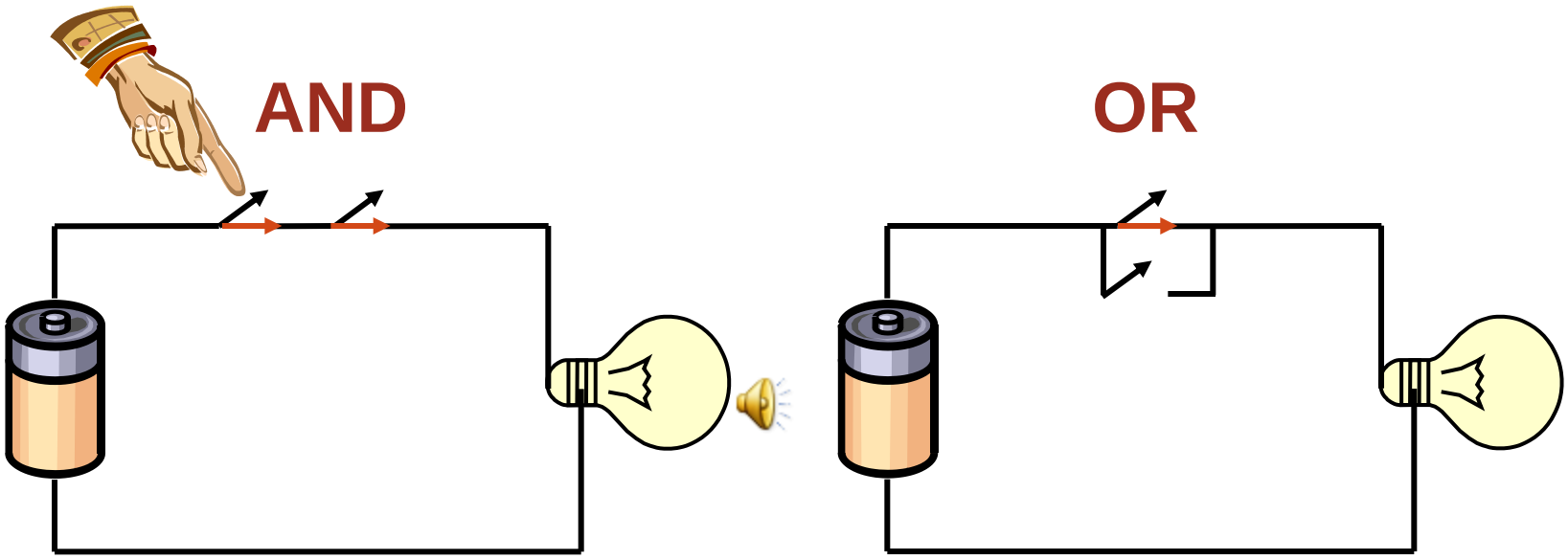
NOT

x	*z*
0	1
1	0

$z = \overline{x} = x'$

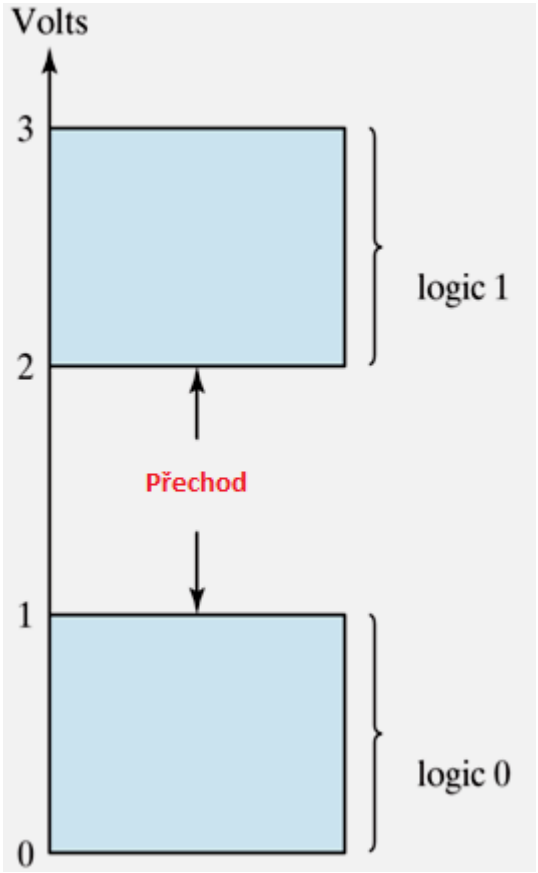


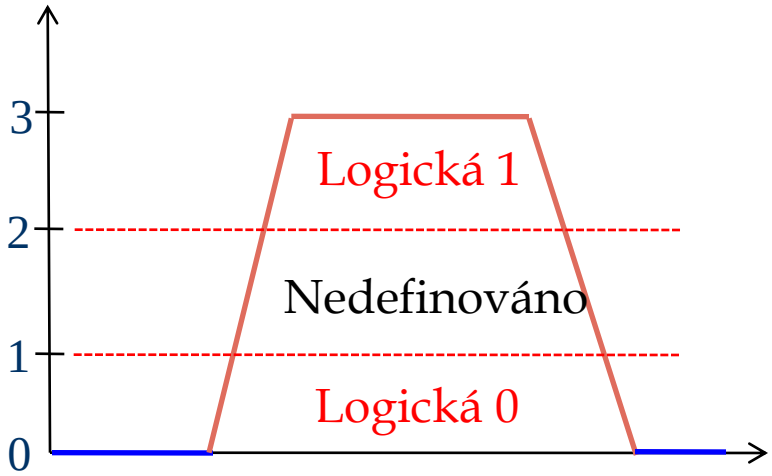
Spínací obvody



Binární logika

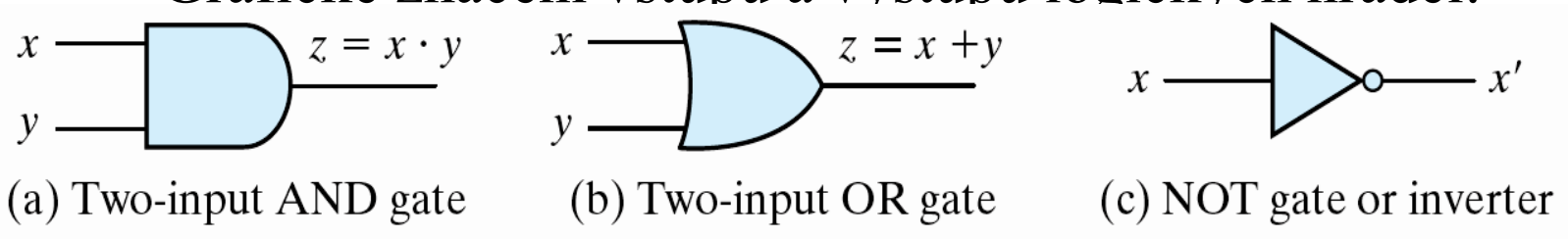
- Logická hradla
  - Příklad logických signálů



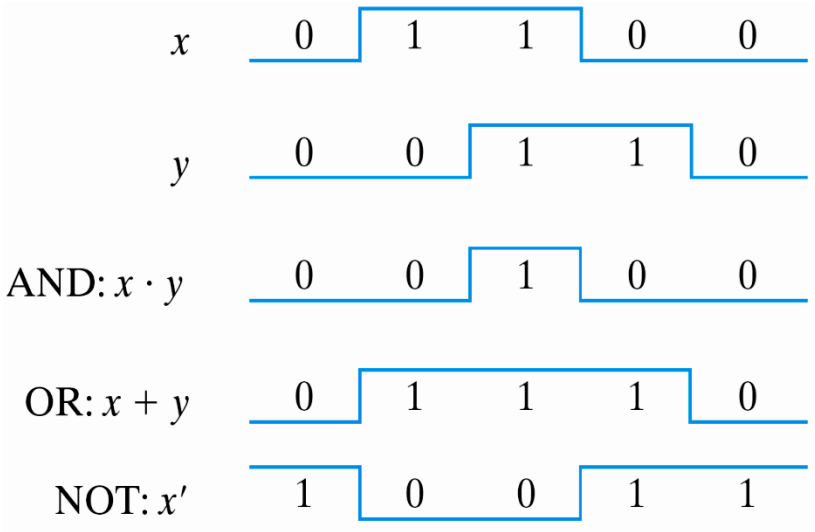


Binární logika

- Logická hradla
  - Grafické značení vstupů a výstupů logických hradel:

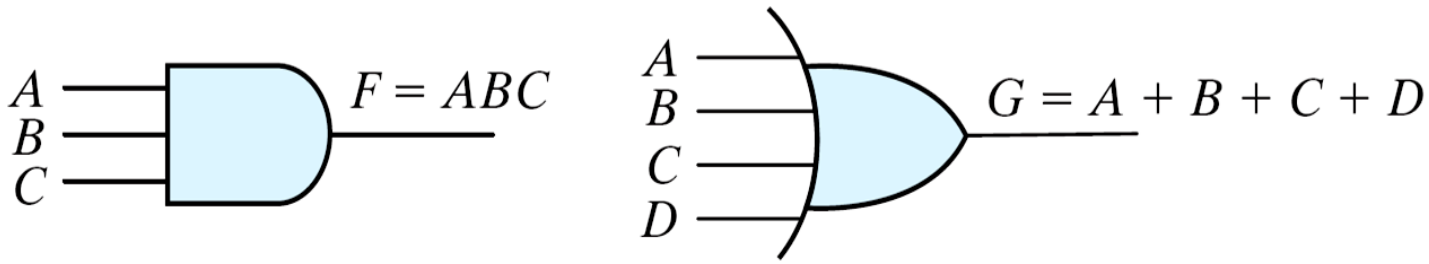


(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter



Binární logika

- Logická hradla
  - Grafické značení vstupů a výstupů logických hradel:



(a) Three-input AND gate (b) Four-input OR gate

Minimalizace

- Minimalizace odkazuje na rozvržení úkolů pro nalezení optimální implementace booleanovských funkcí.
- Optimální implementace můžeme dosáhnout hned několika způsoby

Minimalizace pomocí mapy

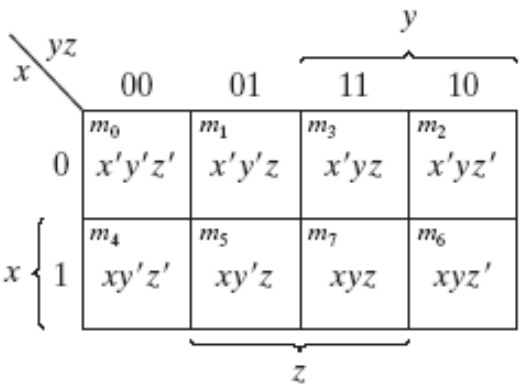
- Karnaughova mapa
  - Jednoduchá prodecura
  - Může mít grafickou formu pravdivostní tabulky
  - Nepraktická pro velké složité obvody
- Je to diagram tvořený čtvercy
  - Každý čtverec reprezentuje jeden minterm

Minimalizace mapy s třemi proměnými

- Mapa s třemi proměnými
  - osm mintermů
  - Jakékoliv přilehlé čtverce liší alespoň jednou
    - m_5 a m_7 může být zjednodušeno
    - $m_5 + m_7 = xy'z + xyz = xz\,(y'+y) = xz$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)



(b)

Minimalizace mapy s třemi proměnými

- m_0 a m_2 (m_4 and m_6) jsou přilehlé
- $m_0 + m_2 = x'y'z' + x'yz' = x'z'(y'+y) = x'z'$
- $m_4 + m_6 = xy'z' + xyz' = xz'(y'+y) = xz'$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)

x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

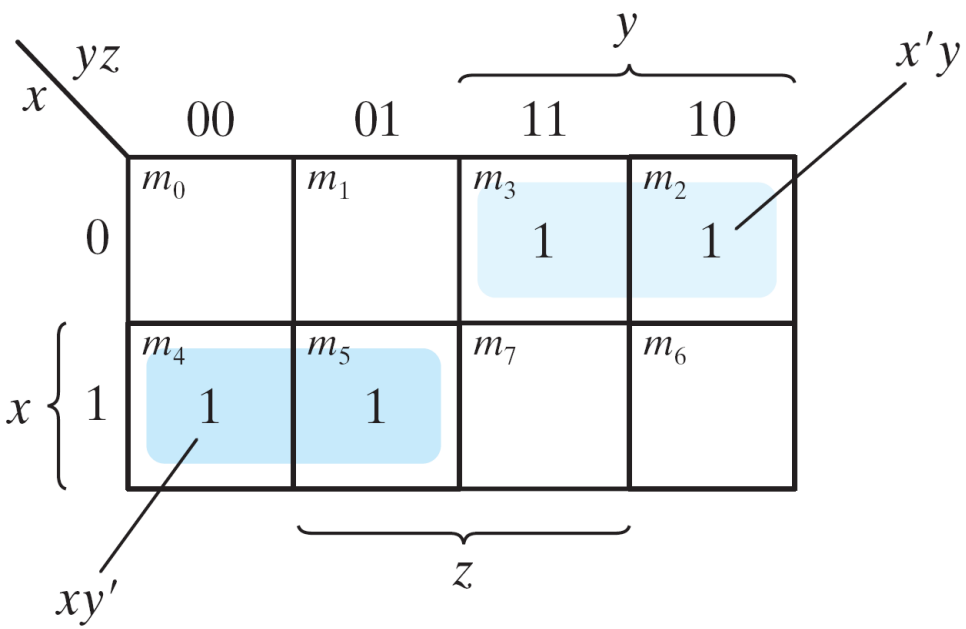
y (11, 10); z (01, 11); x (1)

(b)

Fig. 3-3 Three-variable Map

Minimalizace příklad 1

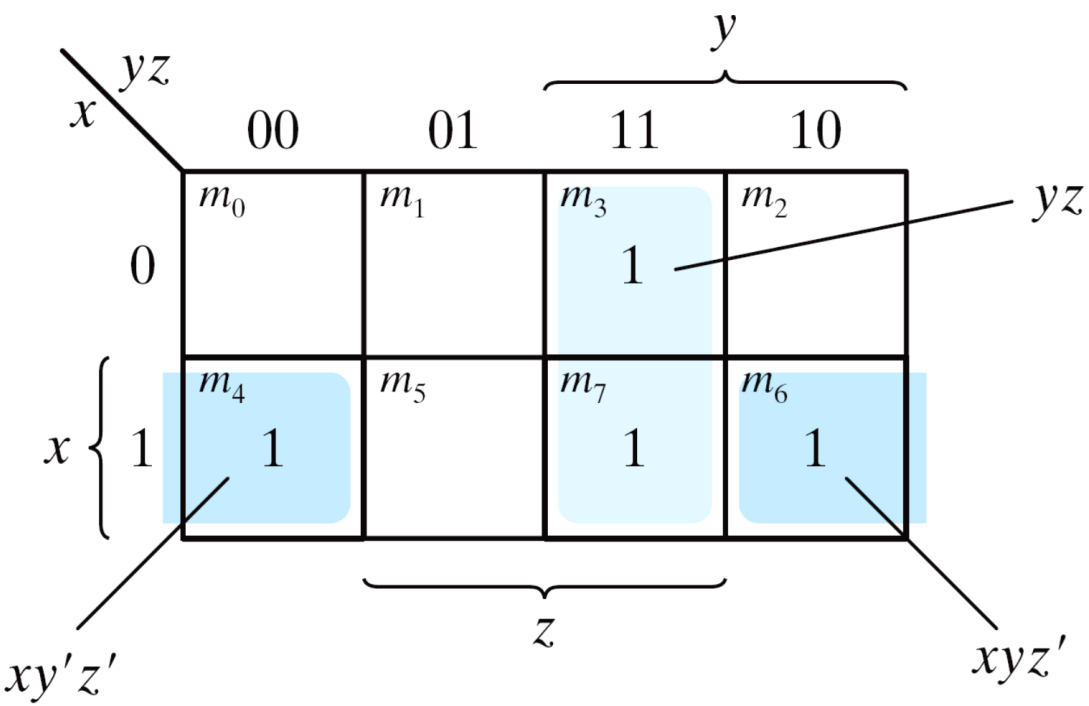
- Zjednodušte funkci $F(x, y, z) = \Sigma(2, 3, 4, 5)$
  - $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$



F(x, y, z) = Σ(2, 3, 4, 5) = x'y + xy'

Minimalizace příklad 2

- Zjednodušte $F(x, y, z) = \Sigma(3, 4, 6, 7)$
  - $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$

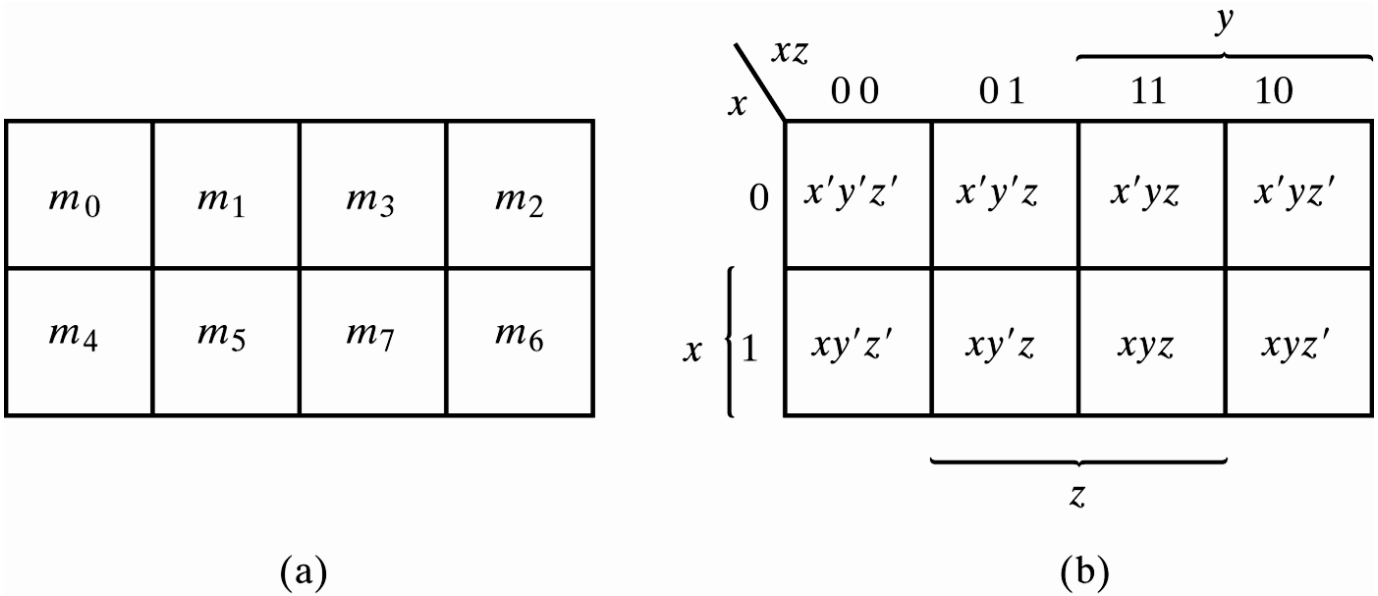


Note: $xy'z' + xyz' = xz'$

F(x, y, z) = Σ(3, 4, 6, 7) = yz + xz'

Čtyři přilehlé čtverce

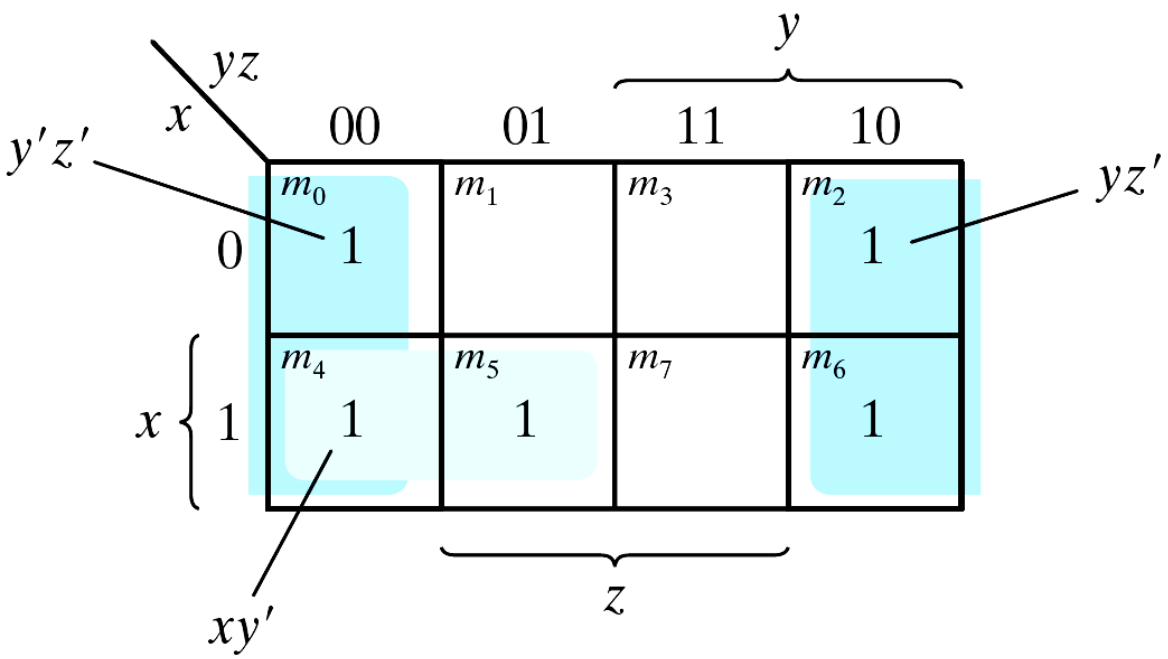
- Čtyři přilehlé čtverce 2, 4, and 8 squares
  - $m_0+m_2+m_4+m_6 = x'y'z'+x'yz'+xy'z'+xyz' = x'z'(y'+y) +xz'(y'+y) = x'z' + xz' = z'$
  - $m_1+m_3+m_5+m_7 = x'y'z+x'yz+xy'z+xyz = x'z(y'+y) + xz(y'+y) = x'z + xz = z$



Minimalizace příklad 3

Zjednodušte $F(x, y, z) = S(0, 2, 4, 5, 6)$

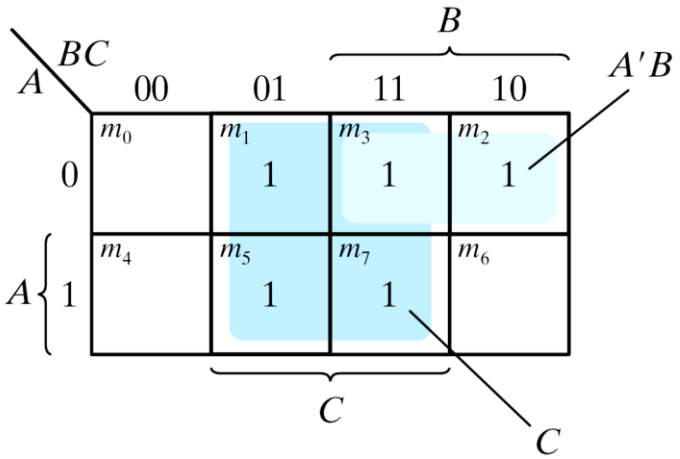
- $F(x, y, z) = S(0, 2, 4, 5, 6) = z' + xy'$



F(x, y, z) = Σ(0, 2, 4, 5, 6) = z' +xy'

Minimalizace příklad 4

- Nechť $F = A'C + A'B + AB'C + BC$
  - Vyjádři součtem mintermů.
  - Najdi optimální tvar.
    - $F(A, B, C) = \mathrm{S}(1, 2, 3, 5, 7) = C + A'B$



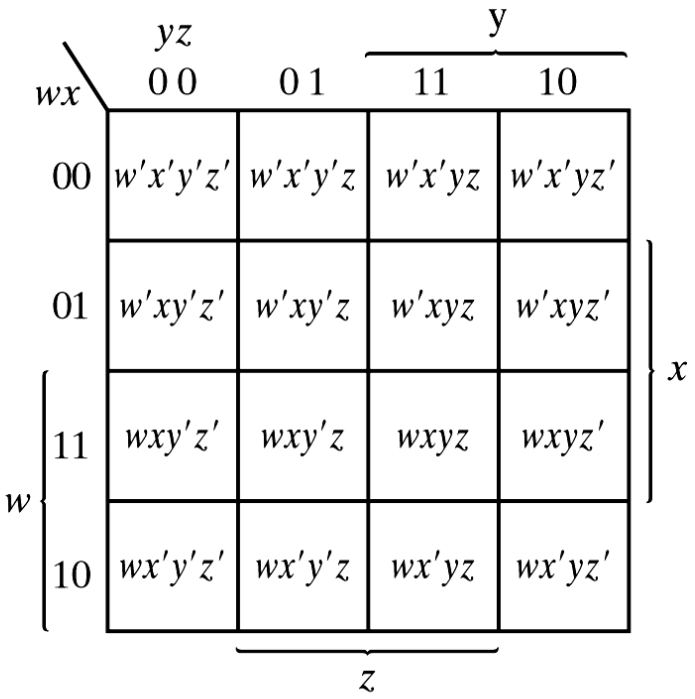
$A'C + A'B + AB'C + BC = C + A'B$

Minimalizace mapy se čtyřmi proměnými

- 16 mintermů
- Kombinace 2, 4, 8, a 16 přilehlých čtverců

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

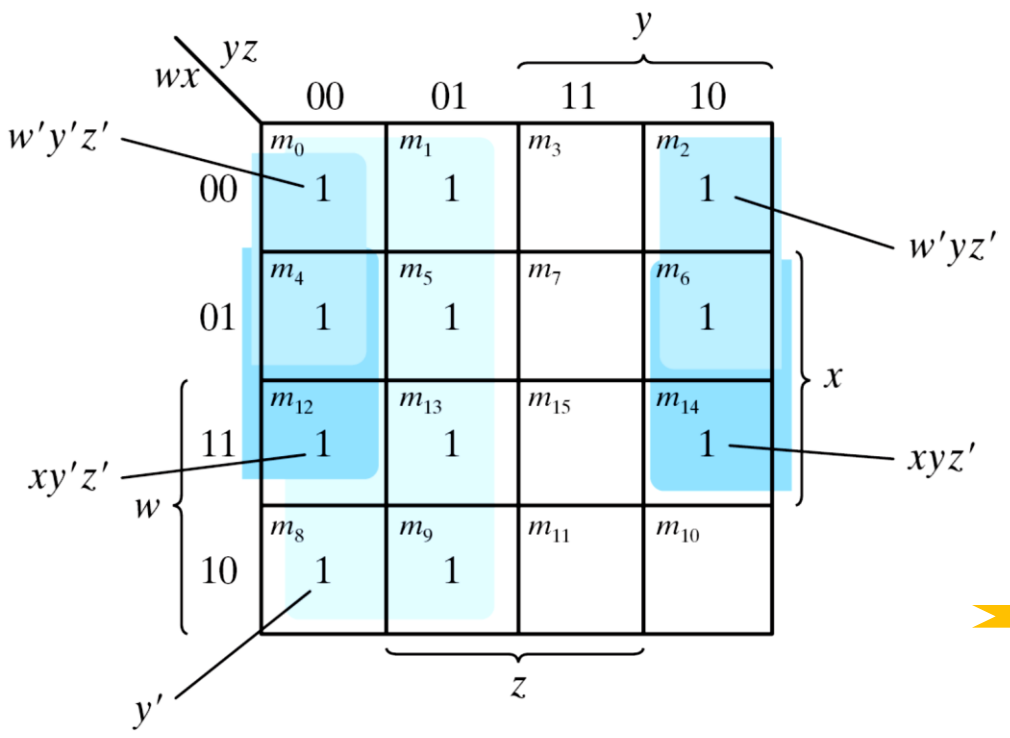
(a)



(b)

Minimalizace příklad 5

- Zjednodušte $F(w, x, y, z) = S(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$



F = y'+w'z'+xz'

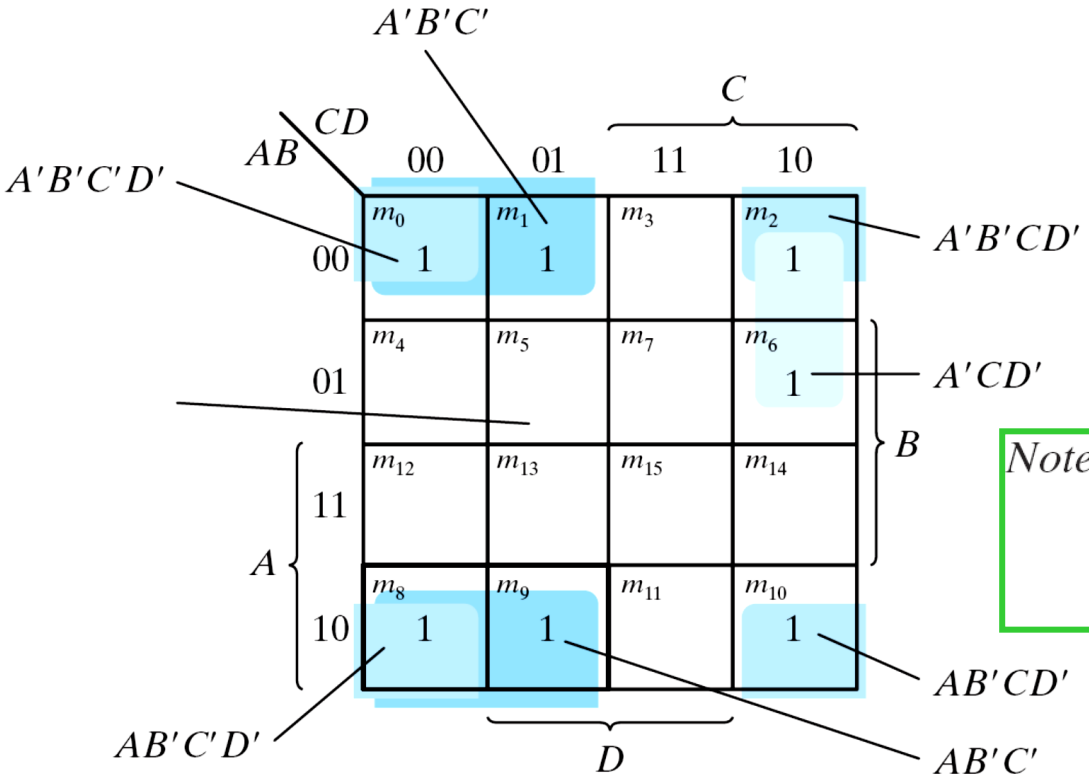
Note: $w'y'z' + w'yz' = w'z'$

$xy'z' + xyz' = xz'$

F(w, x, y, z) = Σ(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w' z' +xz'

Minimalizace příklad 6

- Zjednodušte $F = A'B'C' + B'CD' + A'B'C'D' + AB'C'$

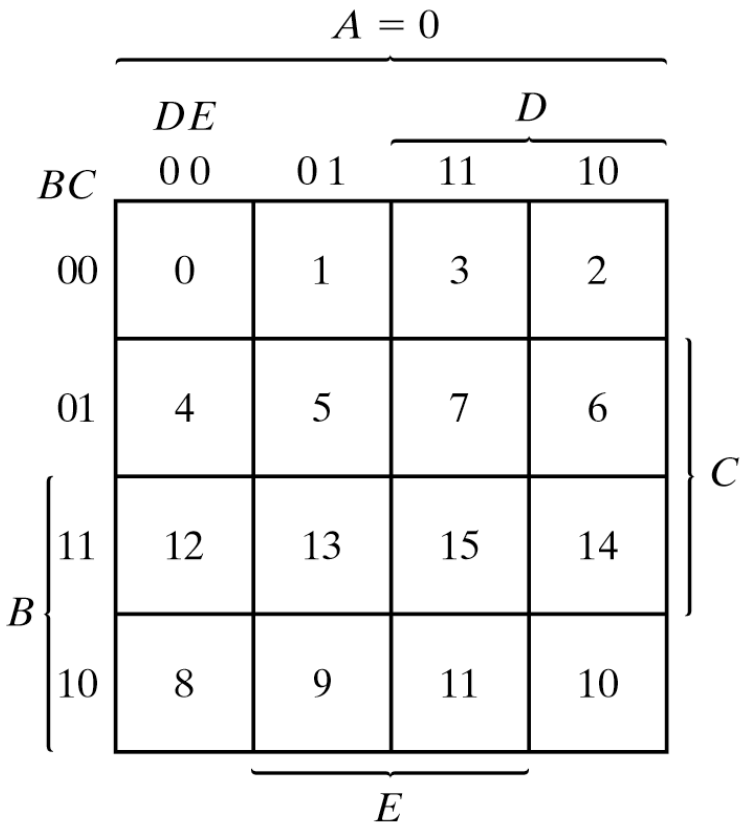


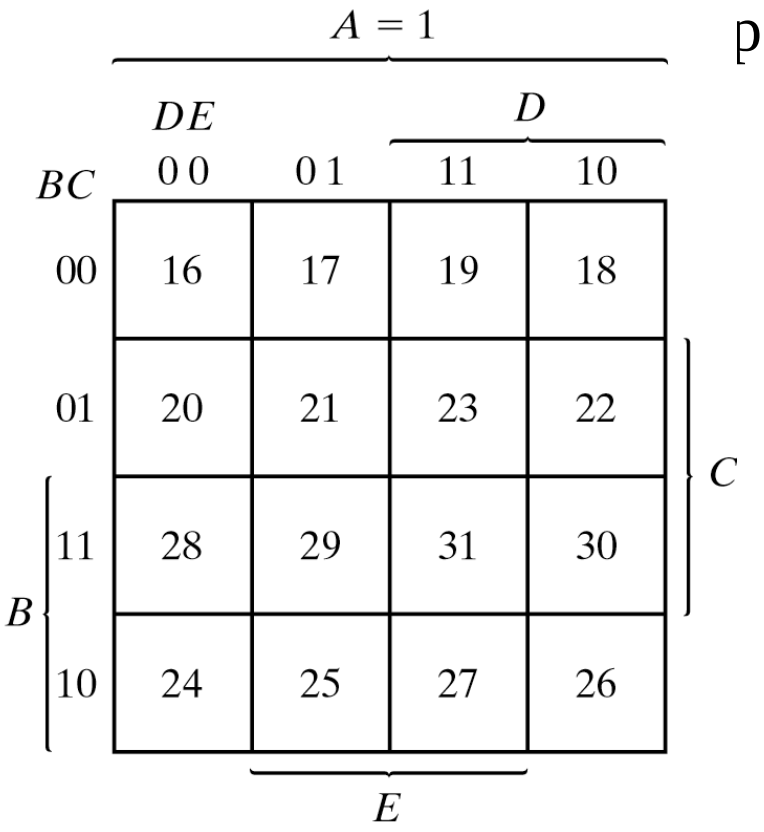
Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$
$A'B'C' + AB'C' = B'C'$

A′B′C′ + B′CD′ + A′B′C′D′ + AB′C′= B′D′ + B′C′ +A′CD′

Minimalizace mapy s pěti proměnnými

- Mapa s více jak čtyřmi proměnnými se stává komplikovanou





Minimalizace příklad 7

- Zjednodušte F = S(0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)

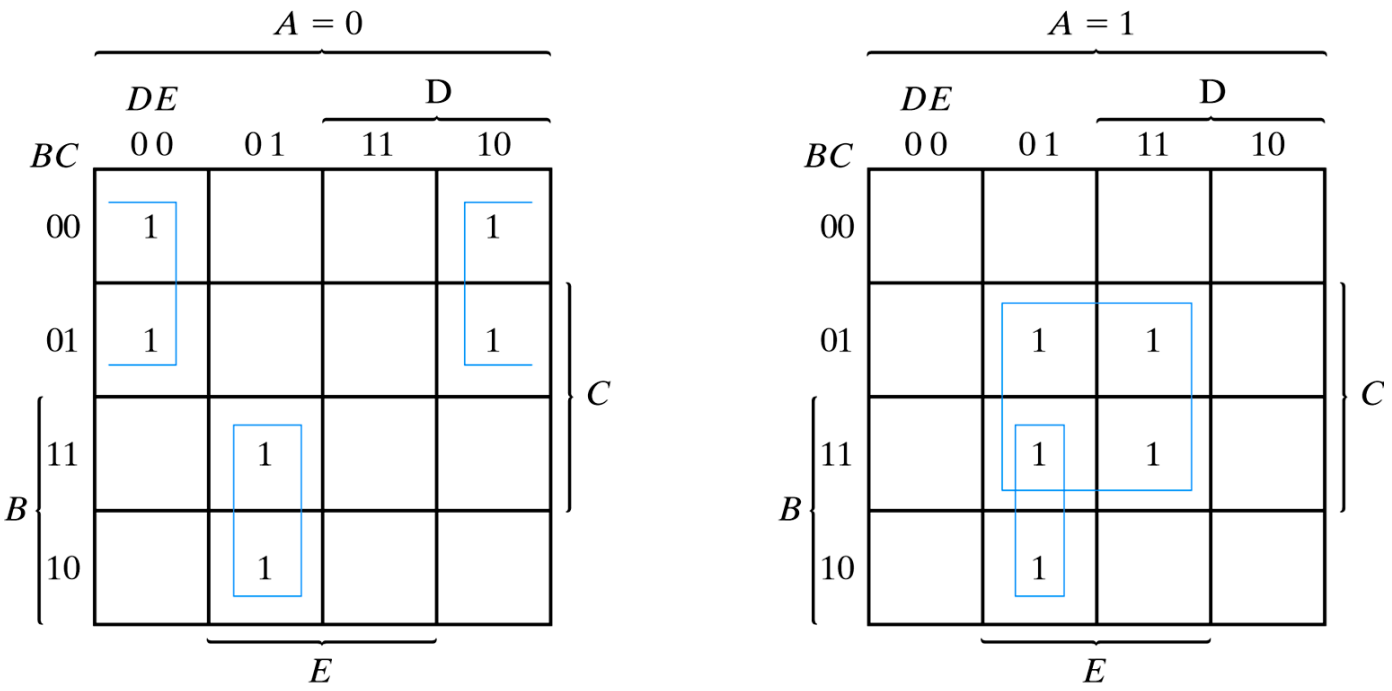
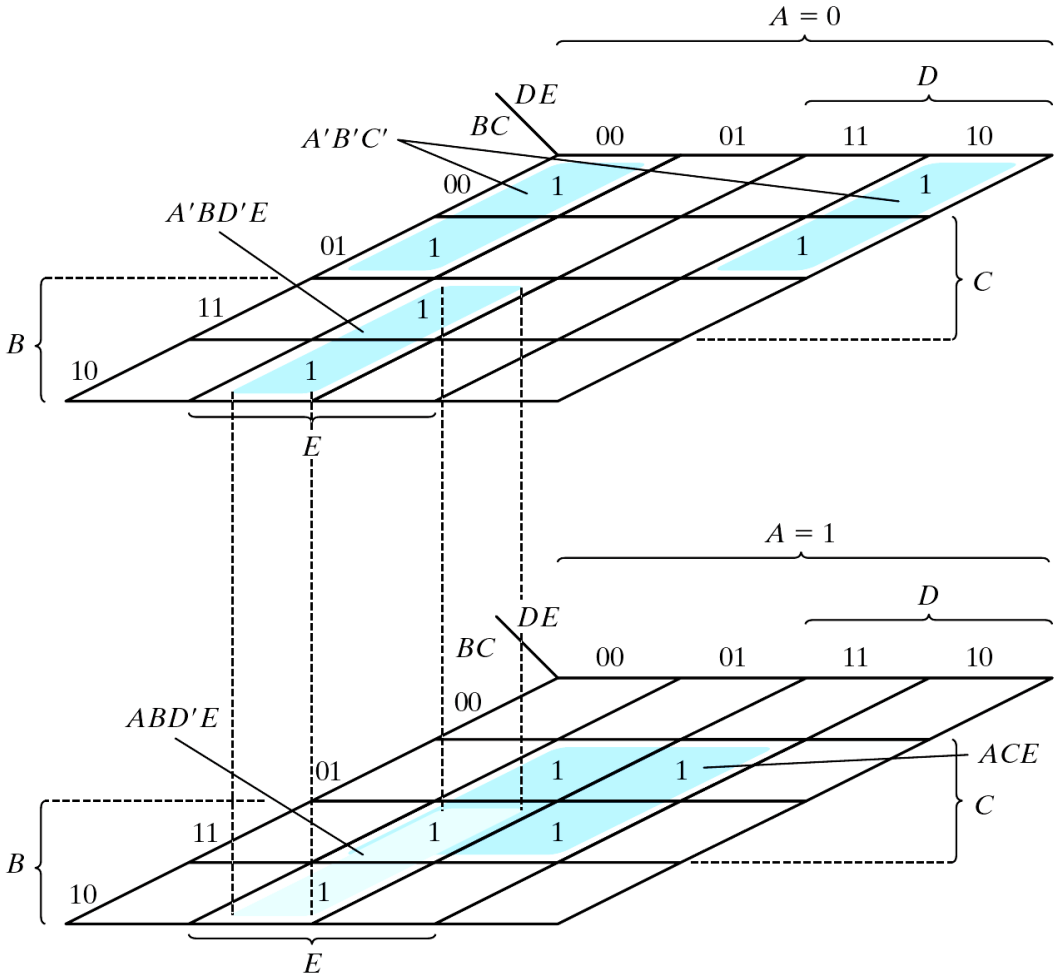


Fig. 3-13 Map for Example 3-7; $F = A'B'E' + BD'E + ACE$

F = A'B'E'+BD'E+ACE

Minimalizace příklad 7 (pokračování)



F = A'B'E'+BD'E+ACE

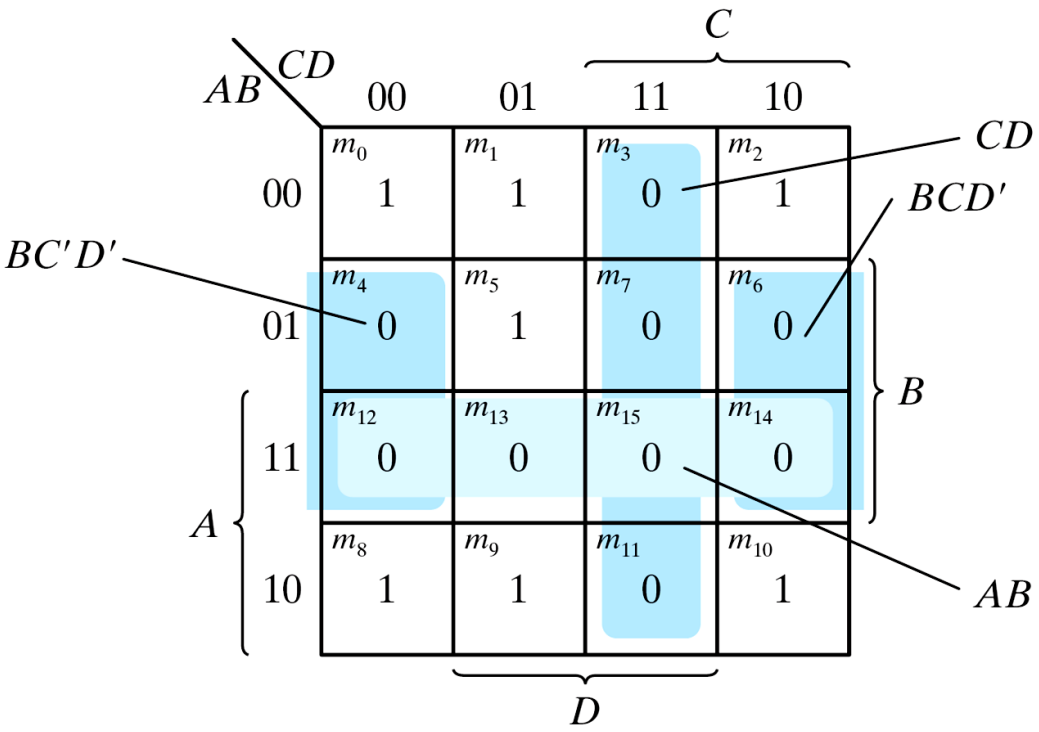
Produkt součtu zjednodušení

- Přístup#1
  - Zjednodušené F' ve formě součtu
  - Aplikujte DeMorganovy zákony $F = (F')'$
  - F': součet produktu $\rightarrow$ F: produkt sum
- Přístup #2: dualita
  - Kombinace maxtermů (bývaly to mintermy)
  - $M_0M_1 = (A+B+C+D)(A+B+C+D') = (A+B+C)+(DD') = A+B+C$

AB \ CD	00	01	11	10
00	M_0	M_1	M_3	M_2
01	M_4	M_5	M_7	M_6
11	M_{12}	M_{13}	M_{15}	M_{14}
10	M_8	M_9	M_{11}	M_{10}

Minimalizace příklad 8

Zjednodušte F = S(0, 1, 2, 5, 8, 9, 10) into (a) sum-of-products form, and (b) product-of-sums form:



Note: $BC'D' + BCD' = BD'$

F(A, B, C, D)= Σ(0, 1, 2, 5, 8, 9, 10) = B'D'+B'C'+A'C'D

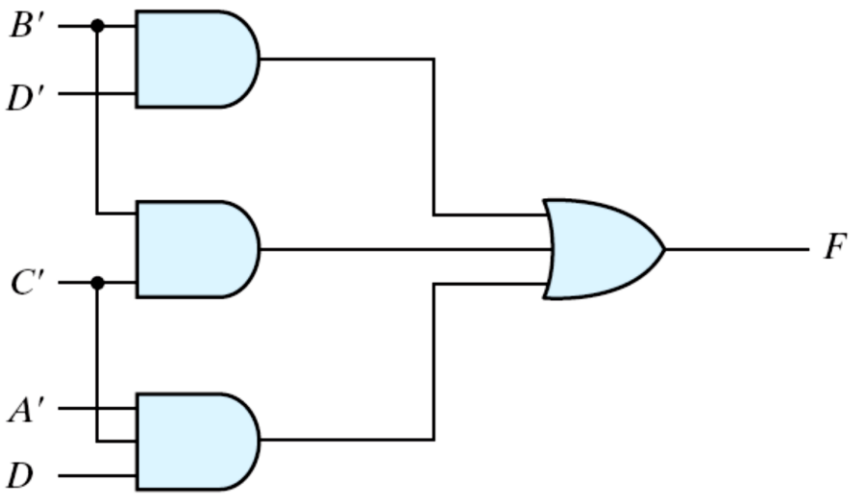
a) F(A, B, C, D)= Σ(0, 1, 2, 5, 8, 9, 10) = B'D'+B'C'+A'C'D

b) F' = AB+CD+BD'

» Aplikujte DeMorganovy zákony; F=(A'+B')(C'+D')(B'+D)

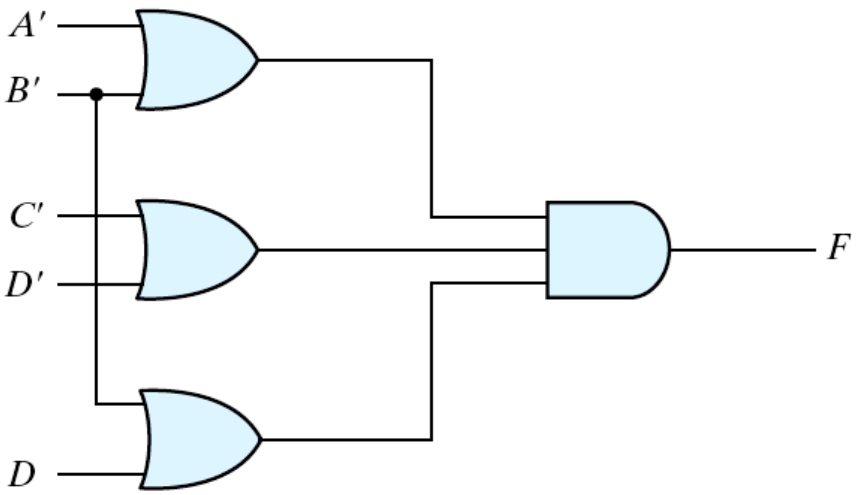
Minimalizace příklad 8 (pokračování)

- Implementace hradel pro funkci



(a) $F = B'D' + B'C' + A'C'D$

Forma součtu produktů



(b) $F = (A' + B')(C' + D')(B' + D)$

Forma produktu součtů

Procedura součtu mintermů

- Funkce z uvedené pravdivostní tabulky
  - V součtu mintermů:

$$F(x, y, z) = \sum(1, 3, 4, 6)$$

  - V součtu maxtermů:

$$F(x, y, z) = \Pi(0, 2, 5, 7)$$

  - Vzetí doplňku F′

$$F(x, y, z) = (x' + z')(x + z)$$

Table 3.2
Truth Table of Function F

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

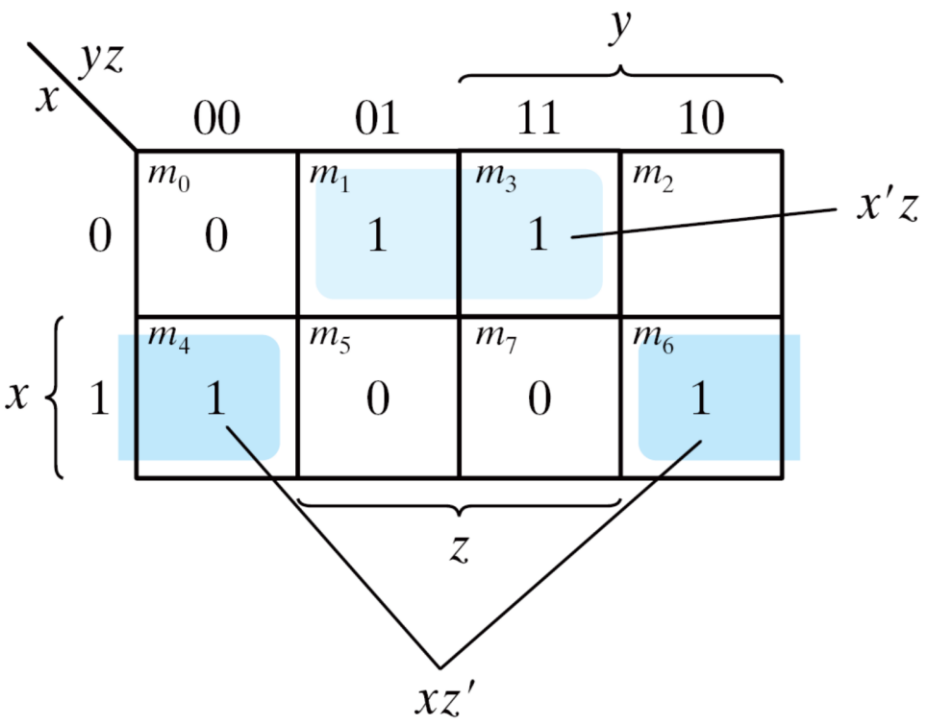
Procedura součtu mintermů

- Předpokládejme funkci vzatou z uvedené mapy
  - Kombinace jednotek:

$$F(x, y, z) = x'z + xz'$$

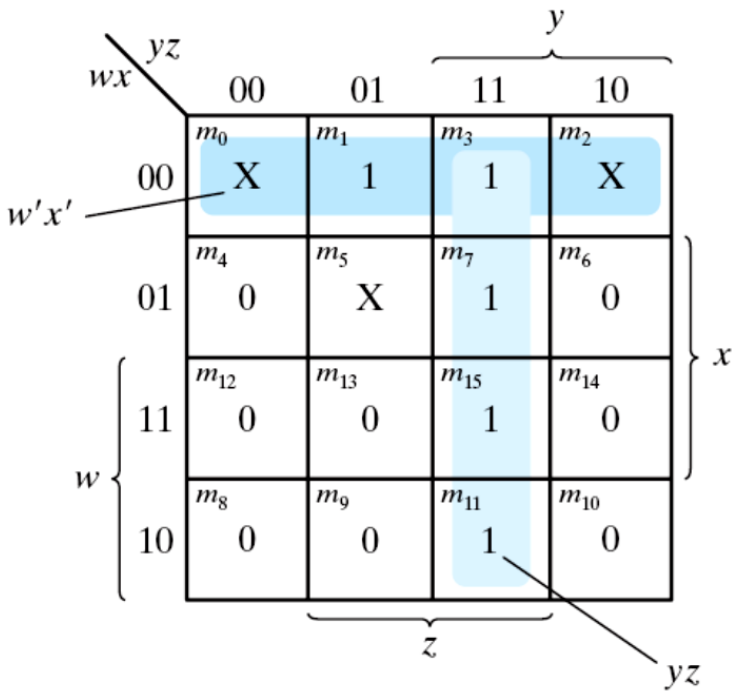
  - Kombinace nul:

$$F(x, y, z) = xz + x'z'$$

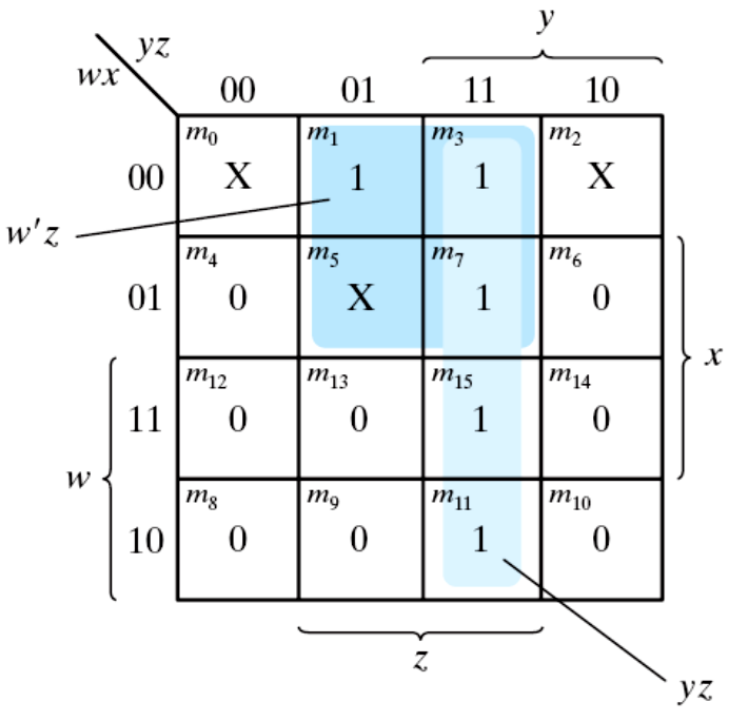


Minimalizace příklad 9 (pokračování)

- $F = yz + w'x'$; $F = yz + w'z$
- F = S(0, 1, 2, 3, 7, 11, 15) ; F = S(1, 3, 5, 7, 11, 15)
- Oba tvary jsou akceptovatelné



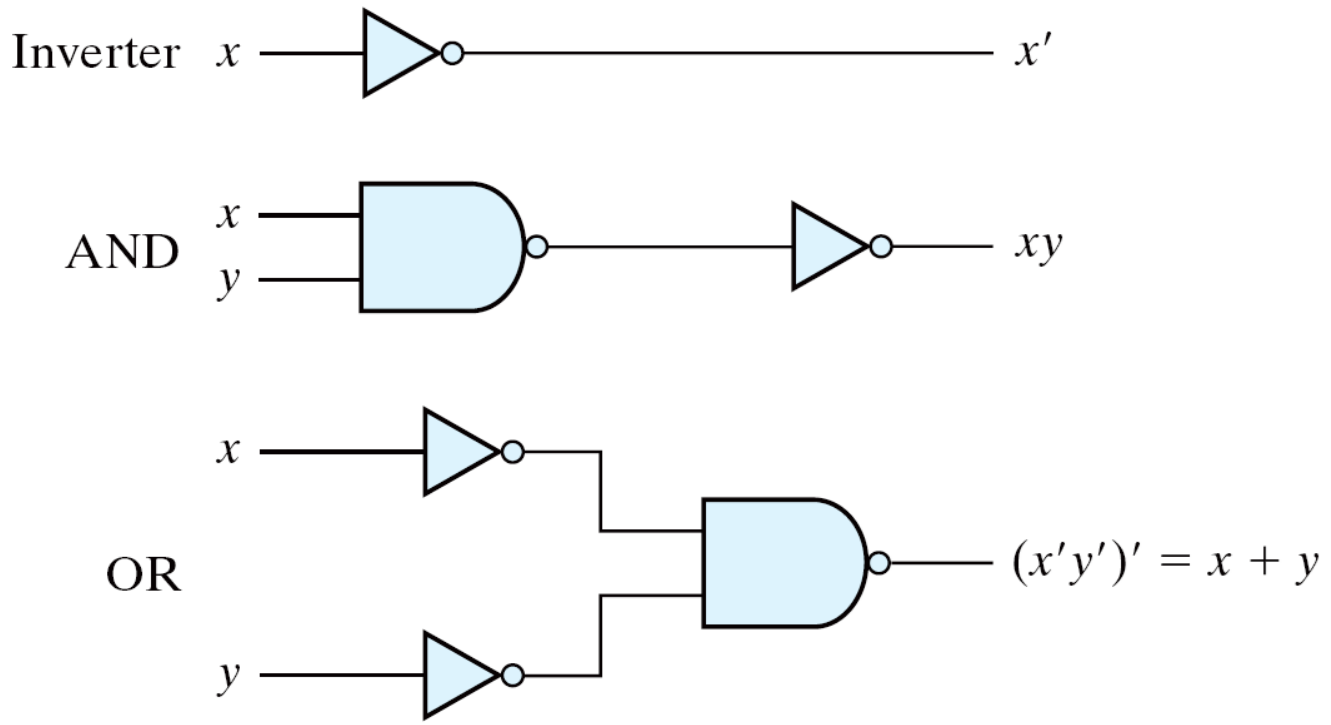
(a) $F = yz + w'x'$



(b) $F = yz + w'z$

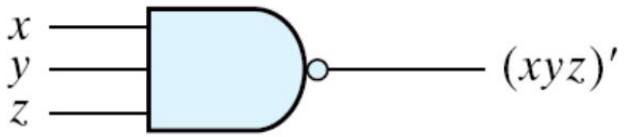
Implementace NANDu a NORu

- NAND je univerzální hradlo
  - Pomocí něj jsme schopni vytvořit jakýkoliv systém

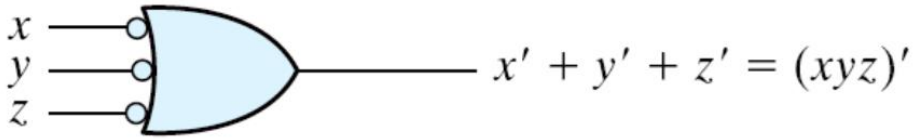


Hradlo NAND

- Dva grafické symboly pro NAND



(a) AND-invert

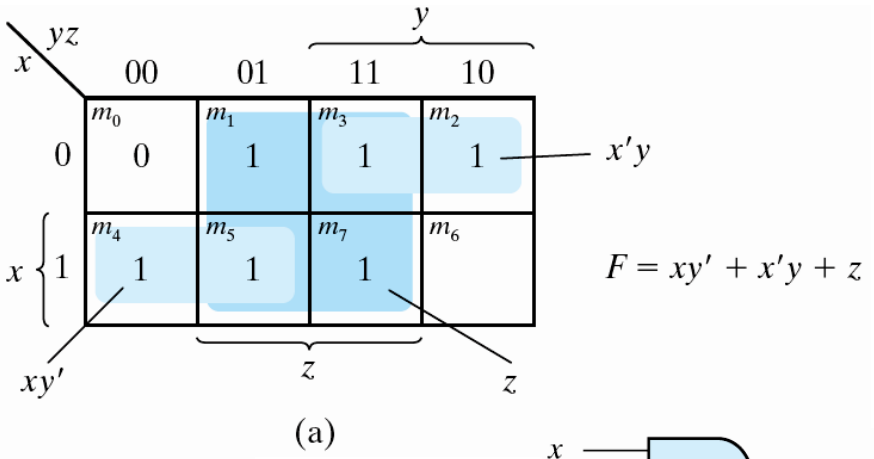


(b) Invert-OR

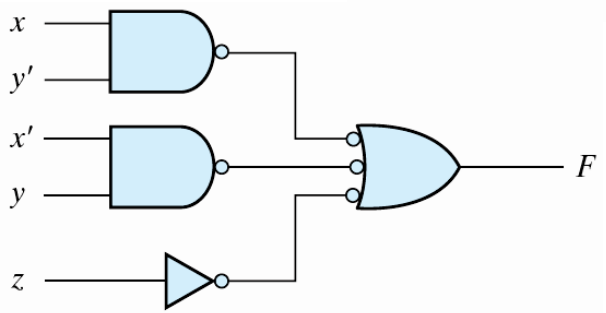
Příklad implementace hradel 1

- Example 3-10: implement $F(x, y, z) = F(x, y, z) = \sum(1,2,3,4,5,7)$

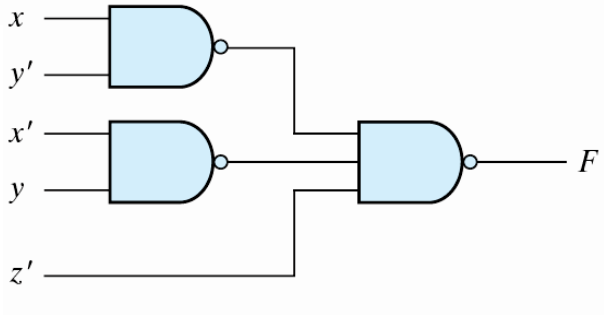
  $F(x, y, z) = xy' + x'y + z$



(a)



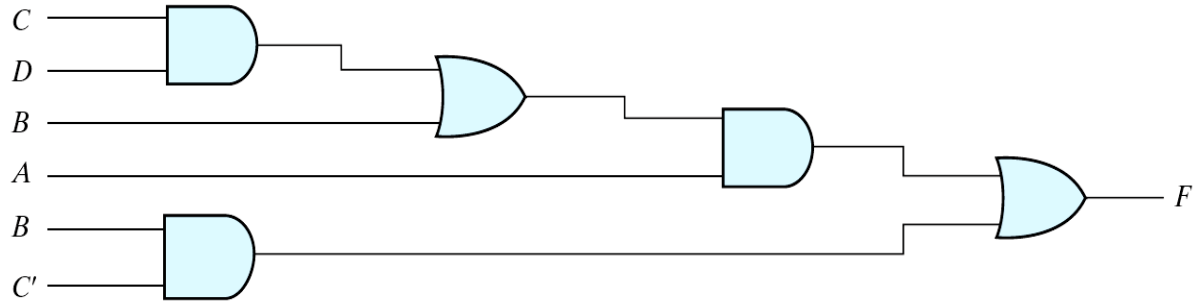
(b)



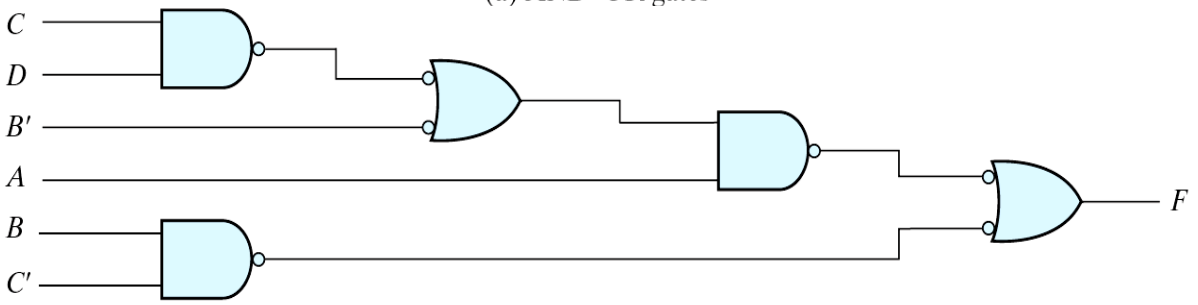
(c)

Mnohovrstvé NAND obvody

- Implementace booleanovské funkce
  - AND-OR logika → NAND-NAND logika
    - AND → AND + invertor
    - OR: invertor + OR = NAND

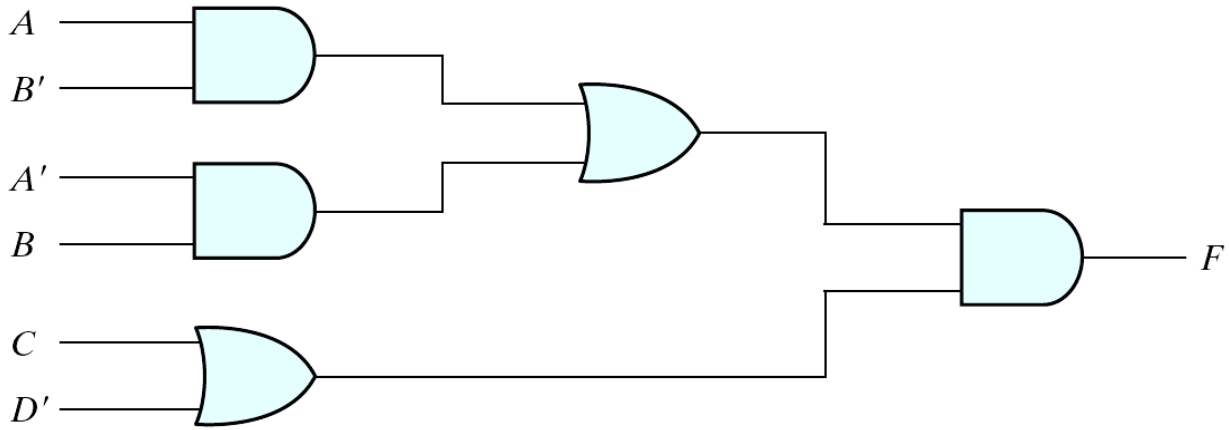


(a) AND–OR gates

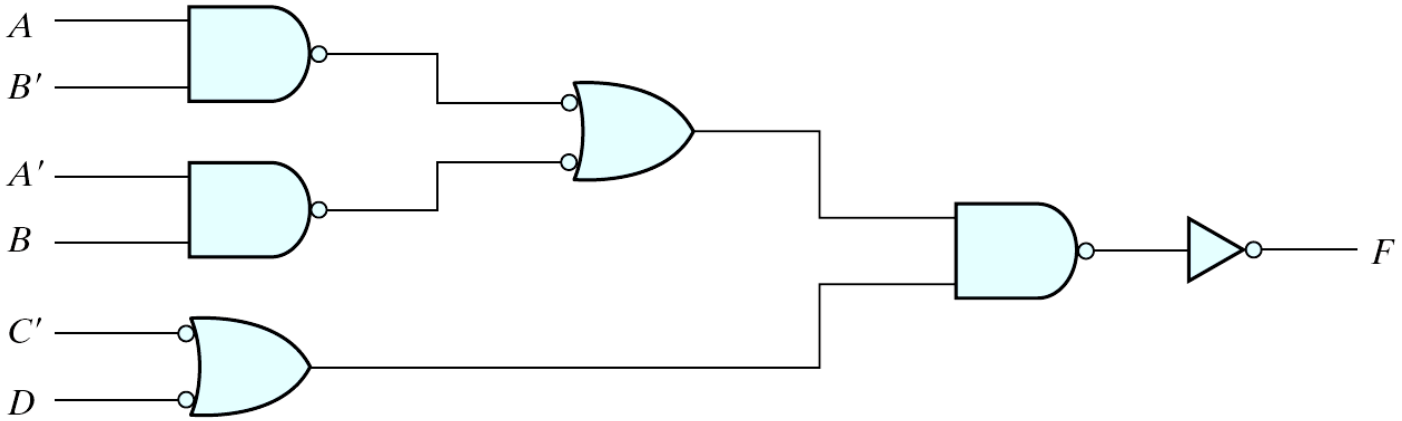


(b) NAND gates

Implementace NAND



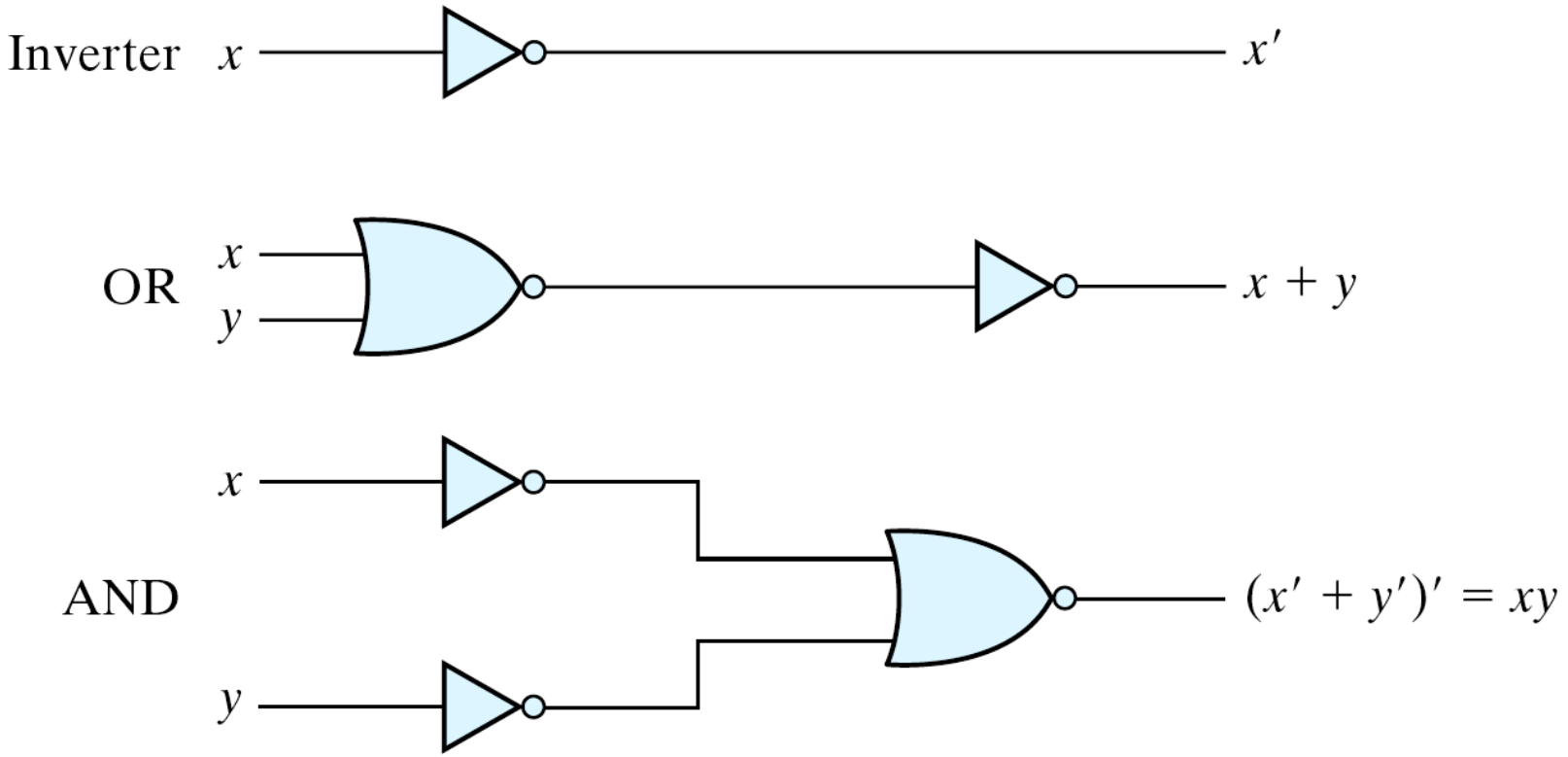
(a) AND–OR gates



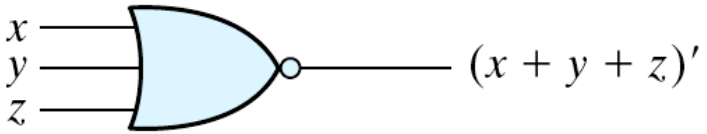
(b) NAND gates

Implementace NOR

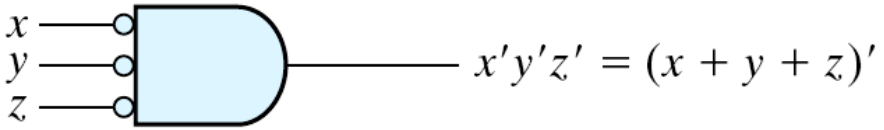
- Hradlo NOR je stejně jako NAND také univerzální



Dva grafické symboly pro hradlo NOR

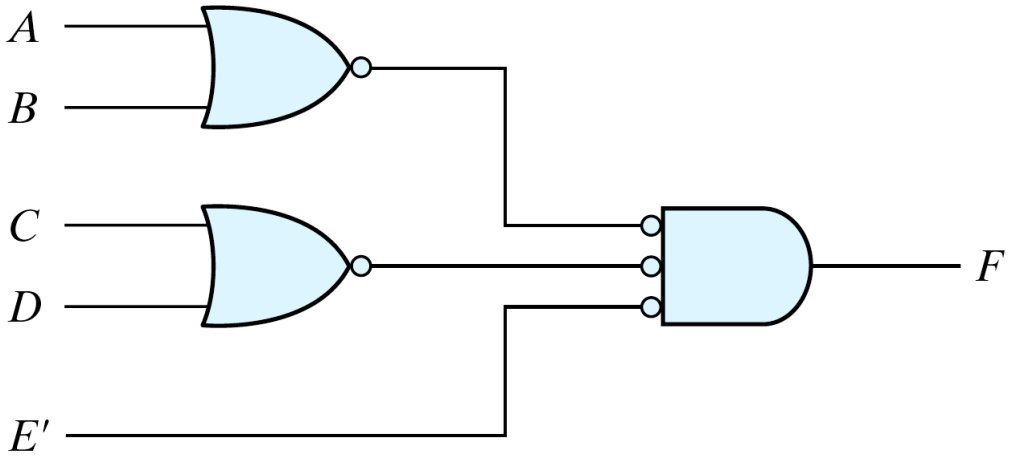


(a) OR-invert



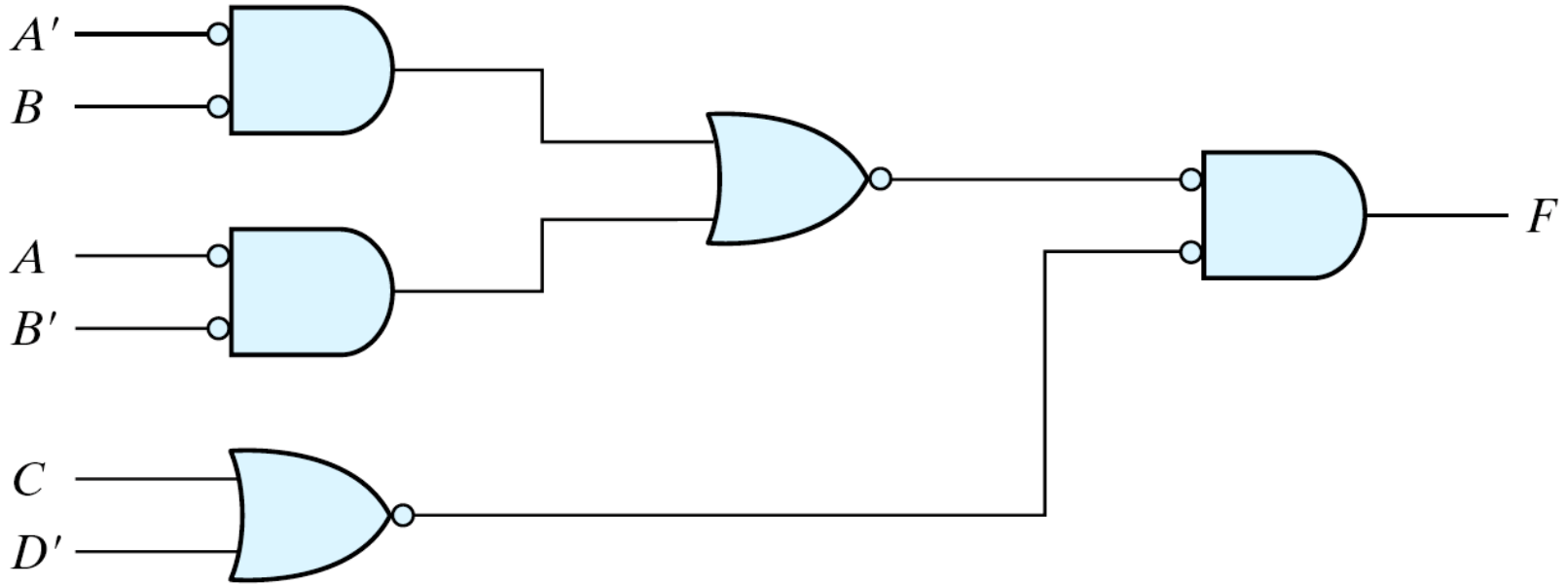
(b) Invert-AND

Příklad: F = (A + B)(C + D)E



Příklad implementace 2

Příklad: F = (AB′ +A′B)(C + D′)



F = (AB′ +A′B)(C + D′) s NOR hradly

Dvouvrstvé implementace

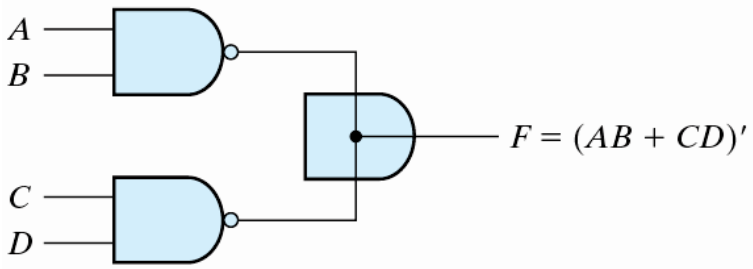
- Zapojovací logika
  - Zapojení mezi výstupem a dvěma hradly
  - Otevřený collector TTL NAND hradel: zapojovací-AND logika
  - NOR výstup ECL hradel: zapojovací-OR logika

$$F = (AB)' \cdot (CD)' = (AB + CD)' = (A' + B')(C' + D')$$

$$F = (A + B)' + (C + D)' = [(A + B)(C + D)]'$$

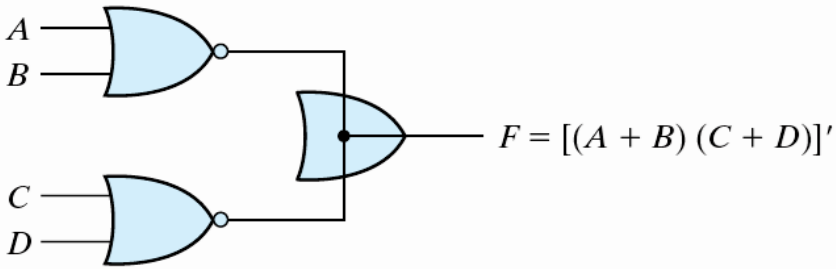
AND-OR-INVERT funkce

OR-AND-INVERT funkce



(a) Wired-AND in open-collector TTL NAND gates.

(AND–OR–INVERT)



(b) Wired-OR in ECL gates

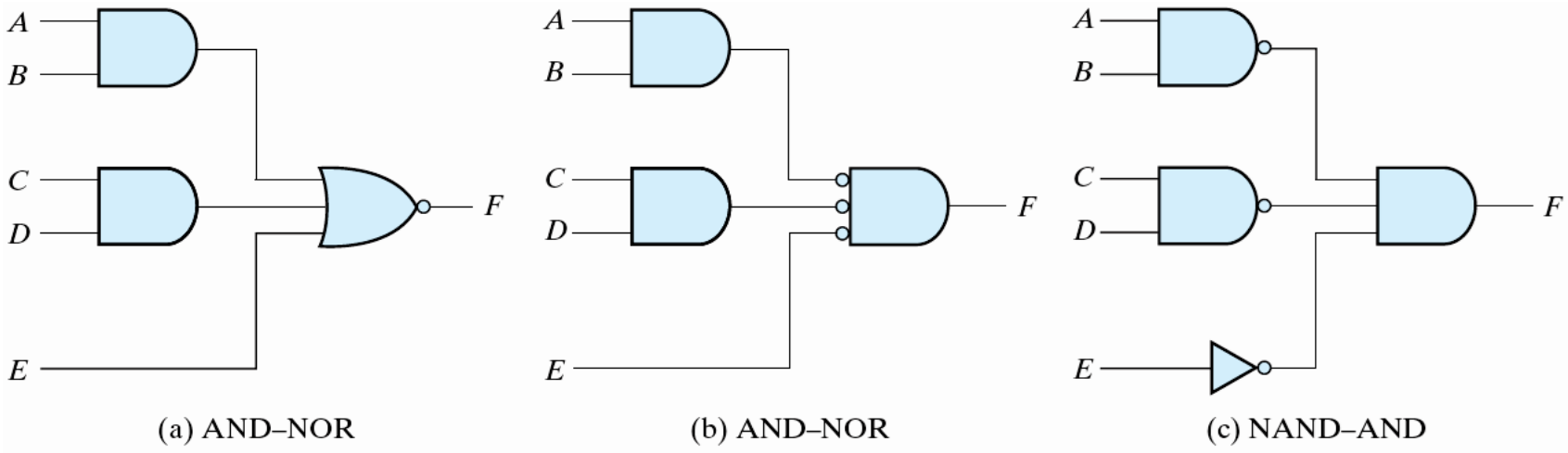
(OR–AND–INVERT)

Ne-degenerované formy

- 16 možných kombinací dvouvrstvé formy
  - Osm z nich: degenerovaná forma = jediná operace
    - AND-AND, AND-NAND, OR-OR, OR-NOR, NAND-OR, NAND-NOR, NOR-AND, NOR-NAND.
  - Osm ne-degenerovaných forem
    - AND-OR, OR-AND, NAND-NAND, NOR-NOR, NOR-OR, NAND-AND, OR-NAND, AND-NOR.
    - AND-OR a NAND-NAND = suma produktů.
    - OR-AND a NOR-NOR = produkt sum.
    - NOR-OR, NAND-AND, OR-NAND, AND-NOR = ?

AND-OR-INVERTOR implementace

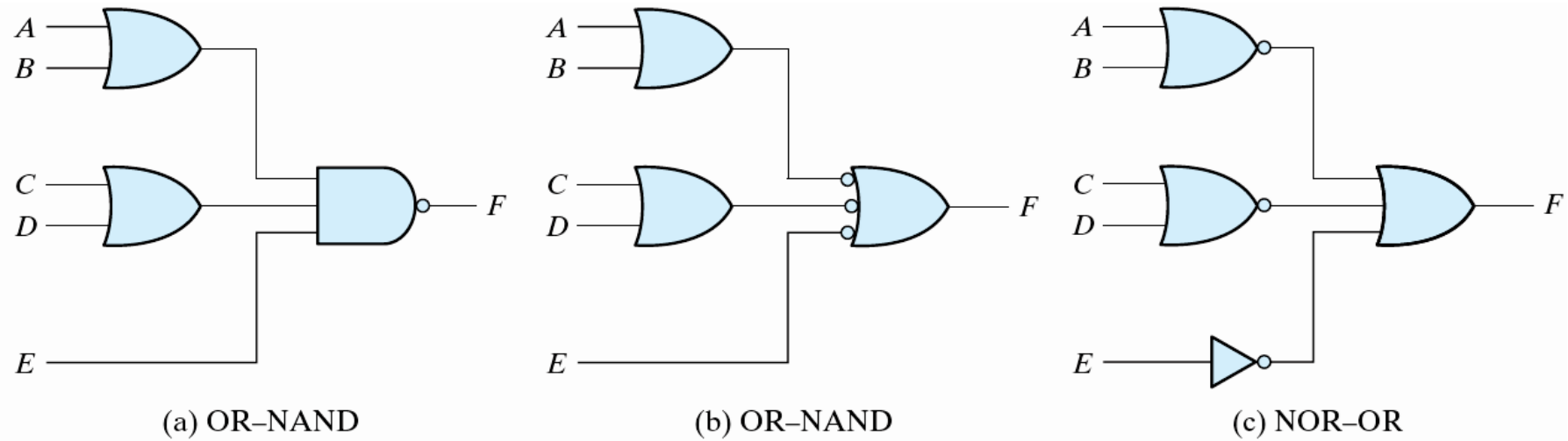
- AND-OR-INVERTOR (AOI) implementace
  - NAND-AND = AND-NOR = AOI
  - $F = (AB+CD+E)'$
  - $F' = AB+CD+E$



F = (AB +CD +E)′

OR-AND-INVERTOR implementace

- OR-AND-INVERT (OAI) implementace
  - OR-NAND = NOR-OR = OAI
  - $F = ((A+B)(C+D)E)'$
  - $F' = (A+B)(C+D)E$



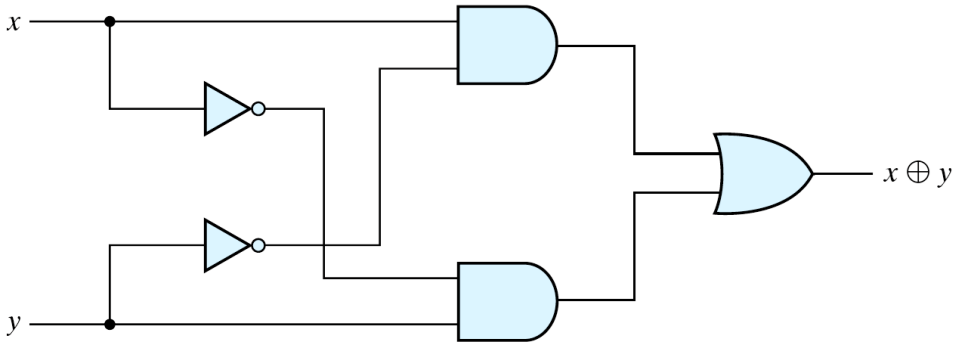
F = ((A+B)(C+D)E)'

Exclusive-OR funkce

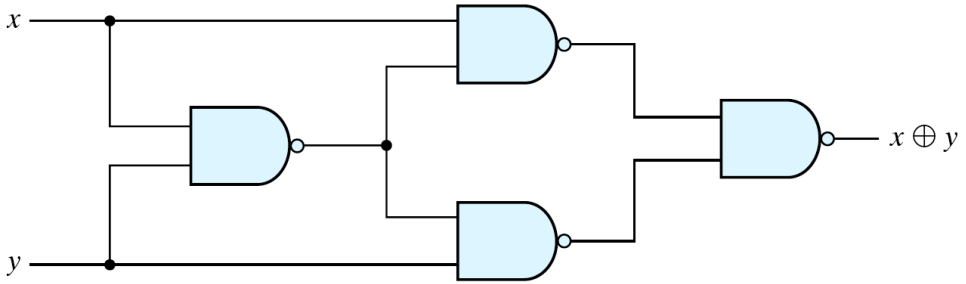
- Exclusive-OR (XOR)
  - $x \oplus y = xy' + x'y$
- Exclusive-NOR (XNOR)
  - $(x \oplus y)' = xy + x'y'$
- Nějaké identity
  - $x \oplus 0 = x$
  - $x \oplus 1 = x'$
  - $x \oplus x = 0$
  - $x \oplus x' = 1$
  - $x \oplus y' = (x \oplus y)'$
  - $x' \oplus y = (x \oplus y)'$
- Komutativní a asociativní
  - $A \oplus B = B \oplus A$
  - $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

Exclusive-OR implementace

- Implementace
  - $(x'+y')x + (x'+y')y = xy'+x'y = x \oplus y$



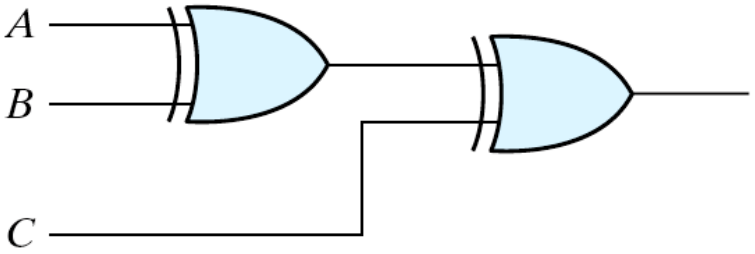
(a) With AND–OR–NOT gates



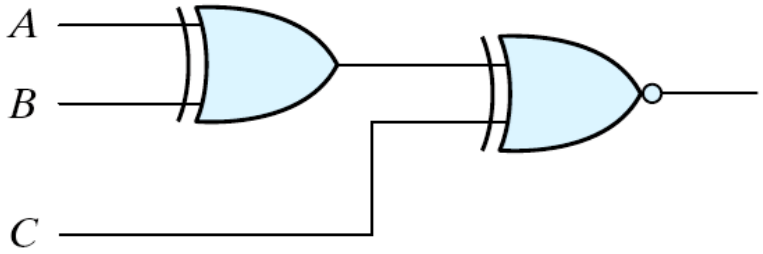
(b) With NAND gates

XOR a XNOR

- Logický diagram a sudé a liché funkce



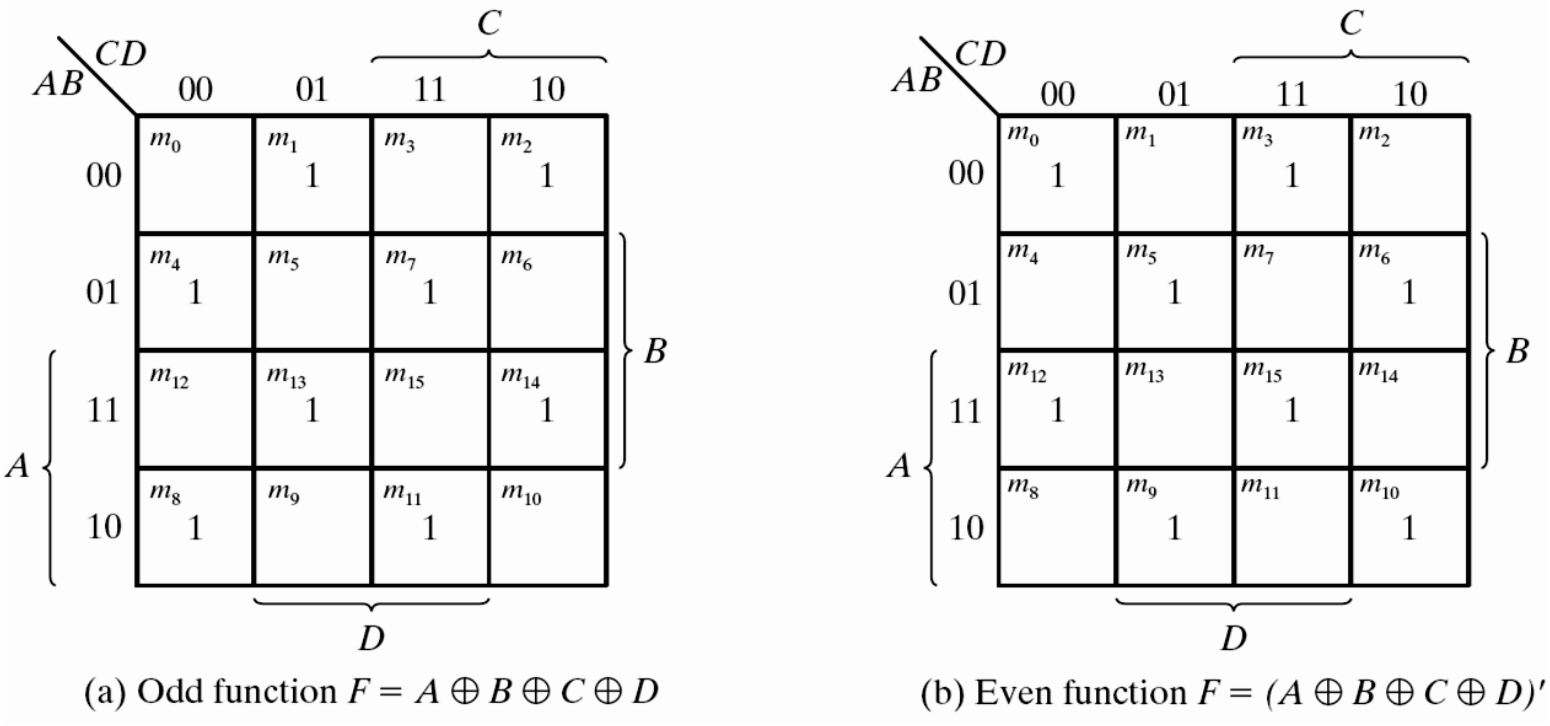
(a) 3-input odd function



(b) 3-input even function

XOR funkce se čtyřmi proměnnými

- XOR funkce se čtyřmi proměnnými
- $A \oplus B \oplus C \oplus D = (AB'+A'B) \oplus (CD'+C'D) = (AB'+A'B)(CD+C'D')+(AB+A'B')(CD'+C'D)$



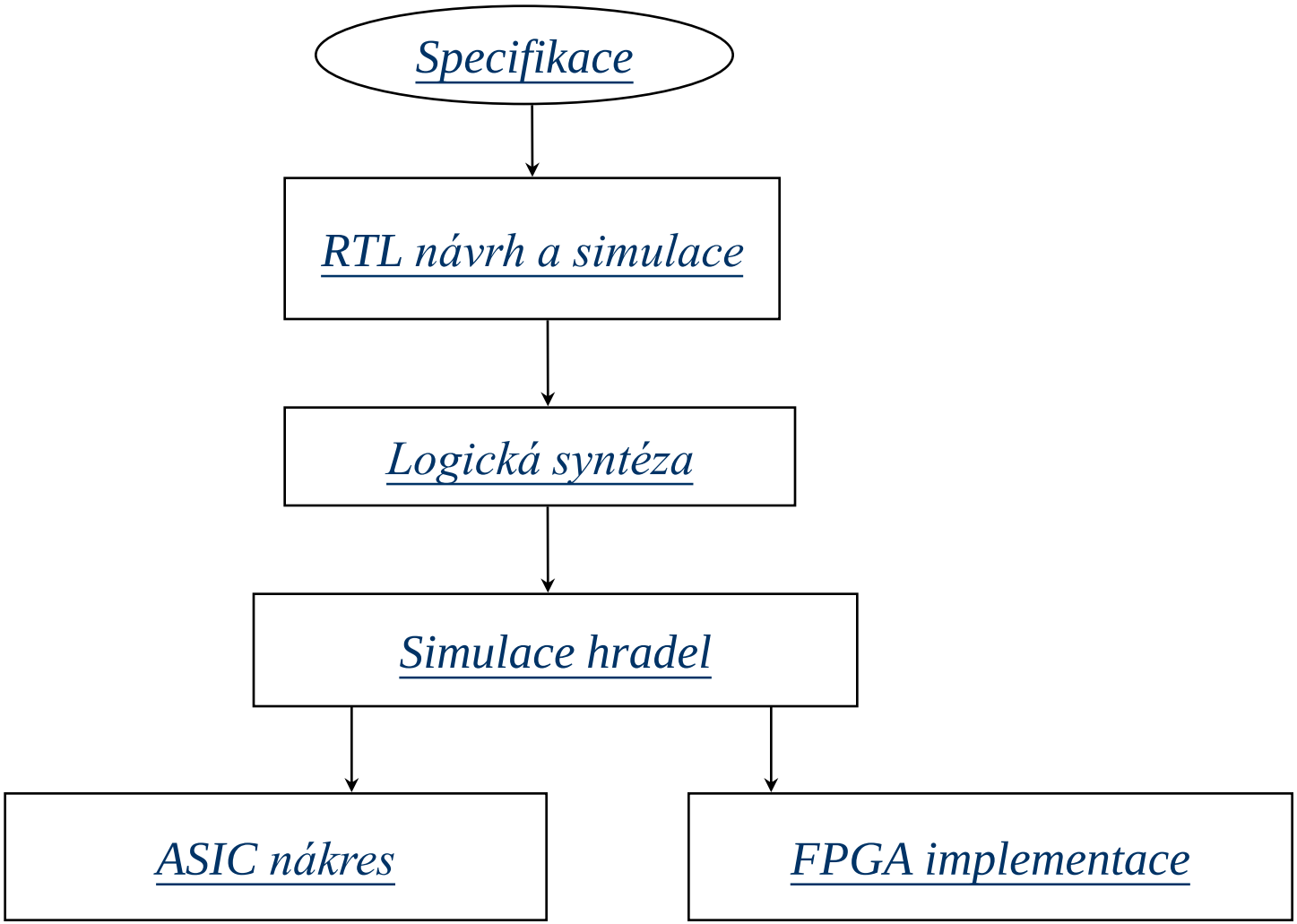
(a) Odd function $F = A \oplus B \oplus C \oplus D$

(b) Even function $F = (A \oplus B \oplus C \oplus D)'$

Jazyk popisující hardware (HDL)

- Popisuje návrh obvodu v textové formě
  - Struktura hardwaru
  - Funkce/chování
  - Časování
- VHDL and Verilog HDL

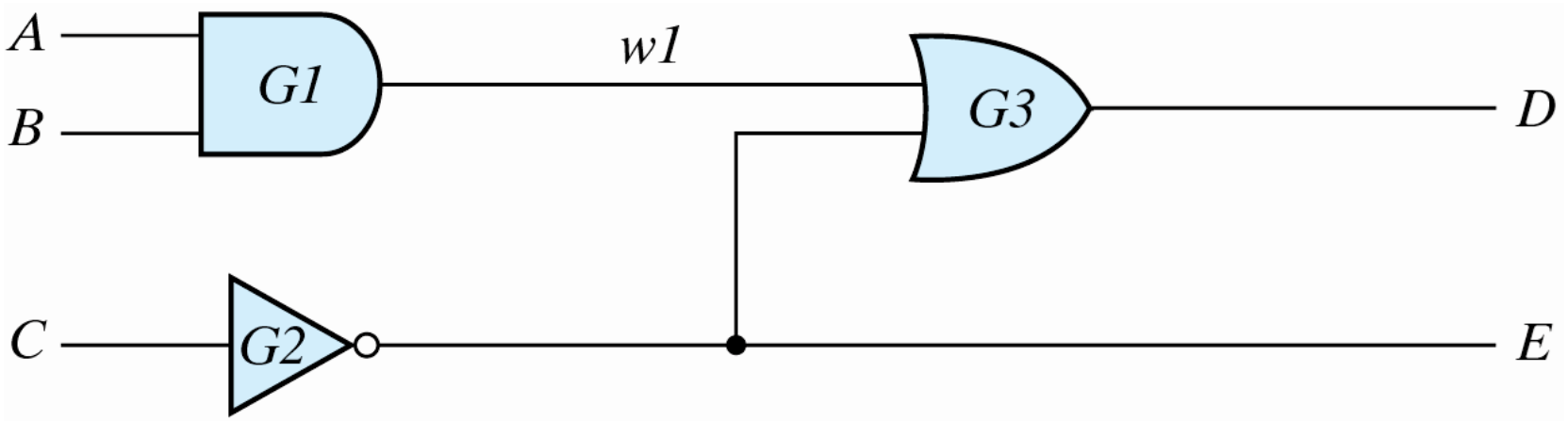
Top-down návrh



Deklarace modulu

- Příklady klíčových slov:

  modul, end-modul, výstup, vstup, propojení and, or, a **not**



HDL příklad 1

- HDL popis pro obvod

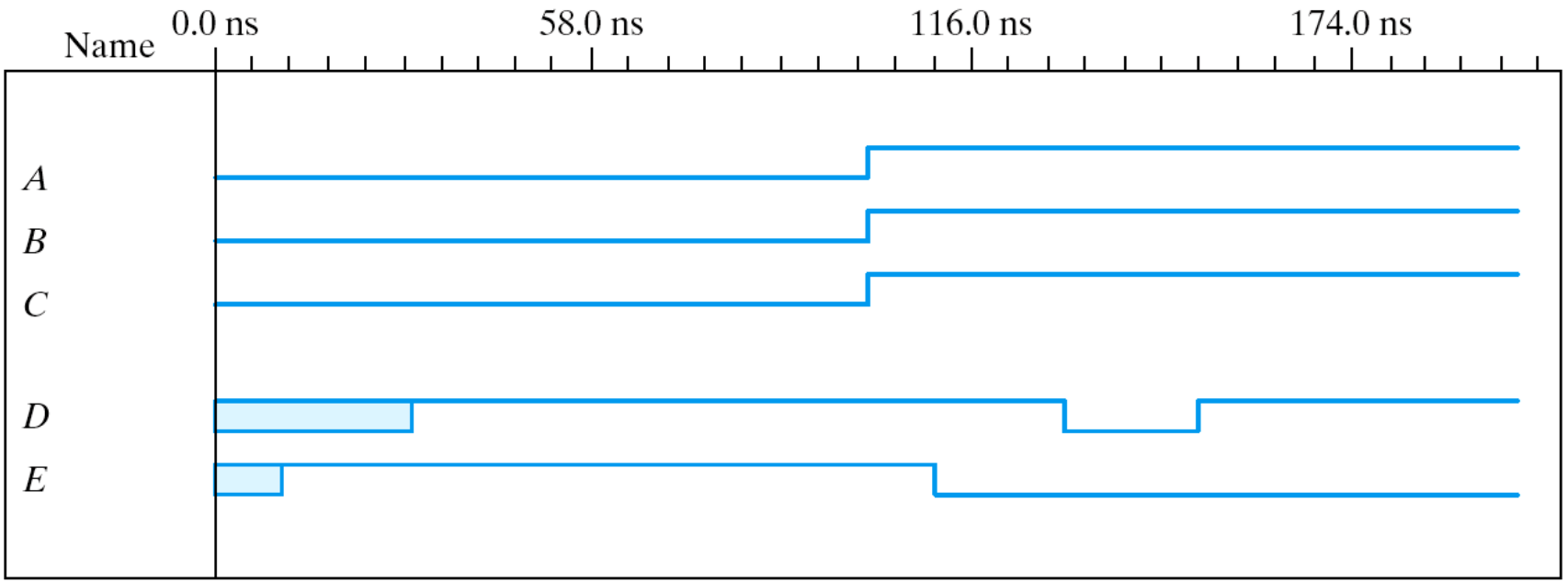
HDL Example 3.1 (Combinational logic modeled with primitives)

```
// Verilog model of circuit of Figure 3.37. IEEE 1364–1995 Syntax

module  Simple_Circuit (A, B, C, D, E);
 output          D, E;
 input           A, B, C;
 wire            w1;

 and             G1 (w1, A, B); // Optional gate instance name
 not             G2 (E, C);
 or              G3 (D, w1, E);
endmodule
```

Simulace výstupu pro HDL



Booleanovské vyjádření

- Booleanovské vyjádření obvodu

```
assign D = (A & B)|~C;
```

- Booleanovské vyjádření:

$$E = A + BC + B'D$$

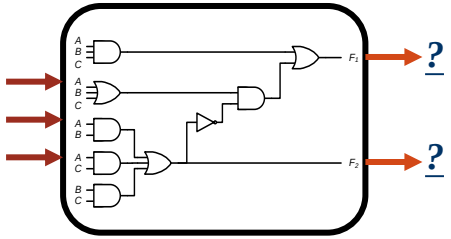
$$F = B'C + BC'D'$$

Kombinační obvody

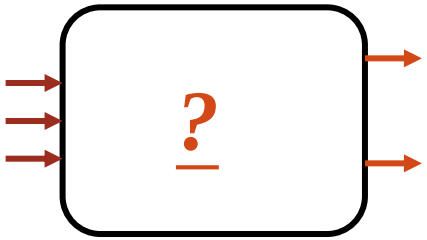
Když se vstup změní, výstup se může změnit se zpožděním

Kombinační obvody

- Analýza
  - Pro daný obvody nalezněte jeho *funkci*
  - Funkce může být vyjádřena jako:
    - Booleanovská funkce
    - Pravdivostní tabulka

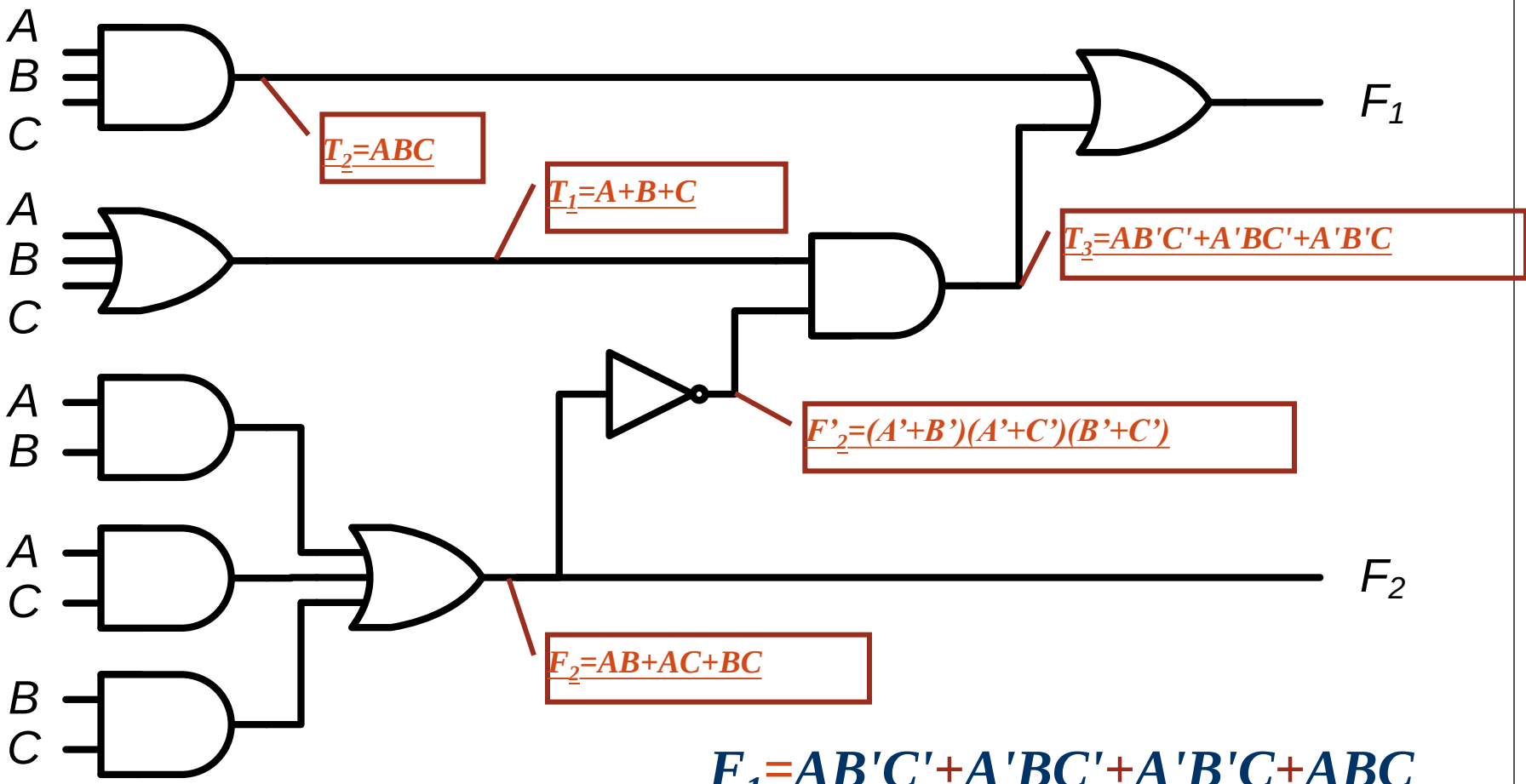


- Návrh
  - Pro požadovanou funkci, navrhněte její *obvod*
  - Funkce může bát vyjádřena jako:
    - Booleanovská funkce
    - Pravdivostní tabullka



Procedura analýzy

- Metoda booleanovského vyjádření

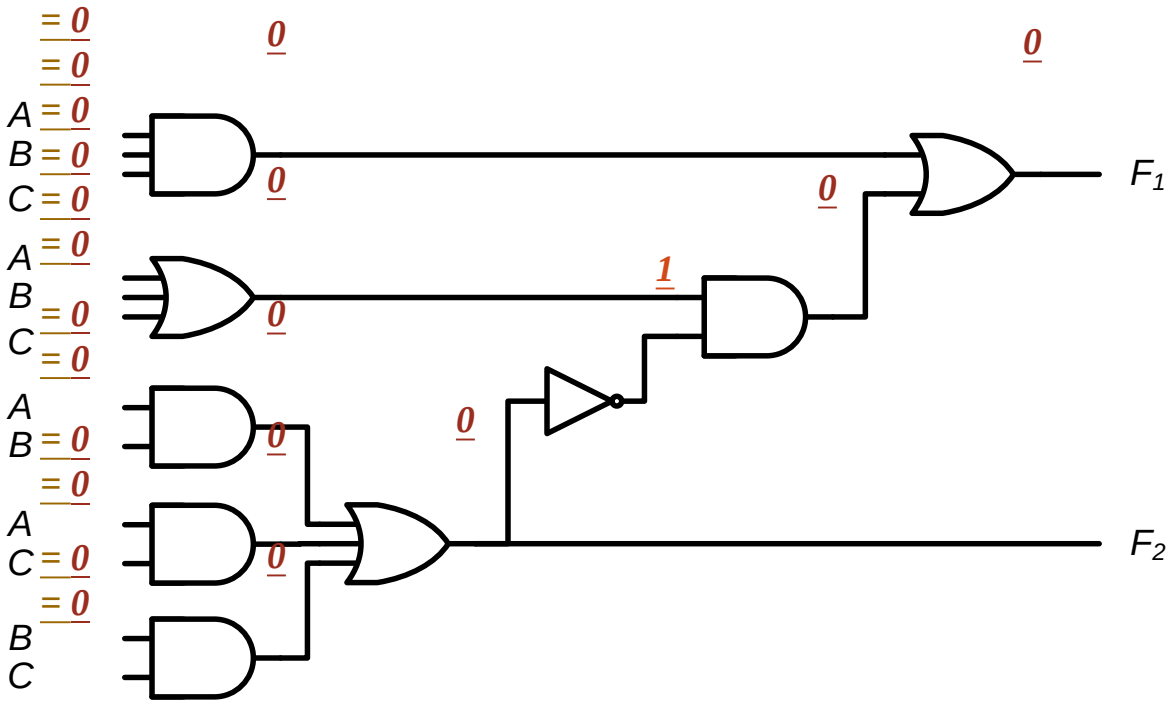


F_1=AB'C'+A'BC'+A'B'C+ABC

F_2=AB+AC+BC

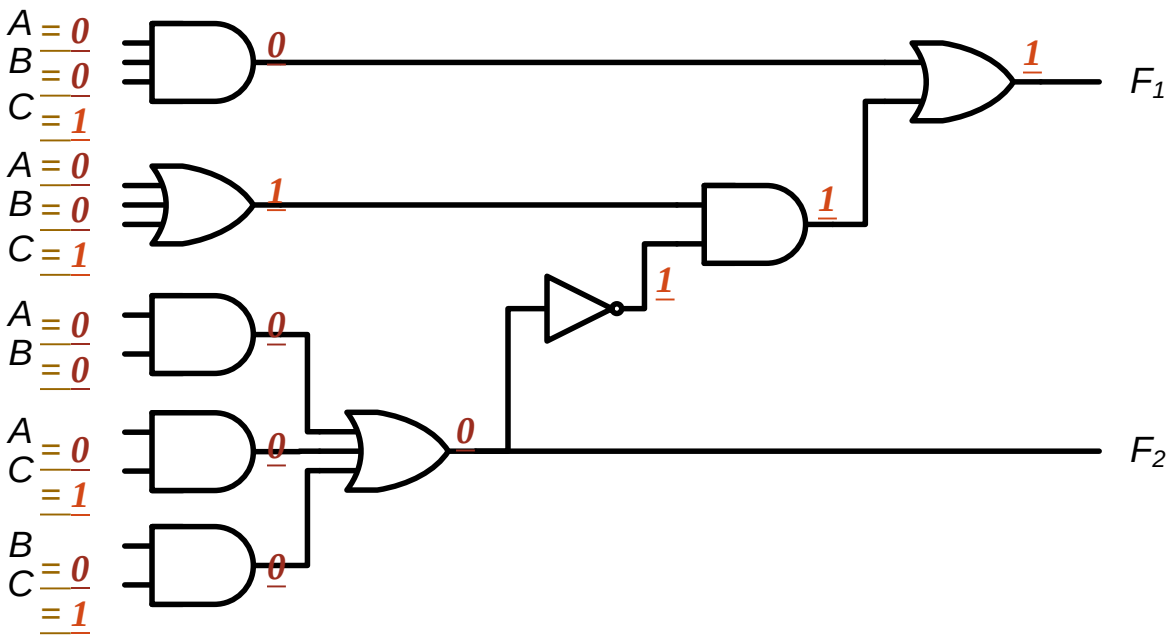
Procedura analýzy

- Metoda pravdivostní tabulky



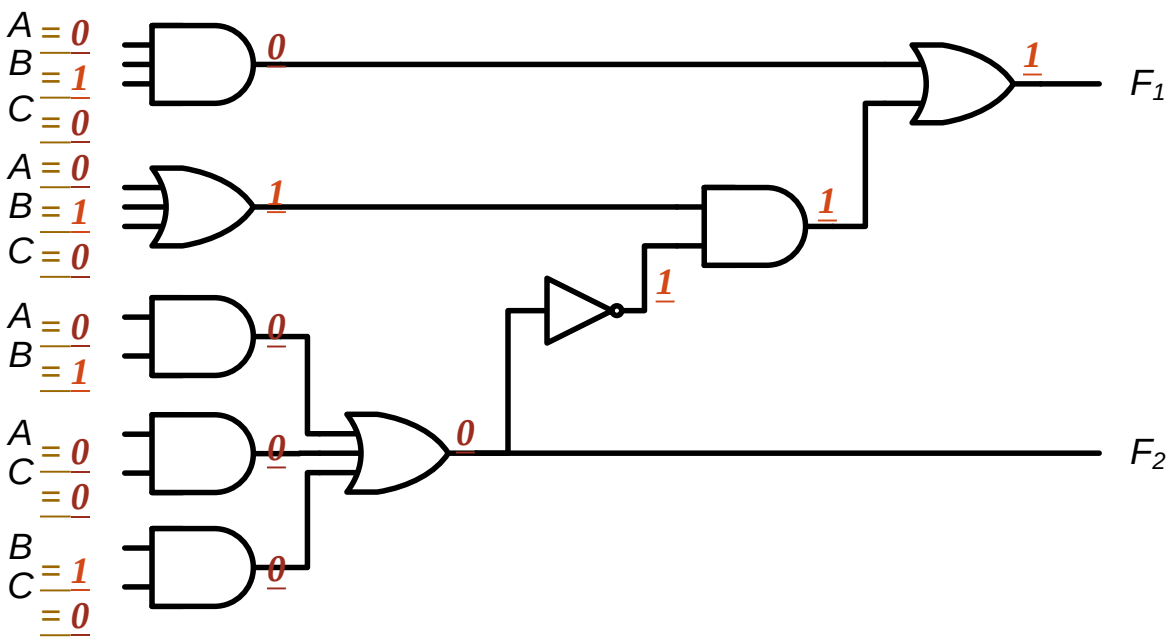
A B C	F_1	F_2
0 0 0	0 0	

Procedura analýzy



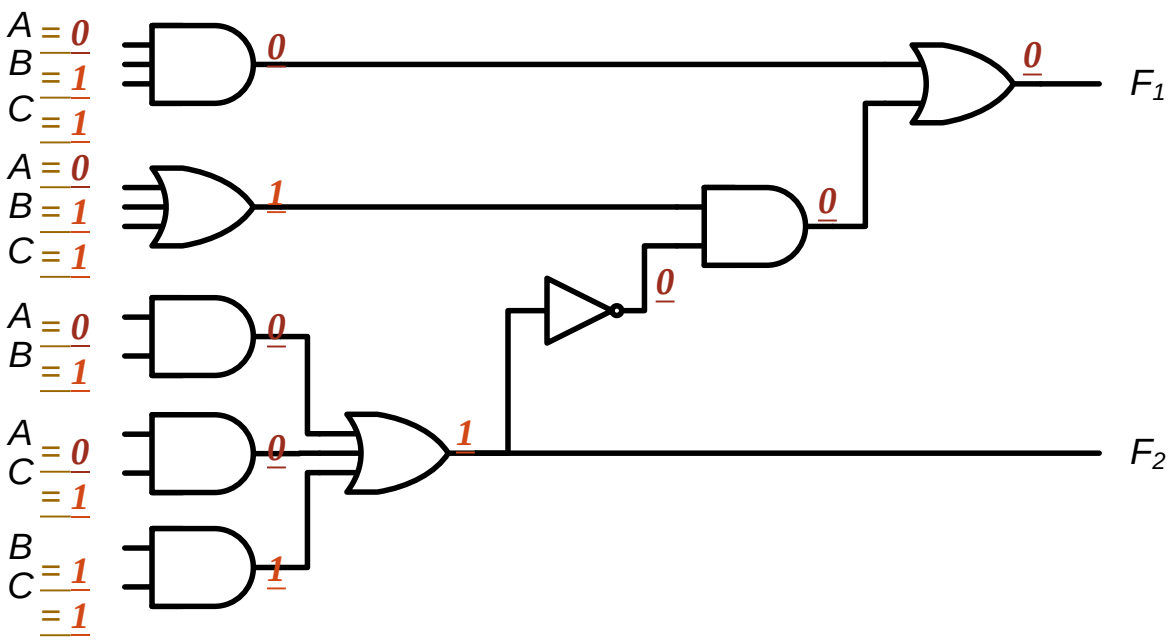
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1 0	

Procedura analýzy



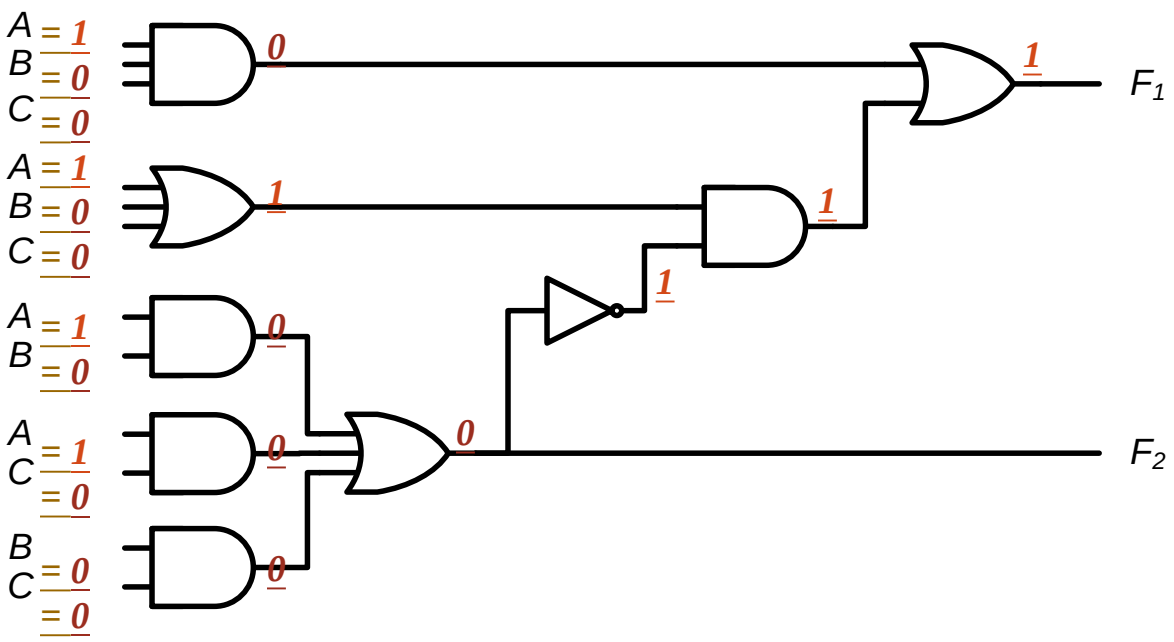
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1 0	

Procedura analýzy



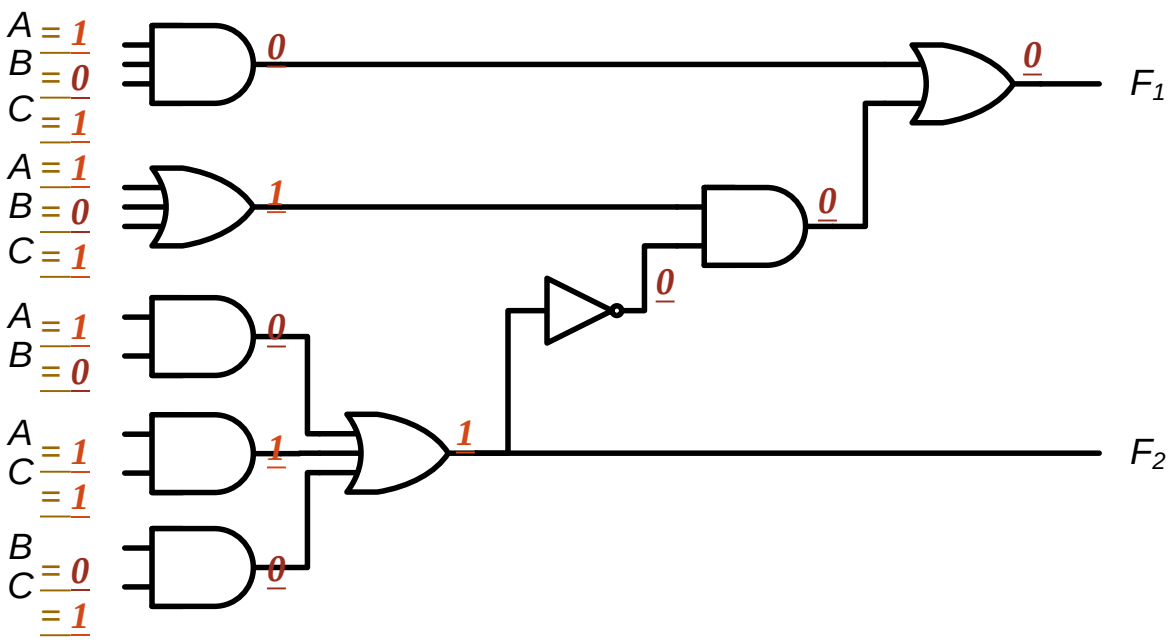
A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0 1	

Procedura analýzy



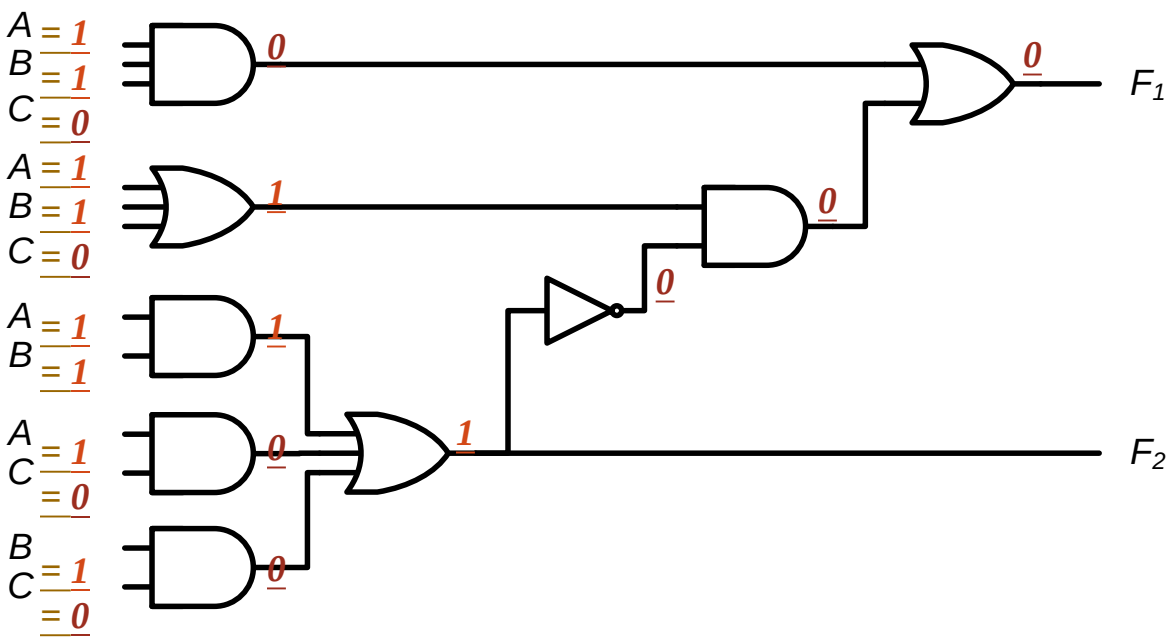
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1 0	

Procedura analýzy



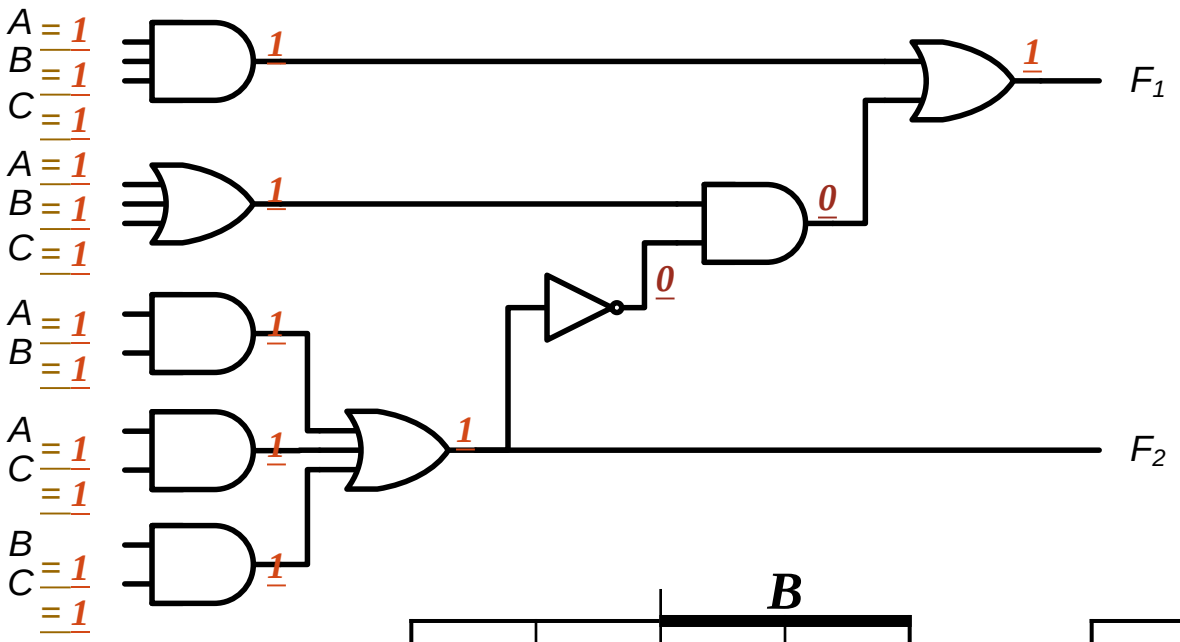
A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0 1	

Procedura analýzy



A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0 1	

Procedura analýzy



A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

A \ B	0	1	0	1
1	0	1	0	

C

A \ B	0	0	1	0
0	1	1	1	

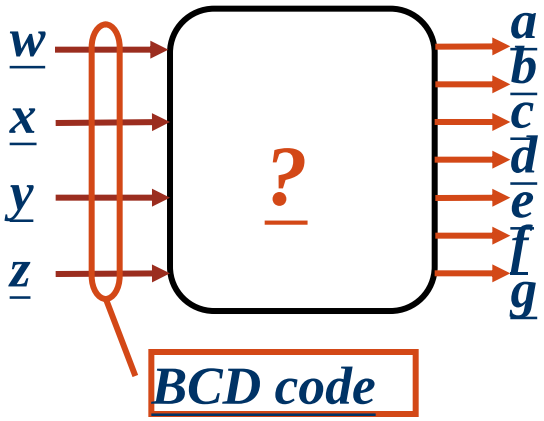
C

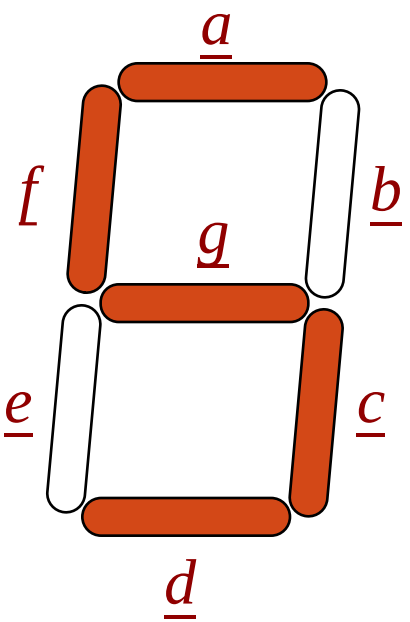
$F_1 = AB'C' + A'BC' + A'B'C + ABC$

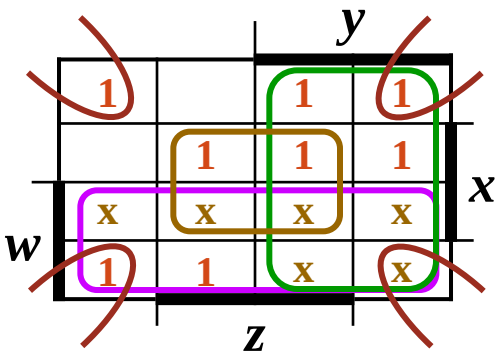
$F_2 = AB + AC + BC$

Sedmi-segmentový dekodér

w x y z	a b c d e f g
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 1 0 1 1
1 0 1 0	x x x x x x x
1 0 1 1	x x x x x x x
1 1 0 0	x x x x x x x
1 1 0 1	x x x x x x x
1 1 1 0	x x x x x x x
1 1 1 1	x x x x x x x







$a = w + y + xz + x'z'$

$b = \ldots$

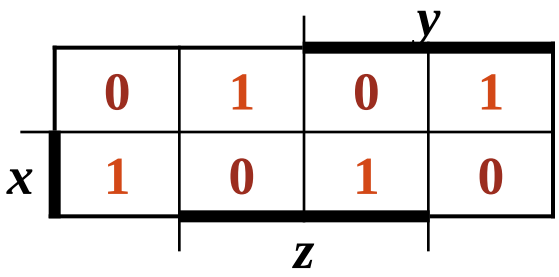
$c = \ldots$

$d = \ldots$

Binární čítač

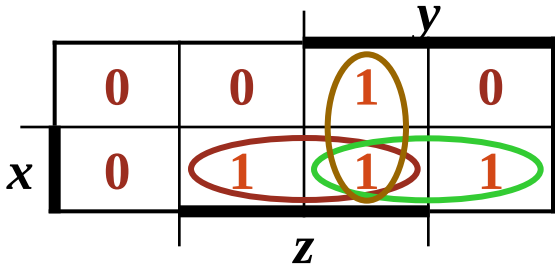
- Plný čítač
  - Součet 1-bit + 1-bit + 1-bit
  - Produkuje součet a carry

x y z	C S
0 0 0	0 0
0 0 1	0 1
0 1 0	0 1
0 1 1	1 0
1 0 0	0 1
1 0 1	1 0
1 1 0	1 0
1 1 1	1 1



$$\begin{array}{r} x \\ + \quad y \\ + \quad z \\ \hline\hline C \quad S \end{array}$$

$$S = xy'z' + x'yz' + x'y'z + xyz = x \oplus y \oplus z$$



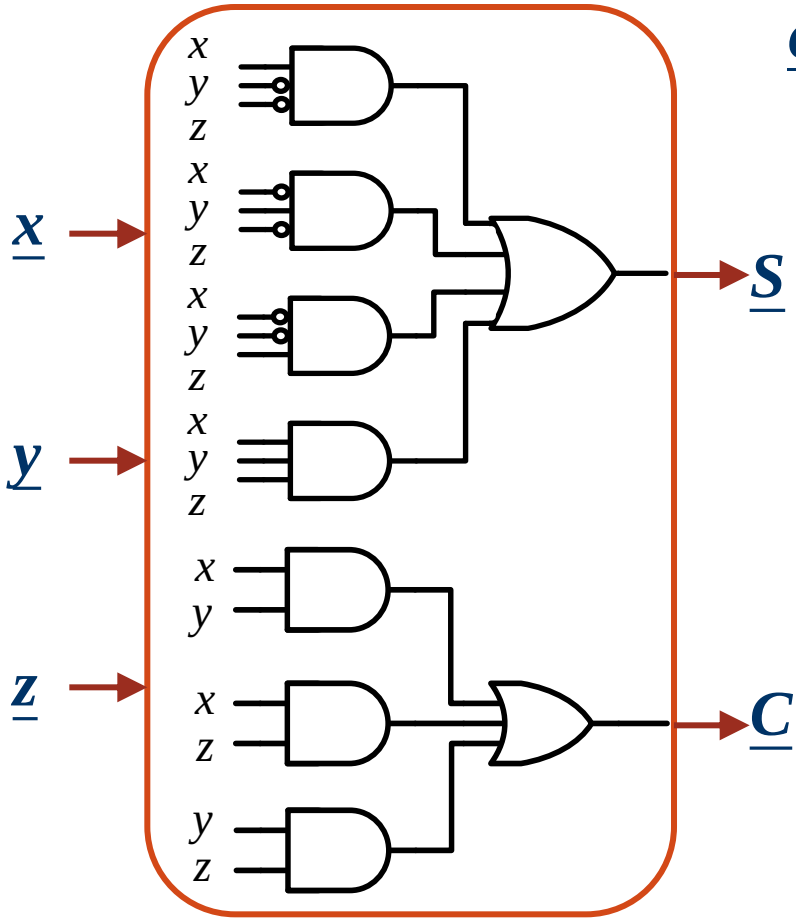
$$C = xy + xz + yz$$

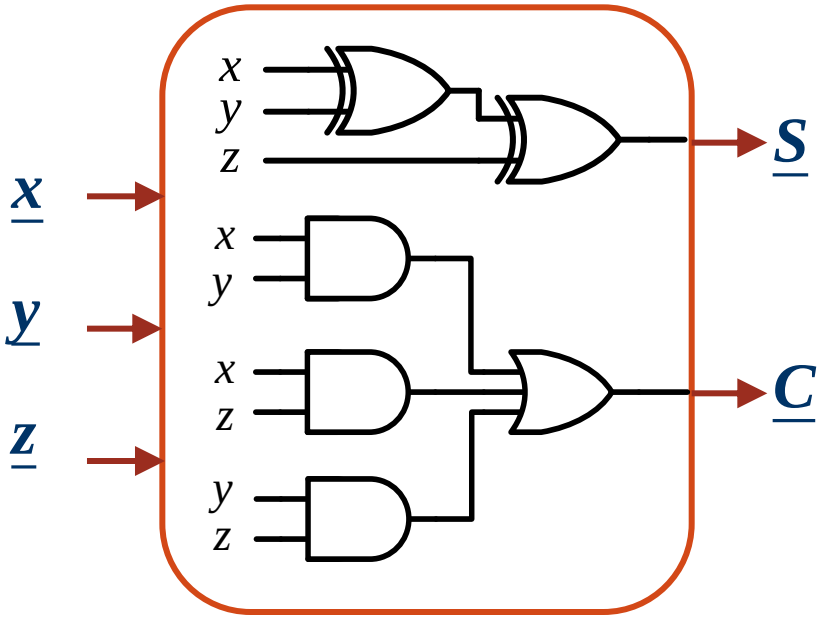
Binární čítač

- Plný čítač

$S = xy'z'+x'yz'+x'y'z+xyz = x \oplus y \oplus z$

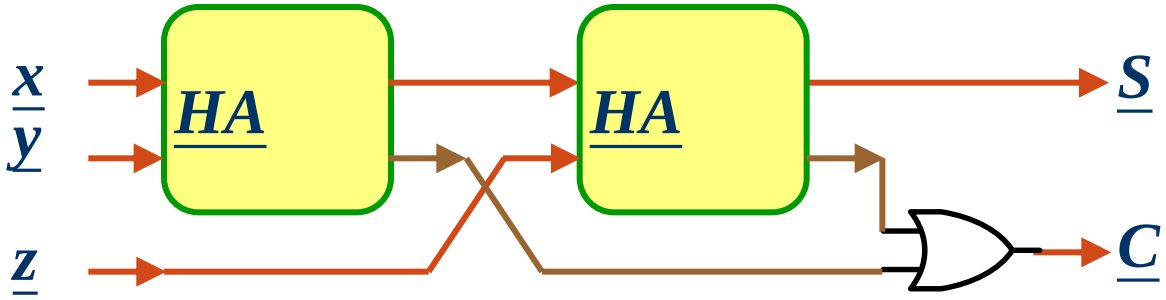
$C = xy + xz + yz$

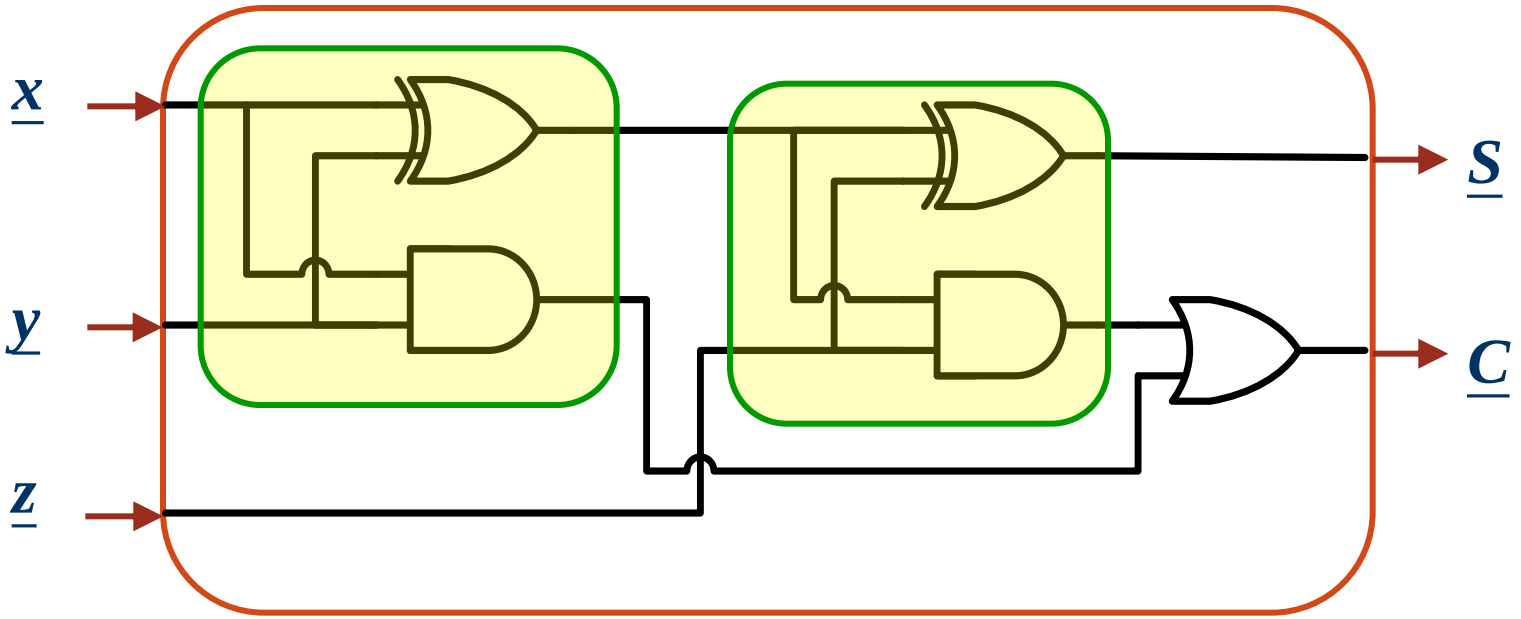




Binární čítač

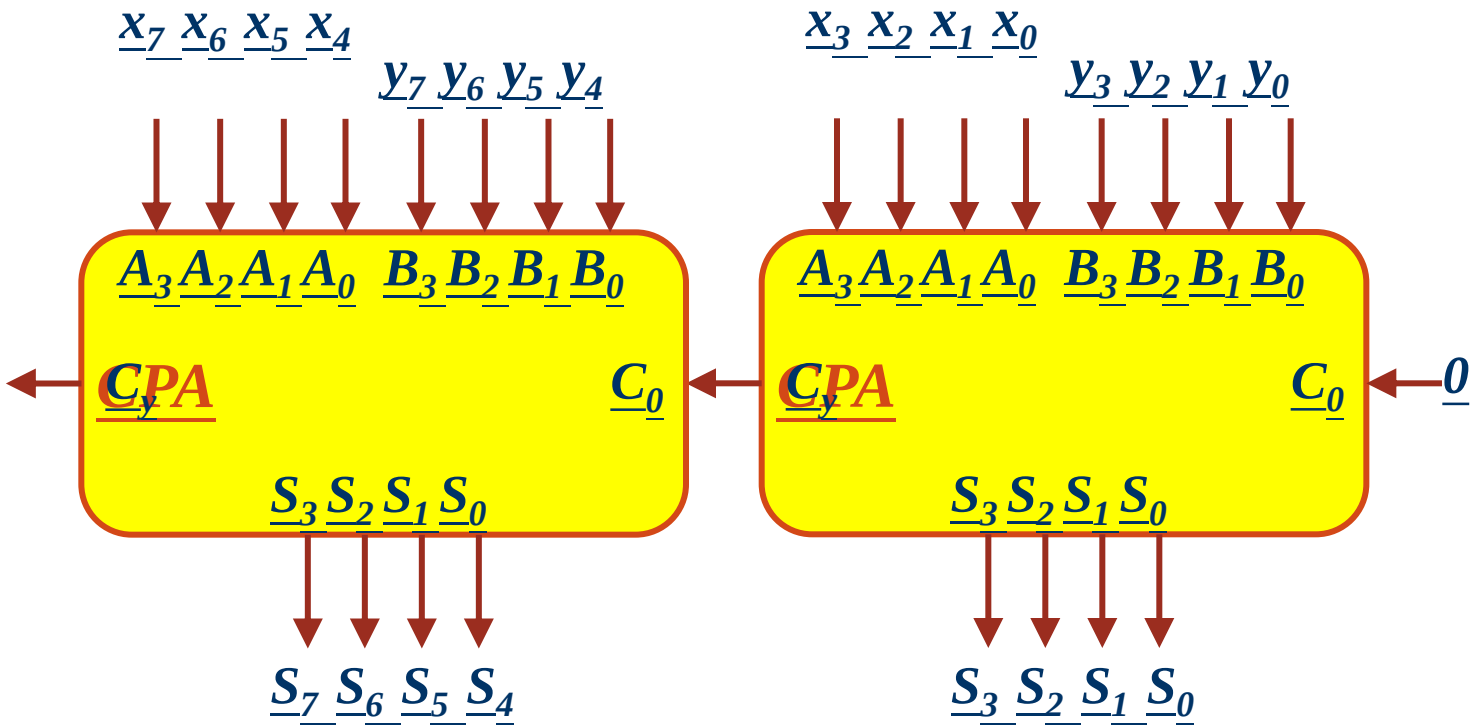
- Plný čítač





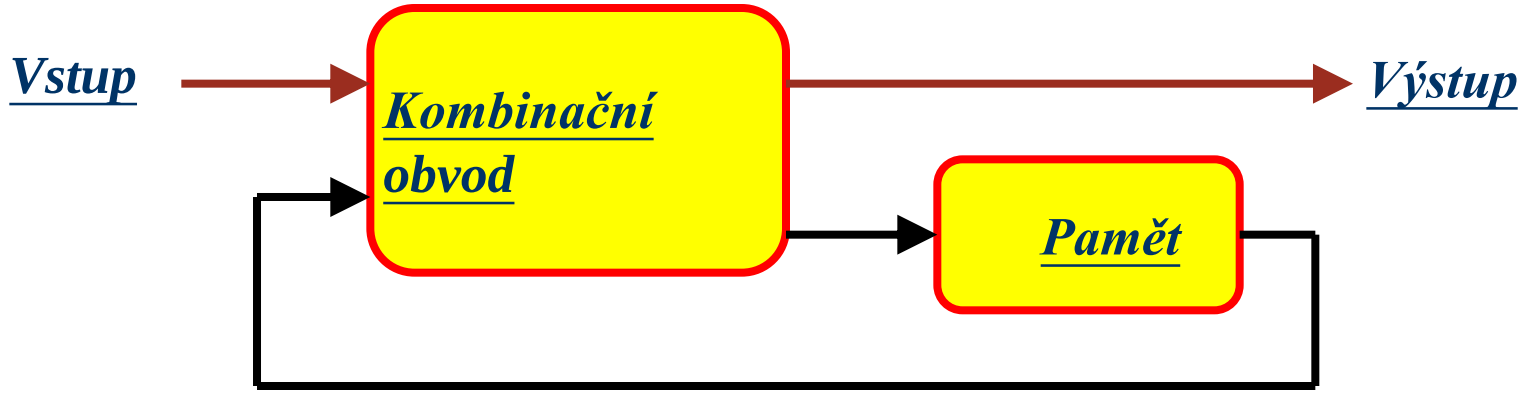
Binární čítač

- Carry propagační čítač

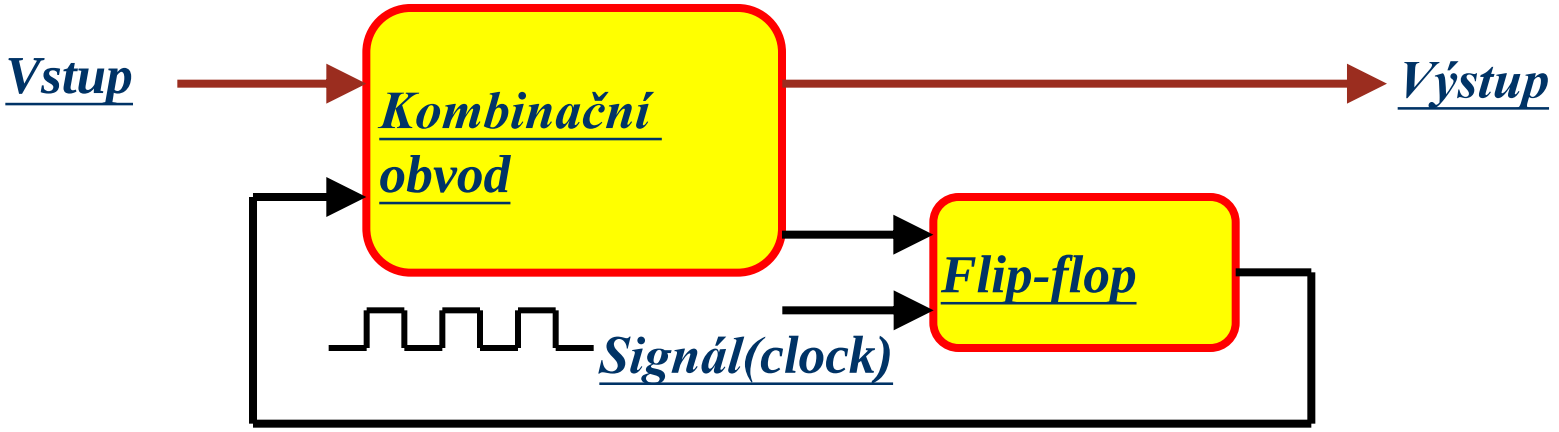


Sekvenční obvody

- Asynchroní

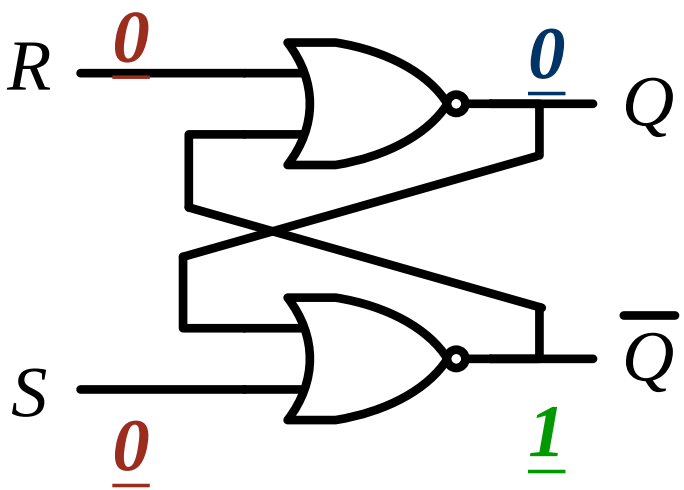


- Sycnhroní



Klopný obvod

- *SR*



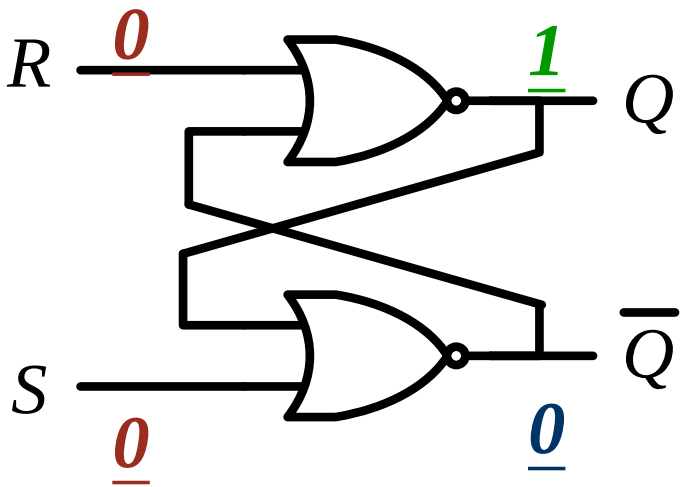
S	R	Q_0	Q	Q'
0	0	0	0	1

$Q = Q_0$

Počáteční hodnota

Klopný obvod

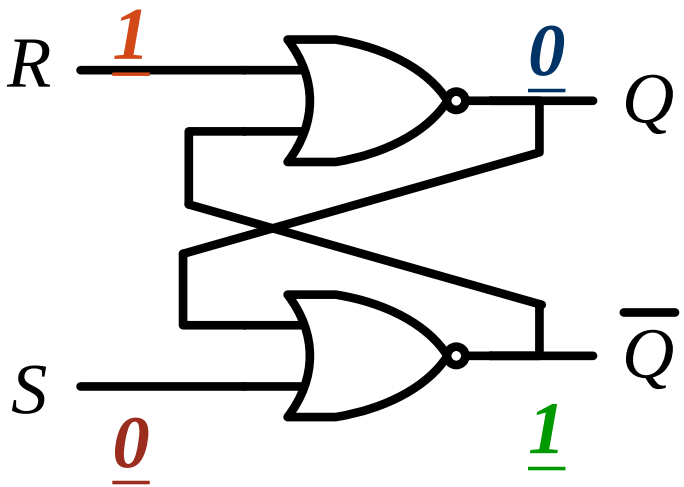
- *SR*



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	$Q = Q_0$

Klopný obvod

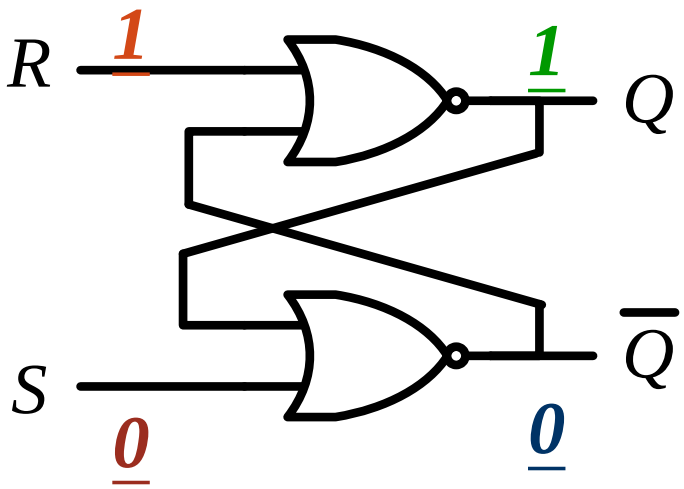
- SR



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$

Klopný obvod

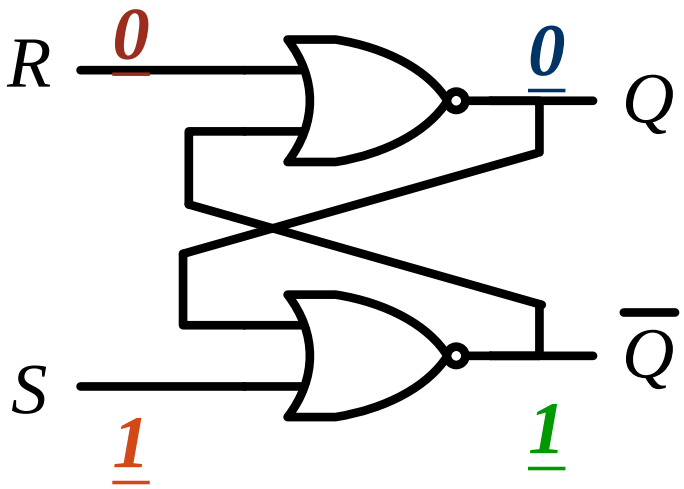
- *SR*



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	$Q = Q_0$
0	1	0	0	1	$Q = 0$
0	1	1	0	1	$Q = 0$

Klopný obvod

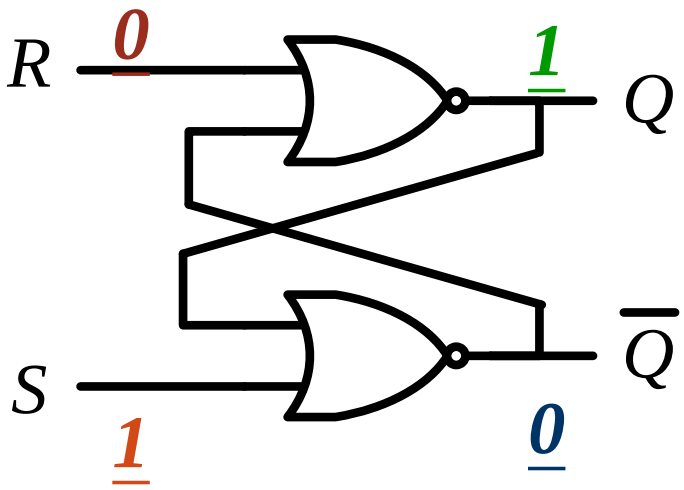
- *SR*



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$

Klopný obvod

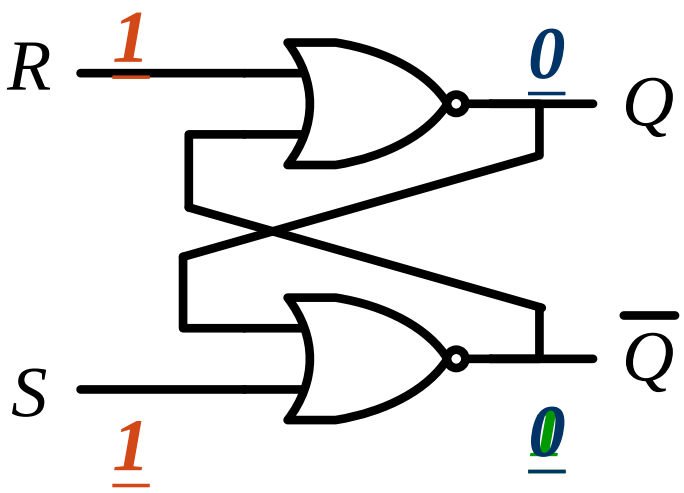
- *SR*



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	$Q = 1$

Klopný obvod

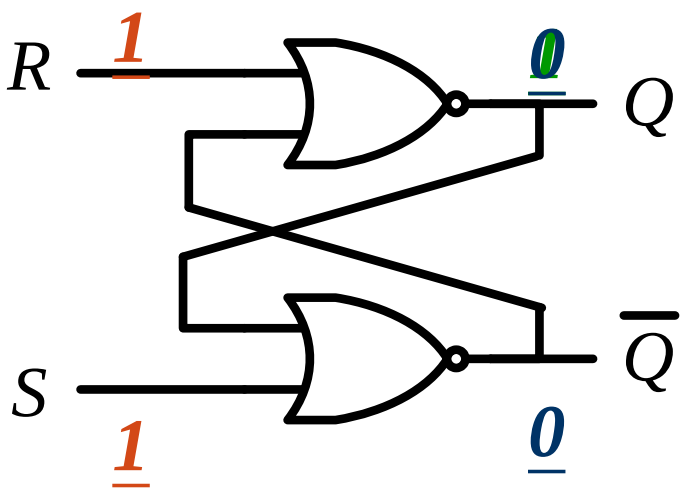
- *SR*



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$

Klopný obvod

- *SR*



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$
1	1	1	0	0	$Q = Q'$