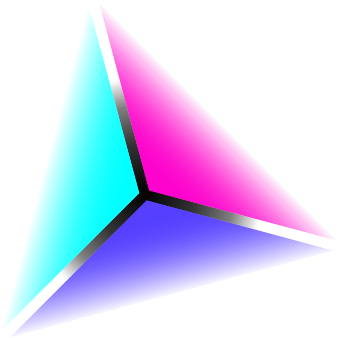
Struktura a architektura počítačů

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické

Aritmetické operace a jejich realizace, pohyblivá řádová čárka

BI-SAP, březen 2021



Obsah

- Zobrazení čísel se znaménkem v počítači
  - Přímý, aditivní a doplňkový kód a operace sčítání a odčítání včetně přetečení
- Realizace aritmetických operací vč. optimalizace
- Čísla s pohyblivou řádovou čárkou
  - Zobrazení v řádové mřížce
  - Provádění základních aritmetických operací
  - Normalizovaný tvar, skrytá jednička

Řádová mřížka (opakování)

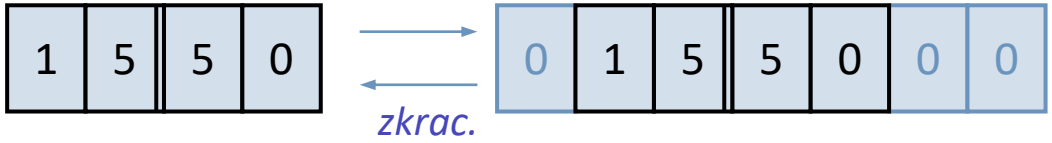
- **Zobrazení čísel na počítači je limitováno** (rozsahem registrů, paměť. míst, apod.)
- **Řádová mřížka určuje formát zobrazitelných čísel** (tj. definuje nejvyšší řád *n* a nejnižší řád *-m*)

- Příklad řádových mřížek pro *l* = 4

Aritmetické operace v ř.m.: chyby

- Někdy se lze chybám vyhnout změnou délky ř.m.
  - **rozšiřování** × **zkracování** ř.m.

Př.



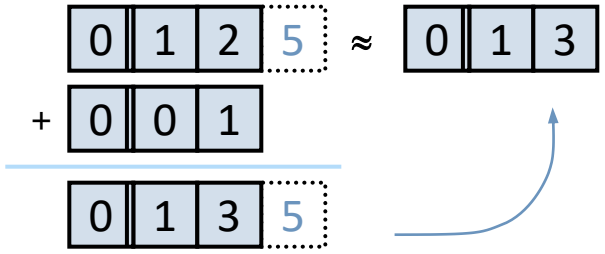
(Rozšíření ř.m. může být realizováno např. uložením do dvou registrů)

- Při ztrátě přesnosti můžeme velikost chyby ovlivnit způsobem **zaokrouhlení**
  - zaokrouhlení nahoru/dolů
  - zaokrouhlení s preferencí sudé číslice
  - zaokrouhlení s preferencí většího čísla

Zaokrouhlení v ř.m.

- **Zaokrouhlení dolů** (oříznutí)

- **Zaokrouhlení nahoru**



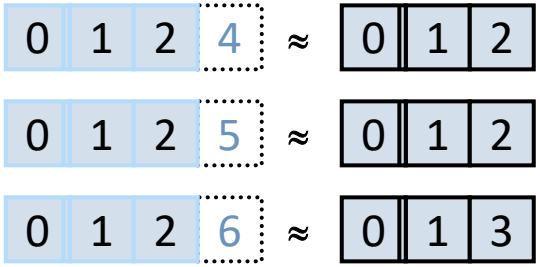
Jak to realizovat v HW?

Zaokrouhli 0,1011
na 3 místa
0,1011
+ 0,0001
0,1100

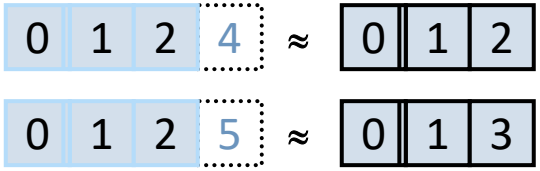
$0{,}6875_{10}$ zaokrouhleno správně na $0{,}688_{10}$
Zde $0{,}75_{10}$
zatímco dolu .... 0,101 tzn. jen $0{,}625_{10}$

… zaokrouhlení

- **Zaokrouhlení s preferencí sudé číslice**

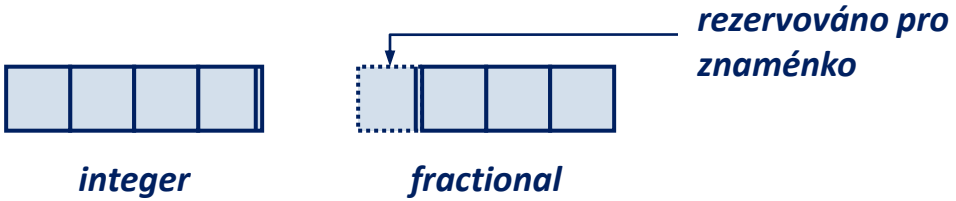


- **Zaokrouhlení s preferencí většího čísla**

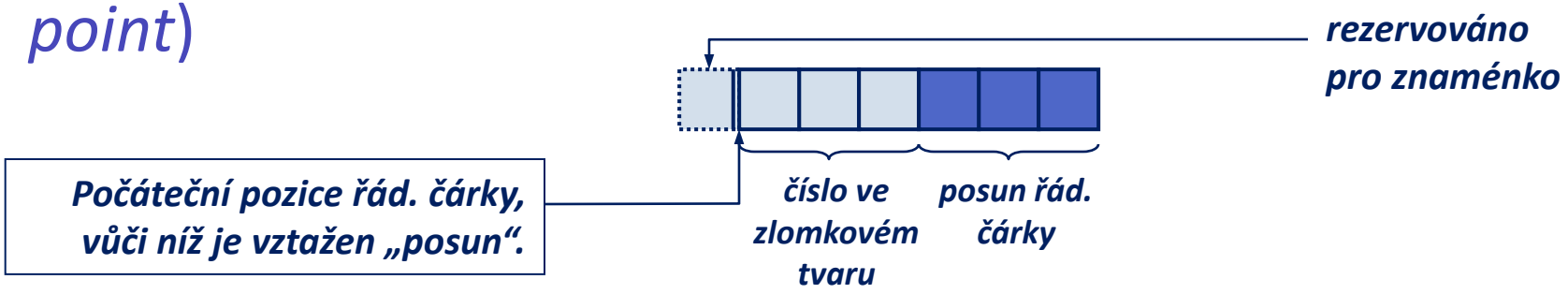


Řádová čárka vzhledem k ř.m.

- Pevně definovaná pozice ⇒ **čísla s pevnou řádovou čárkou** (*fixed-point*)
  - nejpoužívanější



- Řádová čárka je definována posunem vůči definované pozici ⇒ **čísla s pohyblivou řádovou čárkou** (*floating-point*)



Zobrazení čísel se znaménkem

- Standardní polyadické soustavy $\Rightarrow$ pouze nezáporná čísla
- Zobrazení i záporných čísel $\Rightarrow$ **číselné kódy**
  - popisují transformaci z omezené množiny celých čísel do omezené množiny nezáporných čísel
- Nejpoužívanější číselné kódy:
  - **přímý**
  - **aditivní**
  - **doplňkový**

Přímý kód

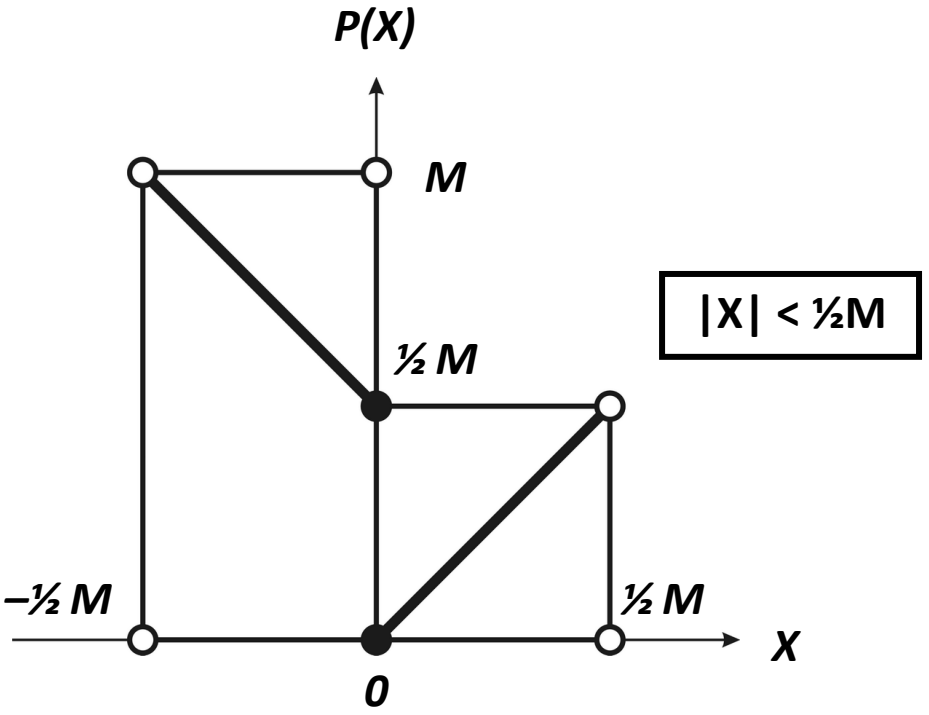
Opakování:

- Nejvyšší řád ř.m. představuje znaménko, zbytek ř.m. je absolutní hodnota
- Znaménko reprezentováno číslicí:

+...0, -...1

- Znázornění zobrazení:

+/-	absolutní hodnota



… přímý kód

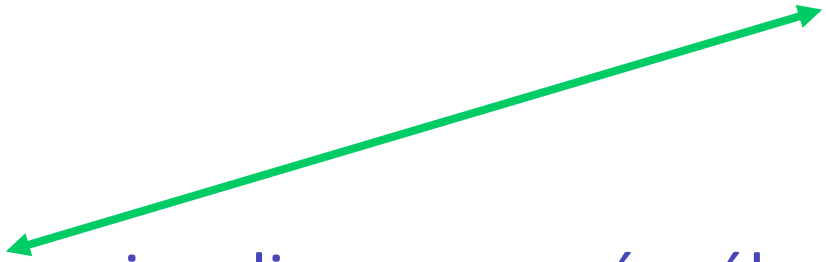
M = 1000 ... tzn. 3bitová čísla

X	*P(X)*			
+0	0	0	0	←kladná nula
+1	0	0	1	
+2	0	1	0	
+3	0	1	1	
-0	1	0	0	←záporná nula
-1	1	0	1	
-2	1	1	0	
-3	1	1	1	

Sčítačka pro nezáporná čísla, popis

- p_i .... přenos do řádu i .....*carry*
- q_i .... přenos z řádu i
- $M = 8 = 2^3 = 1000_2 = (111 + 1)_2$

$$S = \begin{cases} A + B & \text{je-li } q^* = 0 \\ A + B - M & \text{je-li } q^* = 1 \end{cases}$$



- *zde:* q^* .... signalizace nesprávného výsledku

Poznámka: přenos (carry) je vždy výstup z bloku sčítačky, v některých případech (tzn. zde) je použitý pro signalizaci nesprávného výsledku (tzn. přeplnění, overflow)

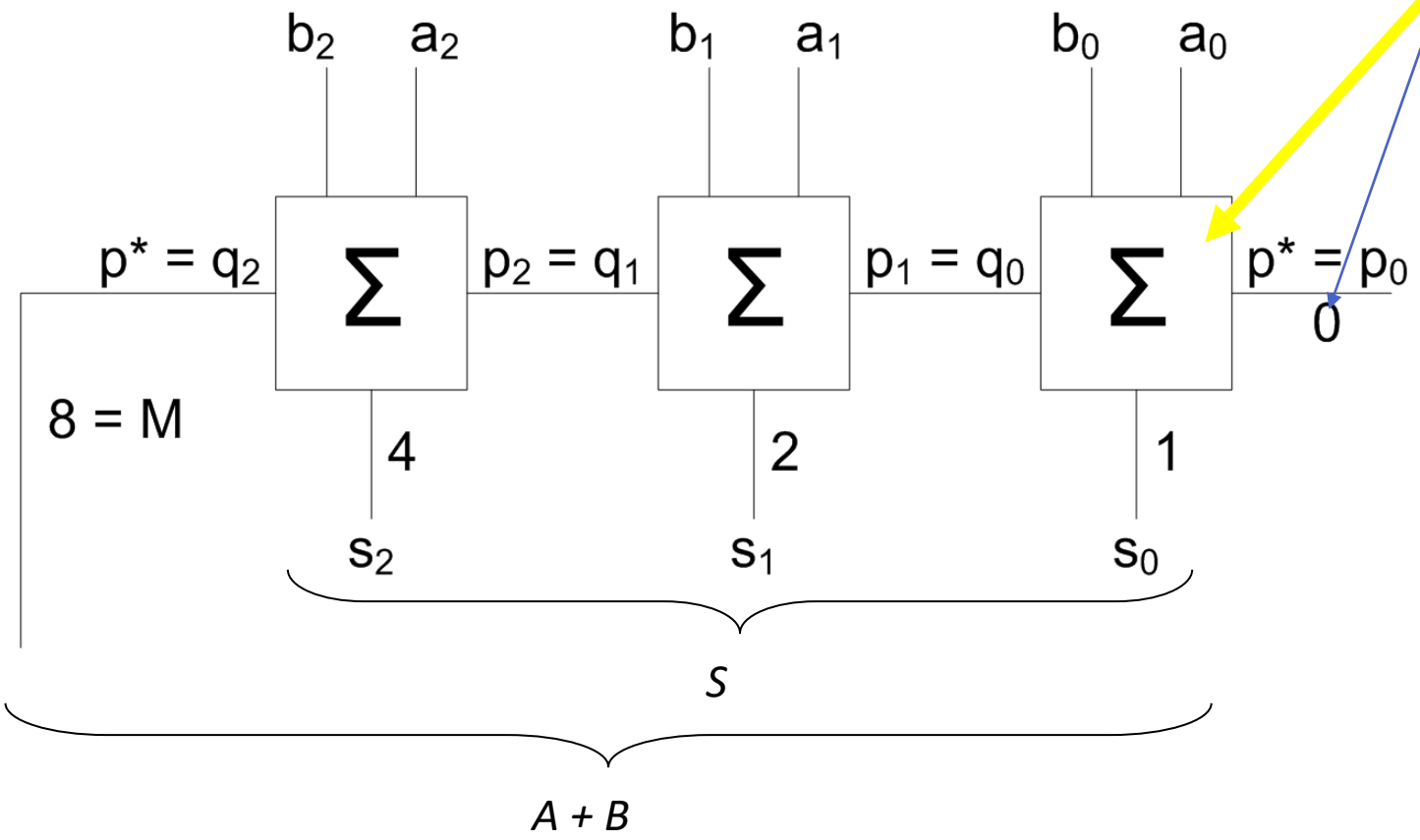
Sčítačka pro nezáporná čísla

S = A + B …. jen někdy

A= …$a_2a_1a_0$

B= …$b_2b_1b_0$

Stačila by půlsčítačka



Odčítání

- Pracuji zvlášť se znaménkem a absolutní hodnotou

- Absolutní hodnota je nezáporné číslo:

Příklad pro 3 bitová nezáporná čísla:

$$B=101 \quad \overline{B}=010$$

$$B + \overline{B} = 111 = 1000 - 1 = M - 1$$

$$-B = \overline{B} + 1 - M$$

$$\boxed{A - B = A + \overline{B} + 1 - M}$$

Aby byl výsledek správně, musí být možné odečíst modul (tj. přenos/carry)!

Odčítání

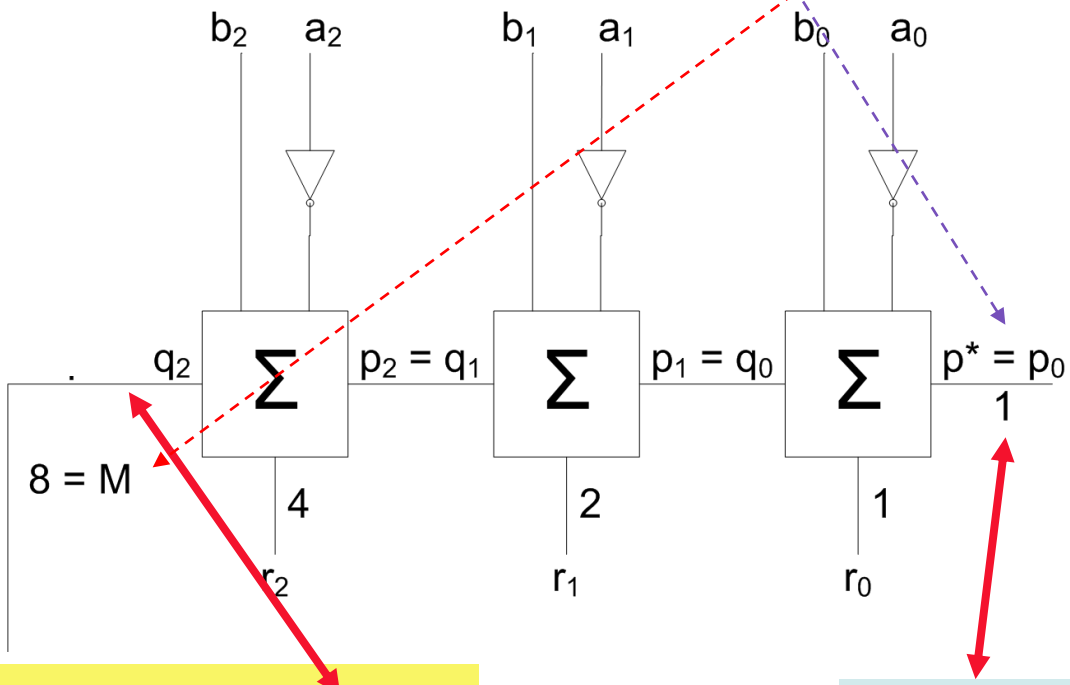
sčítání .... přenos q^* .....carry
odčítání .... výpujčka v^* .....borow

$$\overline{q^*} = v^*$$

$q^* = 1$		$v^* = 0$	$\Leftrightarrow$	$B - A \geq 0$
$q^* = 0$	$\Leftrightarrow$	$v^* = 1$	$\Leftrightarrow$	$B - A < 0$

Odčítačka pro nezáporná čísla

$$A - B = A + \overline{B} + 1 - M$$



$R = B - A$, je-li $q_2 = 1$

$p^* = 1$ .... horká jednička
hot one

Sčítání a odčítání v přímém kódu

A + B, A – B, výsledek ulož do A

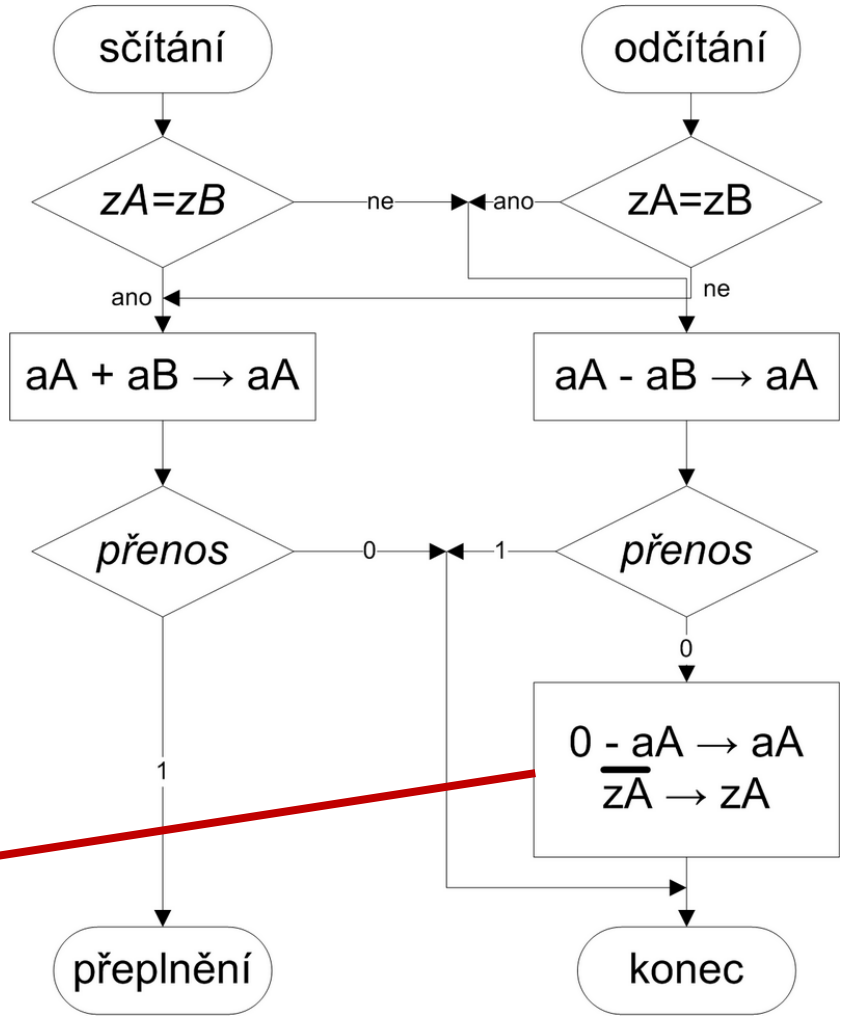
kde

A ~ (zA, aA),

B ~ (zB, aB)

z – znaménko,

a – absolutní hodnota



Pokud nevyjde přenos, vyšel záporný výsledek, který neumíme zobrazit, proto je třeba negovat znaménko a odečíst výsledek od 0.

Sčítačka-odčítačka pro nezáporná čísla

- složitější řízení, protože detekce přeplnění záleží na operaci
- sčítání ... přeplnění (zde přenos) pro $q^*=1$
- odčítání .... přeplnění (zde výpůjčka) pro

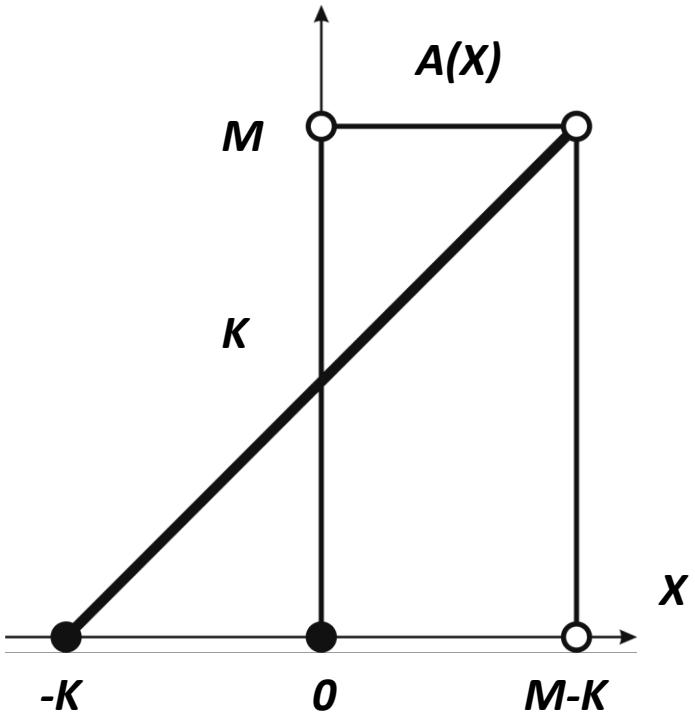
  $v^*=1$ ..... $q^*=0$

Aditivní kód

- Též označovaný jako „kód s posunutou nulou“
- Formální definice: $\mathcal{A}(X) = X + K$ pro $-K \leq X < M - K$
- K – vhodná konstanta

  často se volí: $K = ½\ M$

Označení znaménka není explicitní jako u přímého kódu → je „implicitní“ součástí zobrazovaného čísla jako u dopňkového kódu.



Příklady – aditivní kód

$-25_{10} \xrightarrow[K=5000]{\mathcal{A}}$ | 4 | 9 | 7 | 5 |

$+101_2 \xrightarrow[K=1000_2]{\mathcal{A}}$ | 1 | 1 | 0 | 1 |

$+0{,}05_{10} \xrightarrow[K=1{,}000]{\mathcal{A}}$ | 1 | 0 | 5 | 0 |

$-0{,}11_2 \xrightarrow[K=1{,}000_2]{\mathcal{A}}$ | 0 | 0 | 1 | 0 |

Převod aditivní ↔ doplňkový
Přičtení aditivní konstanty (+ K)

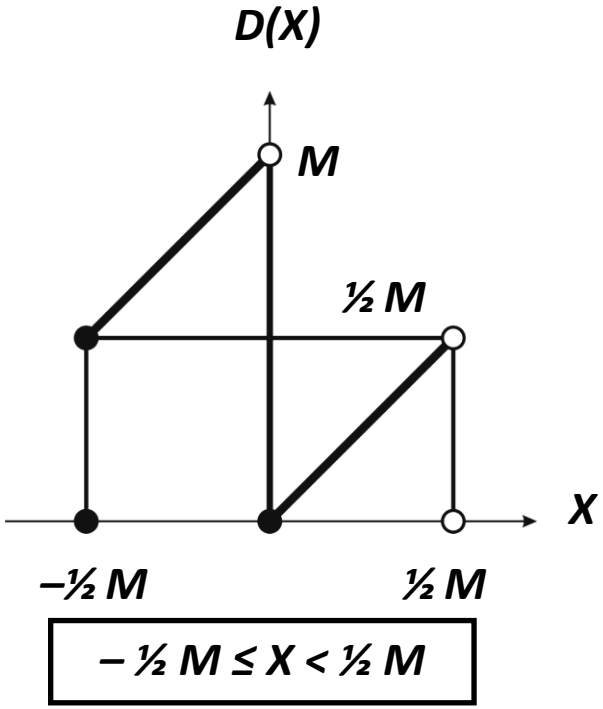
$\mathcal{A}(5)$: 1101
+ K: 1000
$\mathcal{D}(5)$: 0101
+K: 1101

$\mathcal{A}(-0{,}11)$: 0,010
+ K: 1,000
$\mathcal{D}(-0{,}75)$: 1,010 ..... – 0,110
+ K: 1,000
0,010

Doplňkový kód

Opakování ....

Definice: $\mathcal{D}(X) = \begin{cases} X, & \text{je-li } X >= 0 \\ M + X, & \text{je-li } X < 0 \end{cases}$



Příklad – napsat všechna 3 bitová čísla
(M = 1000, ε = 1, l = 3)

X	D(X)		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
-4	1	0	0
-3	1	0	1
-2	1	1	0
-1	1	1	1

Znaménko je určeno prvním bitem zleva,
ale tento bit je organickou součástí obrazu !!!

Sčítání a odčítání v doplňkovém kódu

		D(A) + D(B)	D(A + B)
1	$A \geq 0\ B \geq 0$	$A + B$	$A + B$
2	$A \geq 0\ B < 0$	$A + B + M$	$A + B$
	$A < 0\ B \geq 0$		$A + B + M$
3	$A < 0\ B < 0$	$A + B + M + M$	$A + B + M$

$$D(A + B) = \begin{cases} D(A) + D(B) \\ D(A) + D(B) - M \end{cases}$$

Sečtou se obrazy a ignoruje se přenos !!!

Odčítání:

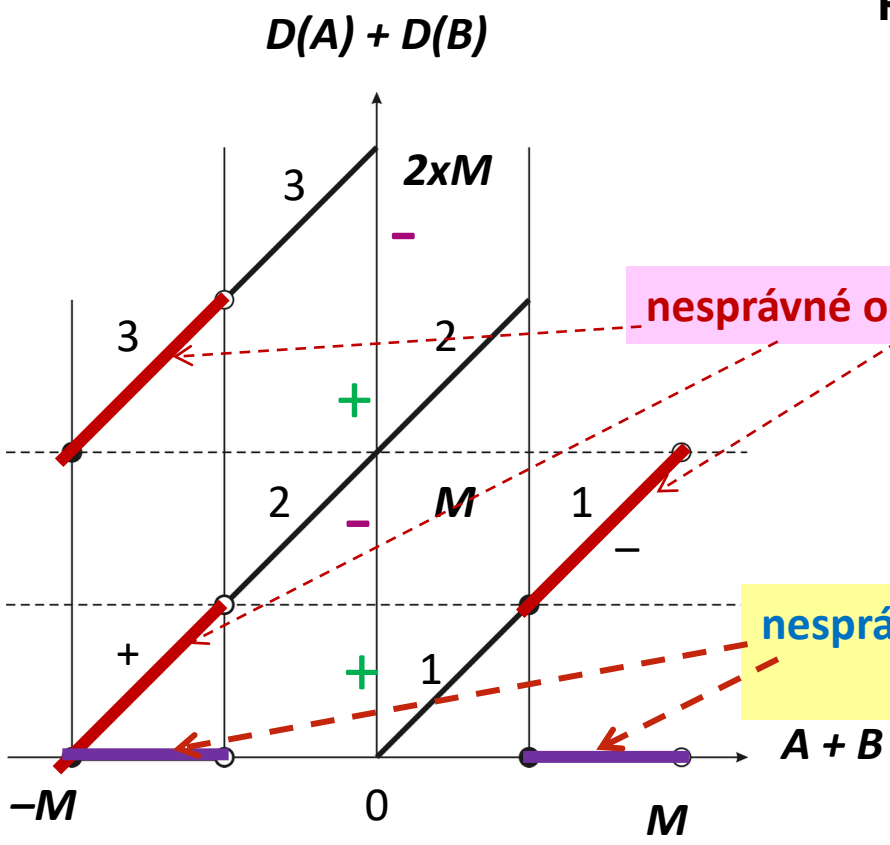
$$\mathcal{D}(-B) = \overline{\mathcal{D}(B)} + 1$$

$$A - B = \mathcal{D}(A) + \overline{\mathcal{D}(B)} + 1$$

detekce přeplnění je stejná jako u sčítání

Přeplnění

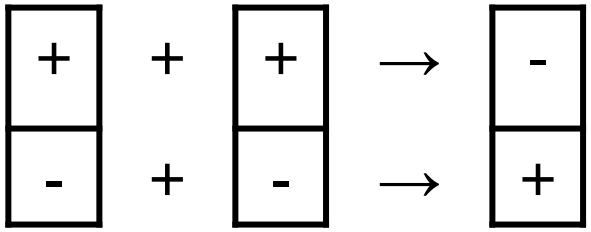
Přeplnění (overflow) není přenos (carry) !!!!!



nesprávné obrazy

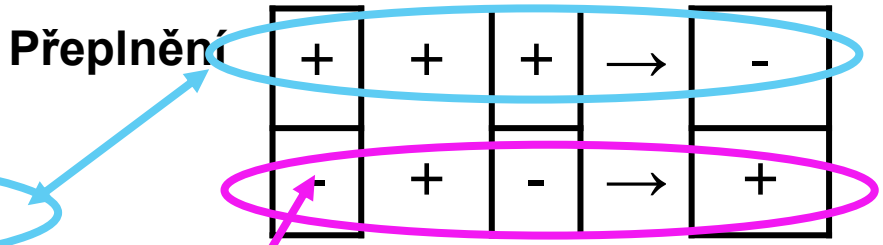
nesprávné součty (nevejde se do řádové mřížky)

Přeplnění:



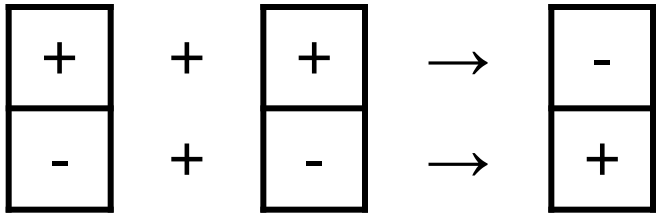
Přeplnění … podle bloku sčítačky pro nejvyšší řád

a	b	p	q	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



Detekce přeplnění

Přeplnění:



tzn. v nejvyšším řádu sčítačky bude:
a=0 b=0 s=1
nebo
a=1 b=1 s=0

$p \neq q$

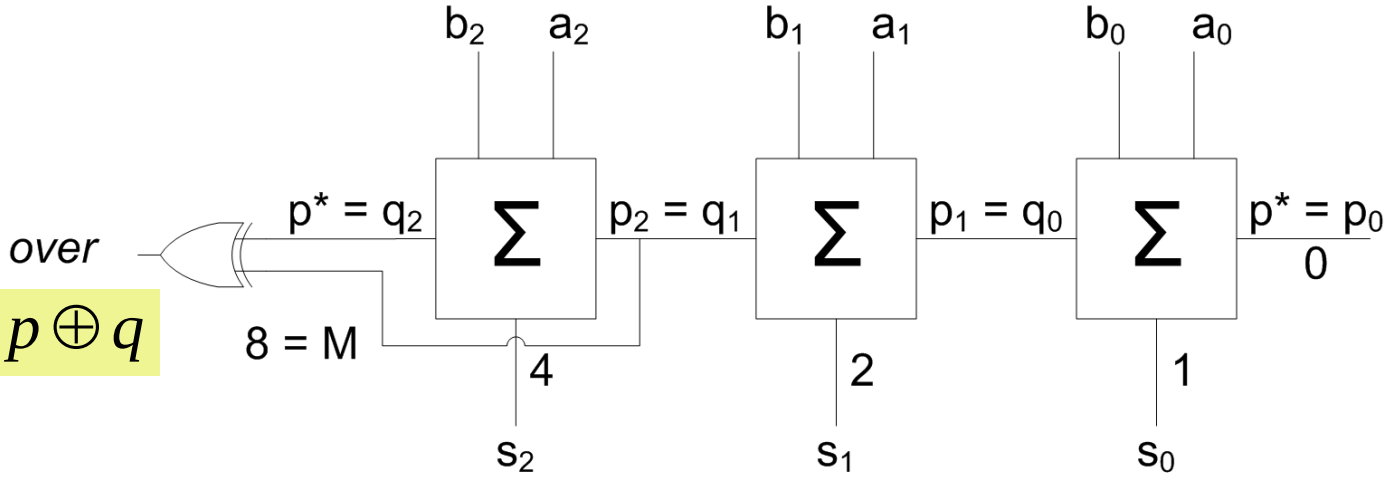
$$over = p \oplus q$$

nebo také:

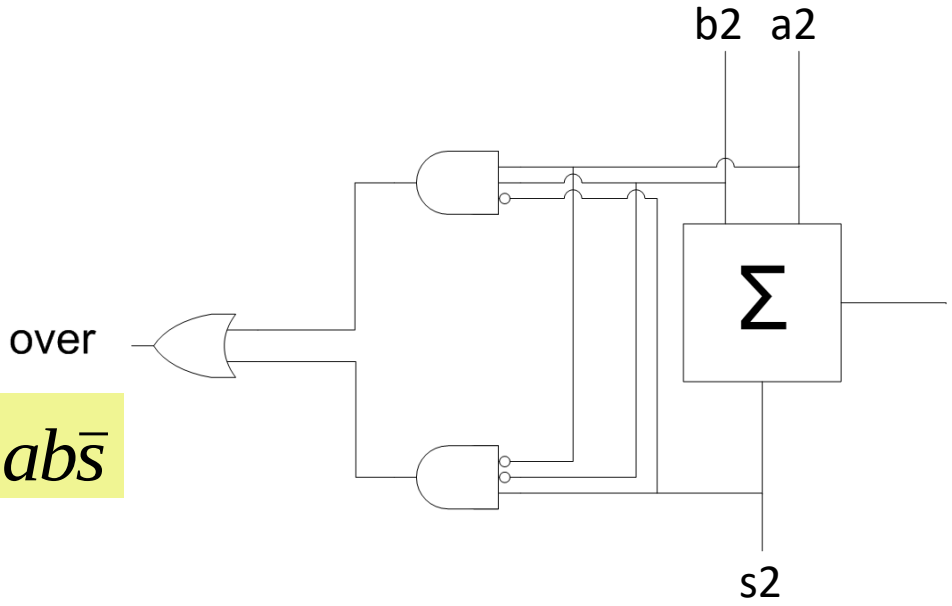
$$over = \bar{a}\bar{b}s + ab\bar{s}$$

s_i	a	b	p	q	S
0	0	0	0	0	0
1	0	0	**1**	**0**	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	**0**	**1**	0
7	1	1	1	1	1

Realizace detekce přeplnění - overflow



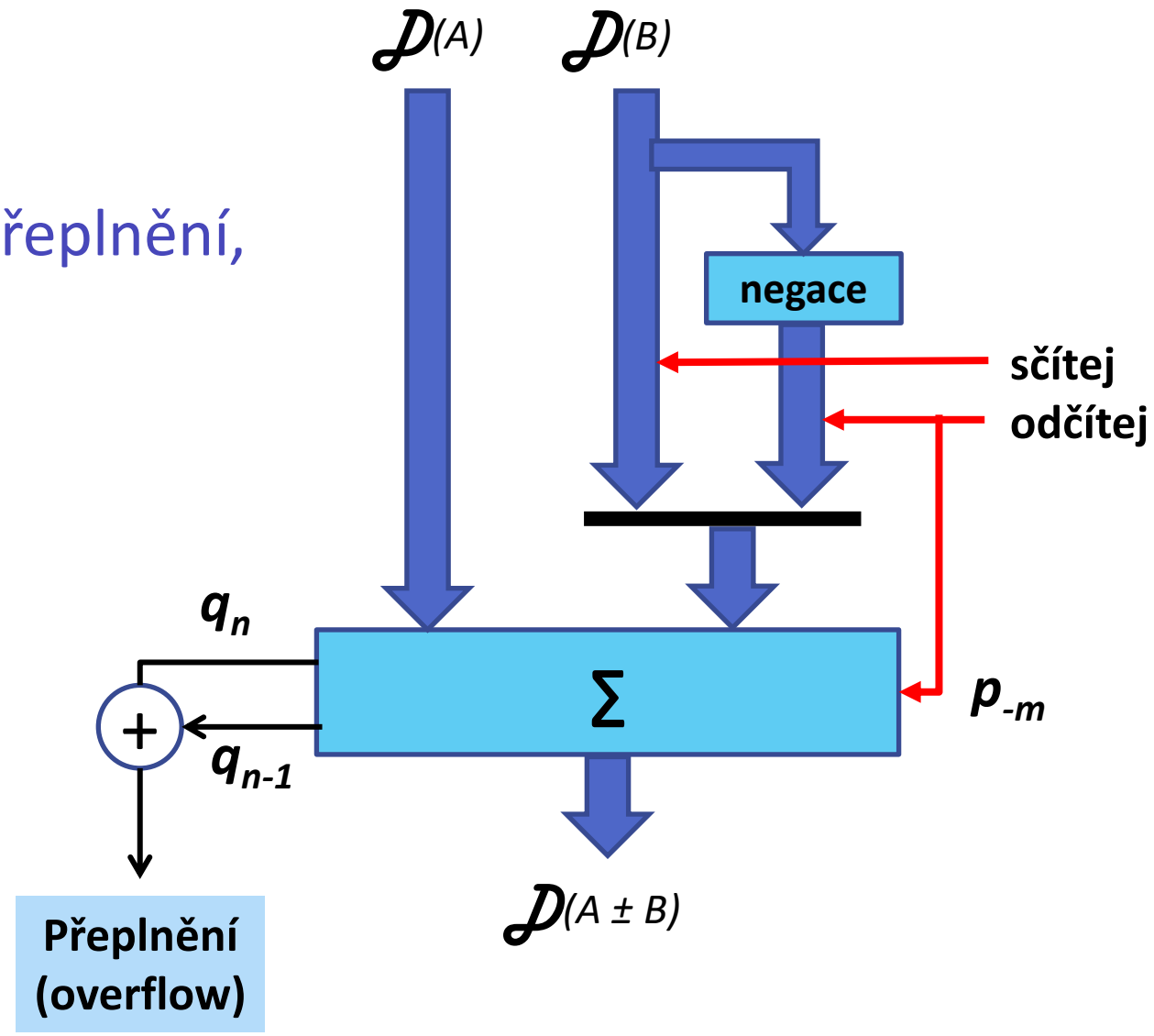
$$over = p \oplus q$$



$$over = \bar{a}\bar{b}s + ab\bar{s}$$

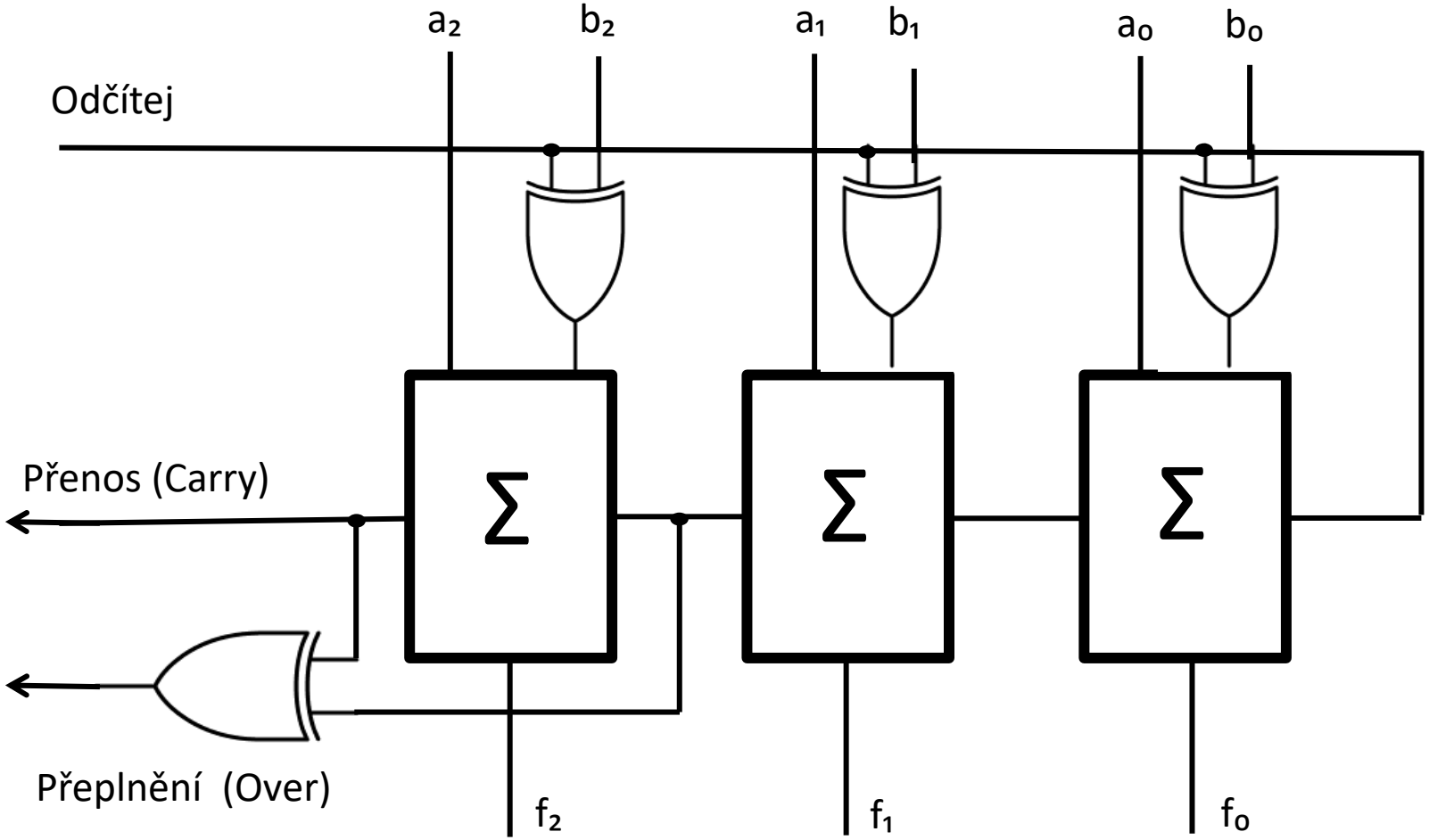
Sčítačka-odčítačka v doplňkovém kódu

včetně detekce přeplnění,
princip blokově



Sčítačka-odčítačka v doplňkovém kódu

včetně detekce přeplnění, realizace z hradel (bloky Σ už známe)



Sčítačka-odčítačka -demo

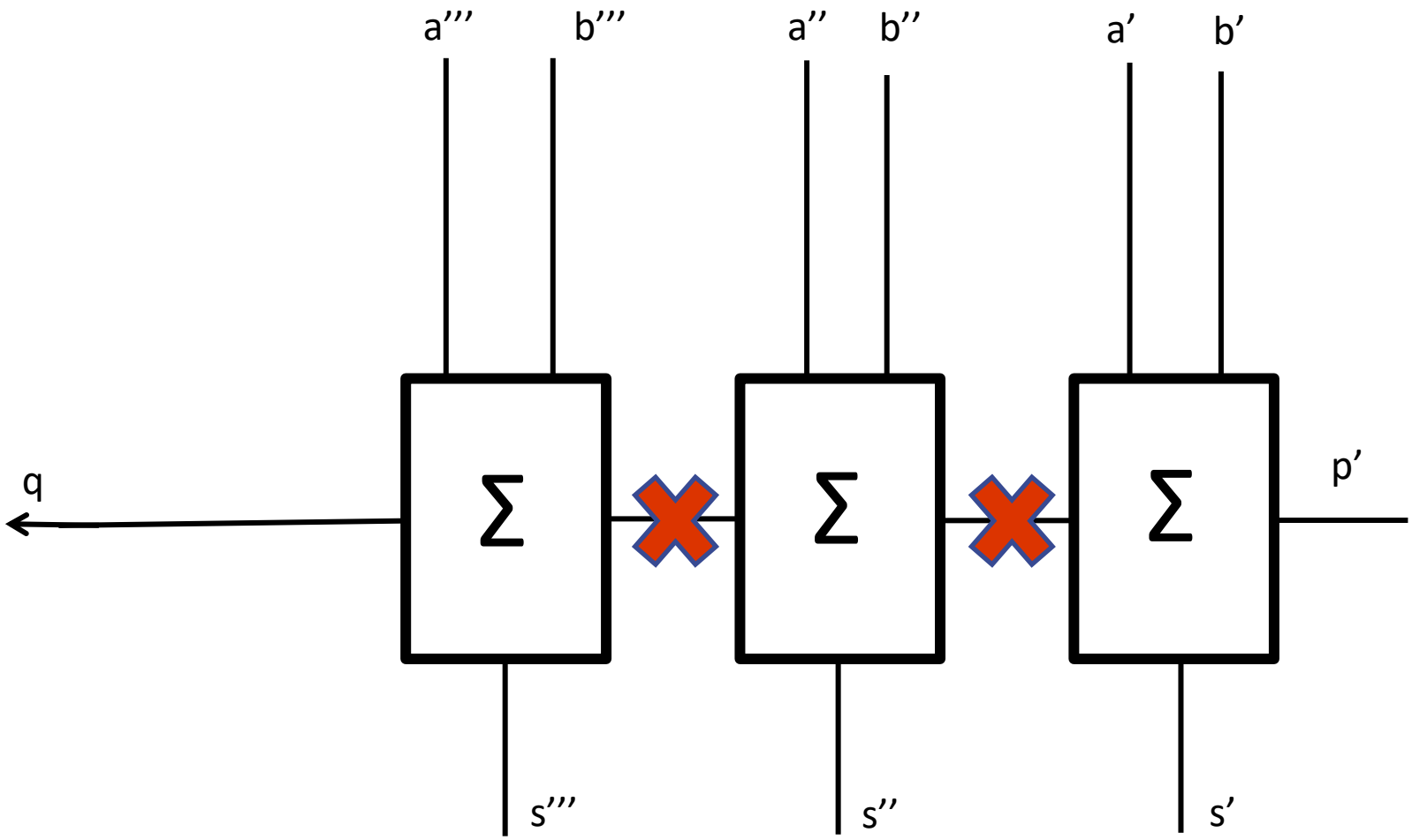
http://www.ecs.umass.edu/ece/koren/arith/simulator/Add/ripple/ripple.html

Problém: správný součet vypadne až po průchodu přes všechny bloky sčítačky

Jak to zrychlit?

- Návrh vícebitové sčítačky dvouúrovňově (tzn. že obvod pro dvoubitovou sčítačku bude mít 5 vstupů a 3 výstupy … které?)
- Přenos se nemusí uplatnit vždy: jen když se v bloku generuje nebo se přes blok šíří ...
- Jak to předpovědět?

Zrychlení činnosti sčítačky



Přerušení vazeb, tzn. zamezení šíření přenosů přes všechny bloky.
Ale bude to znamenat přidání další logiky …. Jak to odvodit ze vstupů?

Predikce přenosů

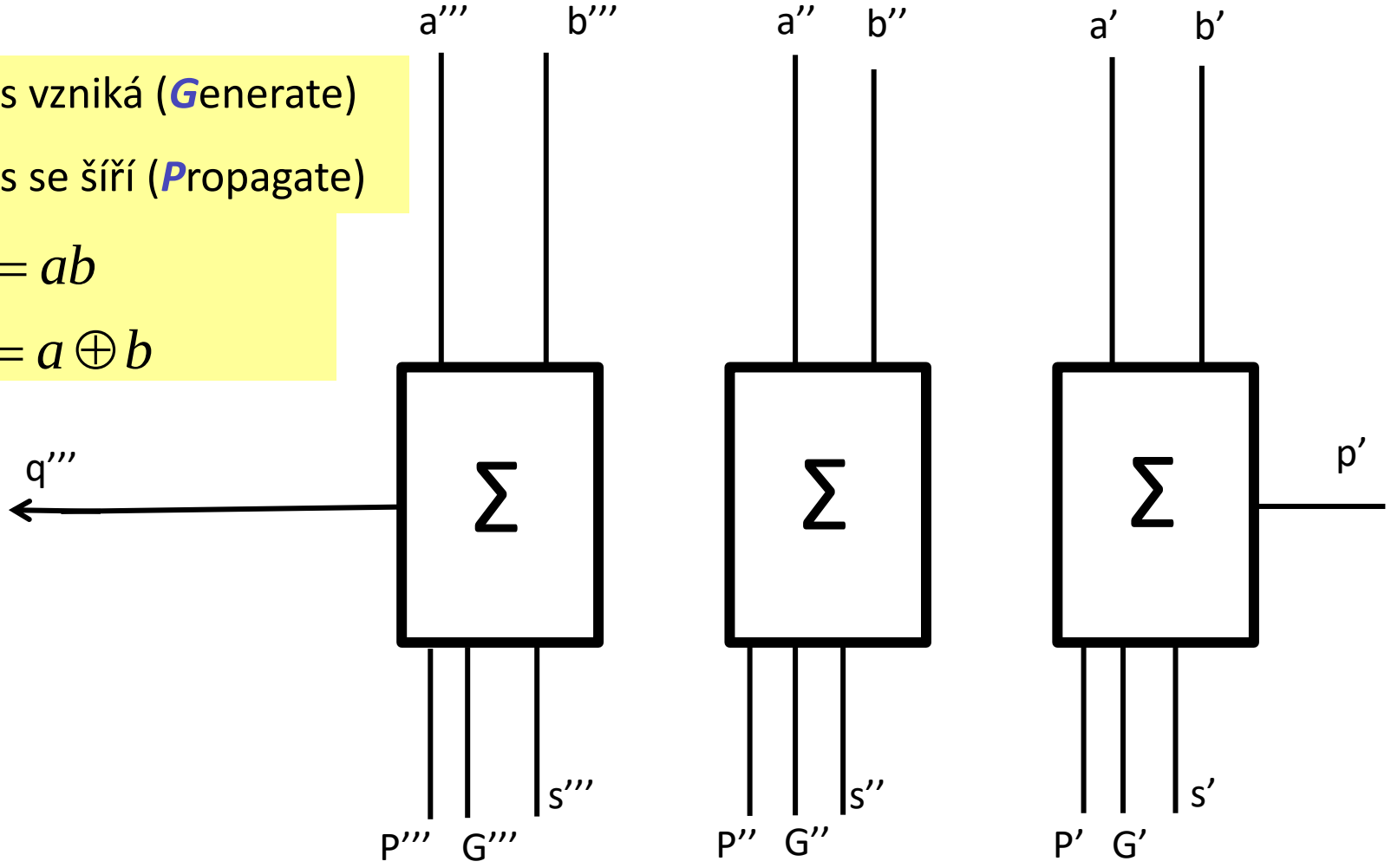
Analýza toho, kdy se s přenos uplatní v následujícím bloku (blocích):

Přenos vzniká (*G*enerate)

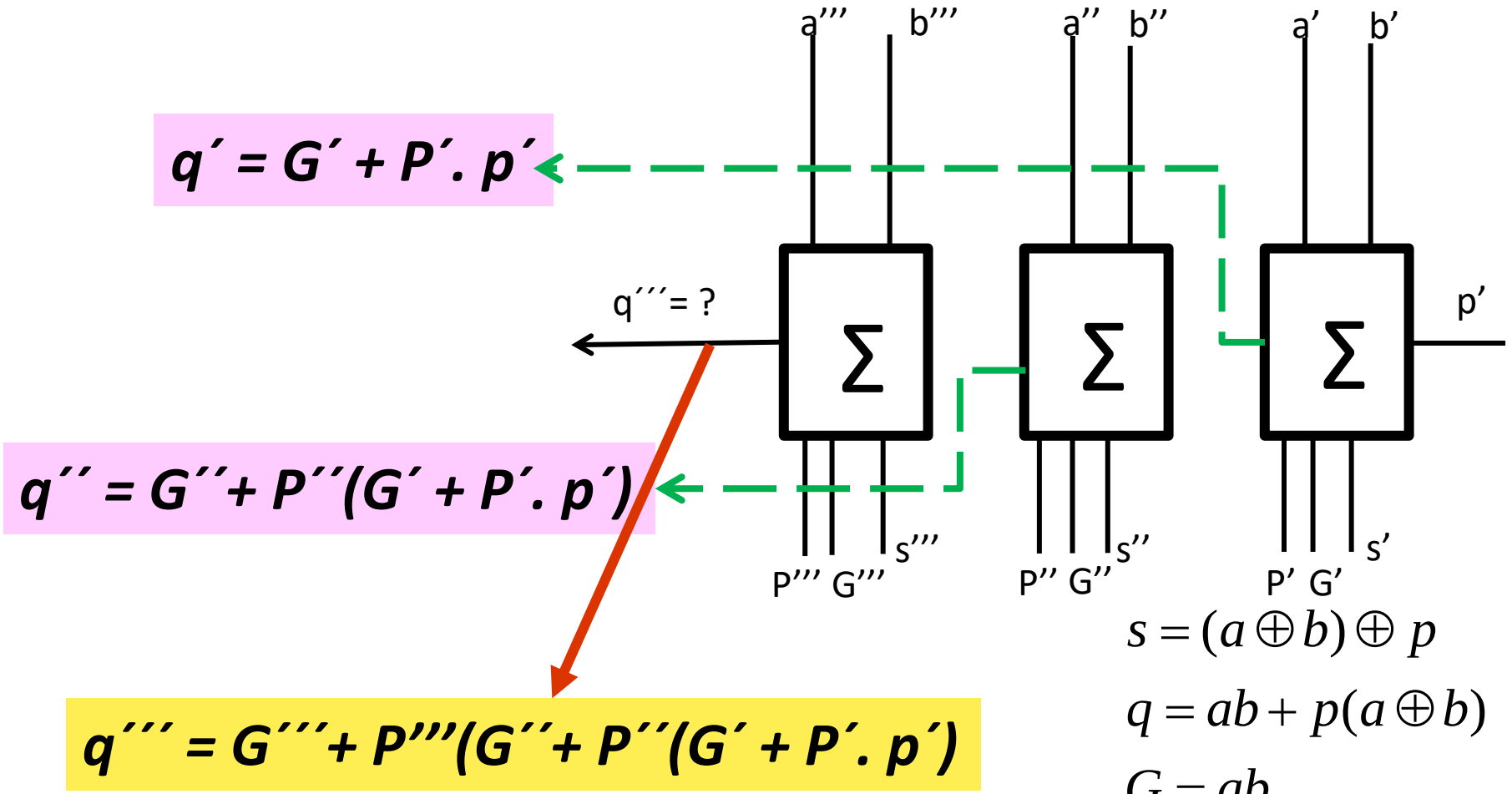
Přenos se šíří (*P*ropagate)

$$G = ab$$

$$P = a \oplus b$$



Predikce přenosů



$$s = (a \oplus b) \oplus p$$

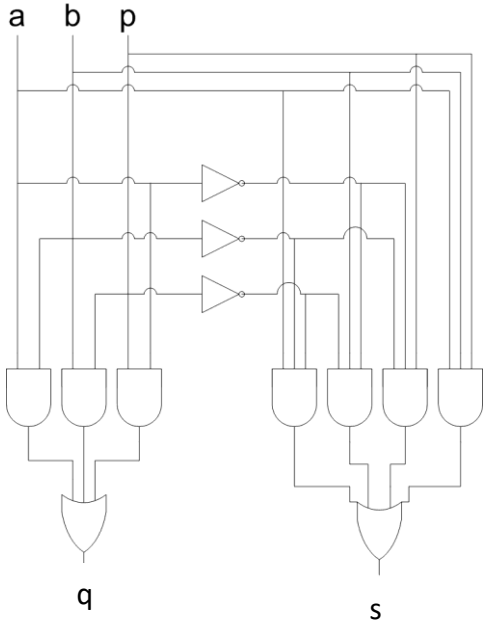
$$q = ab + p(a \oplus b)$$

$$G = ab$$

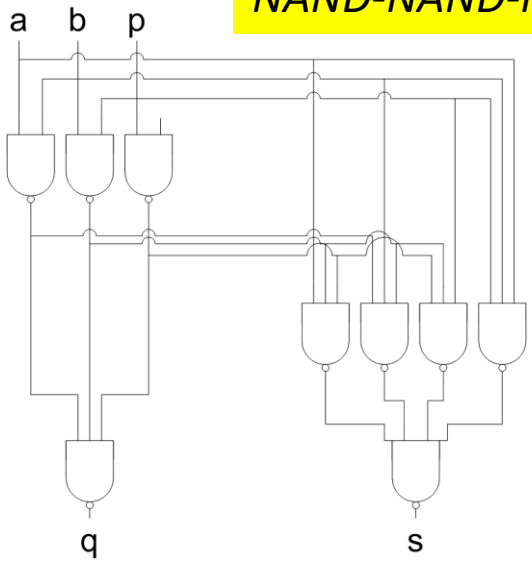
$$P = a \oplus b$$

Realizace (full adder)

NOT-AND-OR

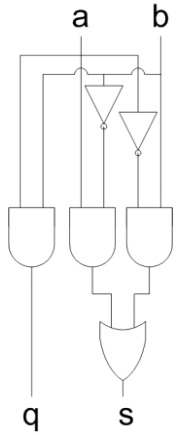


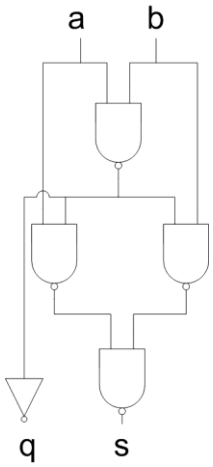
NAND-NAND-NAND



FA:
Full **A**dder

poloviční sčítacka (half adder)

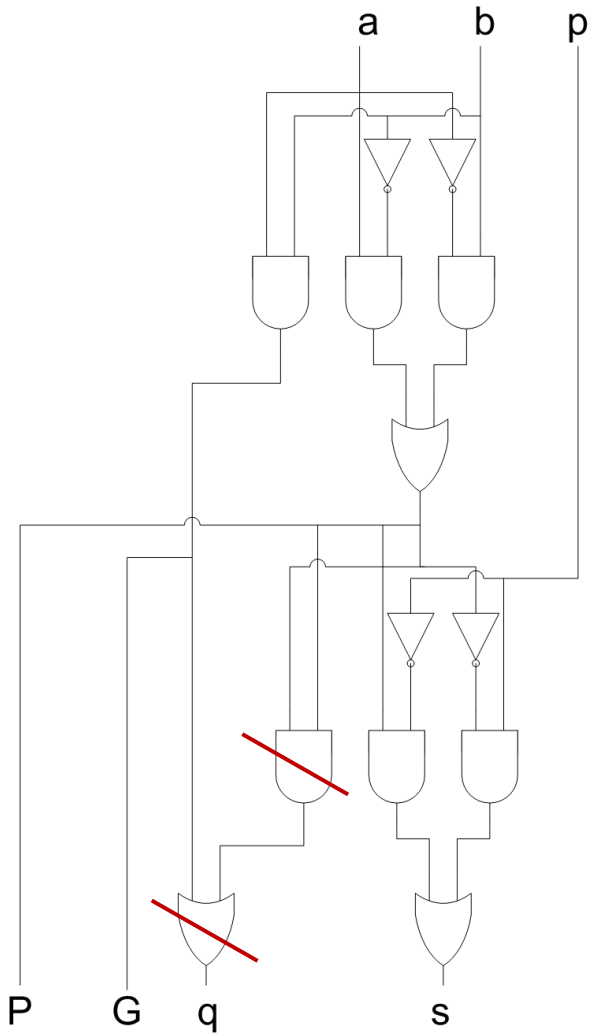




HA
Half **A**dder
S = a xor b
q = a.b

Predikce přenosů

NOT-AND-OR



$$s = (a \oplus b) \oplus p$$

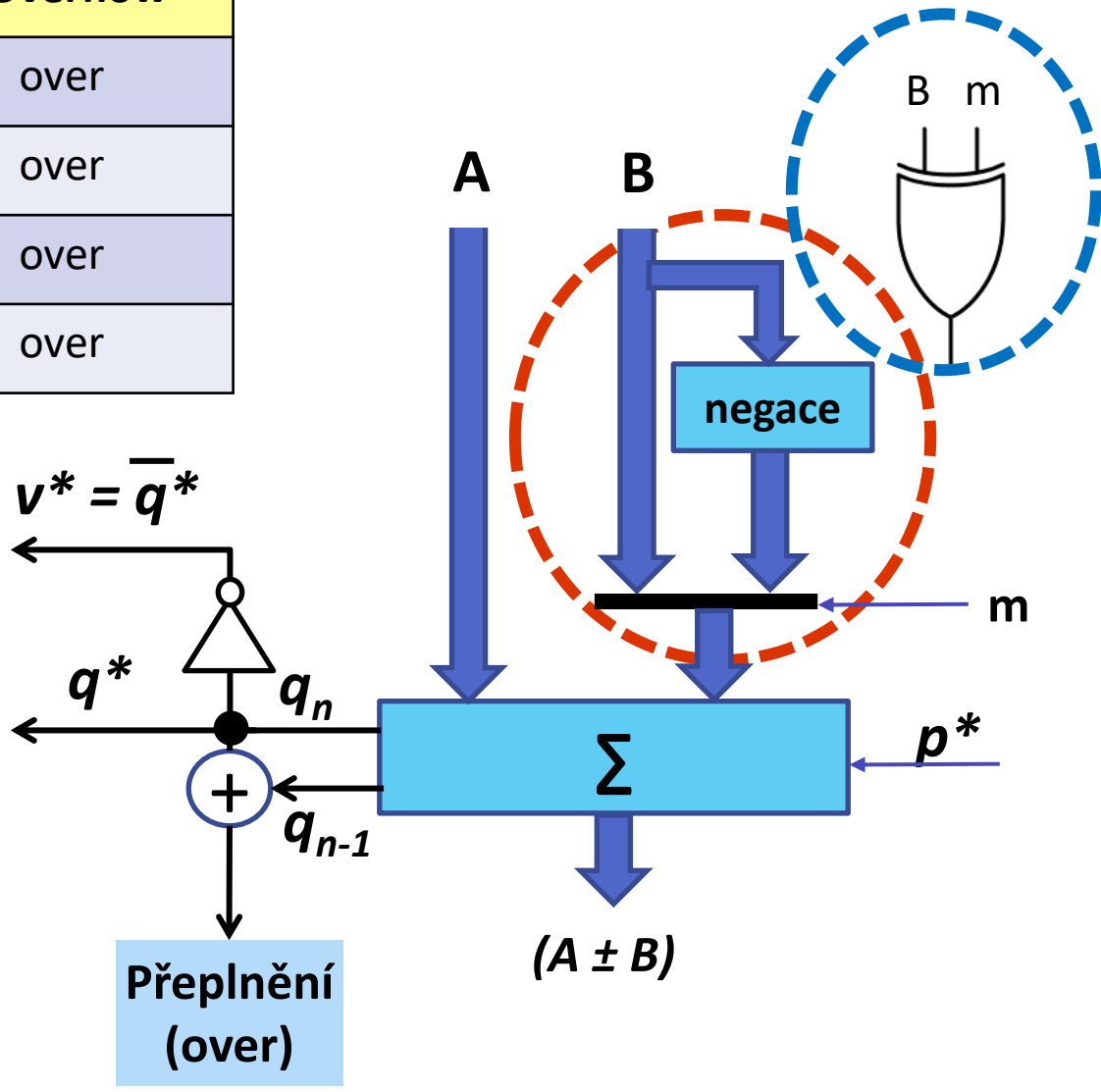
$$q = ab + p(a \oplus b)$$

$$G = ab$$

$$P = a \oplus b$$

Sčítačka-odčítačka v procesoru

	m	p*	Carry	Overflow
ADD	0	0	q*	over
ADC	0	Carry	q*	over
SUB	1	1	$\overline{q^*}$	over
SBB	1	$\overline{Carry}$	$\overline{q^*}$	over



Realita v procesoru

	m	p*	Carry	Overflow
ADD	0	0	q*	over
ADC	0	Carry	q*	over
SUB	1	1	$\overline{q^*}$	over
SBB	1	$\overline{Carry}$	$\overline{q^*}$	over

Hardware (ALU, kombinační logika, řadič) realizuje operaci, nastaví příznaky (přeplnění – overflow, přenos - carry, a i další informace o výsledku, např. znaménko, nulový výsledek), které pak software interpretuje podle aktuální potřeby.

Výsledek sčítání a odčítání (to co vypadne z ALU) je stejné, záleží na interpretaci, tzn. zda jde o číslo nezáporné nebo se znaménkem

Příklady pro 8 bitová čísla

Sečtěte čísla 75_{16} + BC_{16} v 8 bitovém procesoru

7 5	
+ B C	
1 3 1	

```
  0111 0101
+ 1011 1100
-----------
1 0011 0001
```

Interpretace:

- čísla nezáporná (celá): součet je 31 s detekcí nesprávného výsledku (1 v carry), ale včetně přenosu lze na 9 bitech získat 131 (dva registry)
- čísla v doplňkovém kódu: součet je 31 a je správně, protože sčítáme +75 + (- 44), overflow = 0
- čísla v přímém kódu: vlastně odčítáme, ale 7 bitů: 75 – 3C = + 39

```
  0111 0101
+ 1011 1100
-----------
          ?
```

```
  111 0101
- 011 1100
----------
         ?
```

```
  111 0101
+ 100 0011
+       1
---------
1 011 1001
```

vyšel přenos, tzn. že výsledek je správně!!

Na 8b: + 39 .... **39**

Příklady pro 8 bitová čísla

Sečtěte čísla $4A_{16} + E5_{16}$ v 8 bitovém procesoru

4 A
+ E 5
1 2 F

```
  0100 1010
+ 1110 0101
-----------
1 0010 1111
```

Interpretace:

- čísla nezáporná (celá): součet je 2F s detekcí nesprávného výsledku (1 v carry), ale včetně přenosu lze na 9 bitech získat 12F (dva registry)
- v doplňkovém kódu: součet je 2F a je správně, protože sčítáme +4A + (-1B), overflow = 0
- v přímém kódu: odčítáme 7 bitů: 4A – 65 = **-**1B, výsledek na 8b: 9B

```
  0100 1010
+ 1110 0101
-----------
          ?
```

```
  100 1010
- 110 0101
---------
        ?
```

```
  100 1010
+ 001 1010
+        1
---------
0 110 0101
```

```
  001 1010
+        1
---------
  001 1011
```

nevyšel přenos, tzn. že výsledek není správně – musíme ho odečíst od 0 a negovat znaménko

Příklady pro 8 bitová čísla

Sečtěte čísla $8A_{16} + 35_{16}$ v 8 bitovém procesoru

8 A
+ 3 5
B F

```
  1000 1010
+ 0011 0101
-----------
0 1011 1111
```

Interpretace:

- čísla nezáporná (celá): součet je BF a nulový přenos (carry=0),
- v doplňkovém kódu: součet je BF (-41) a je správně, protože sčítáme -76 + 35, overflow = 0
- v přímém kódu: odčítáme 7 bitů: -0A + 35 = + 2B, na 8b: 2B

```
  1000 1010
+ 0011 0101
-----------
          ?
```

```
  000 1010
- 011 0101
---------
        ?
```

```
   000 1010
+  100 1010
+         1
-----------
0 101 0101
```

```
   010 1010
+         1
-----------
   010 1011
```

nevyšel přenos, tzn. že výsledek není správně – musíme ho odečíst od 0 a negovat znaménko A

Pohyblivá řádová čárka

- Čísla s pevnou řádovou čárkou mají výrazně omezený rozsah

  Př. $z = 2$, délka ř.m. $l = 32$ (tj. 32-bitové číslo)

max. celé číslo	… $A < 2^{32} < 5 \cdot 10^{9}$
min. zlomkové číslo	… $A > 2^{-32} > 2 \cdot 10^{-9}$

- Pro zvětšení rozsahu přidáme exponent e

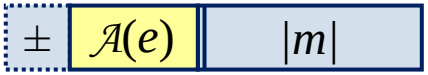
  $X \cdot z^e$ … odpovídá posunu řádové čárky v čísle X o e

  $\Rightarrow$ čísla s pohyblivou řádovou čárkou

...pohyblivá řádová čárka

- Řádová mřížka má 2 části (podmřížky):
  - **mantisa** (*m*) - informace o „hodnotě“ čísla, často zlomkový tvar
  - **exponent** (*e*) - informace o pozici řád. čárky, celé číslo
- *m* i *e* používají kódy pro zobrazení čísel se znaménkem
- Ukázky možných formátů ř. m.

Př. $(-25 \cdot 10^2)_{10}$

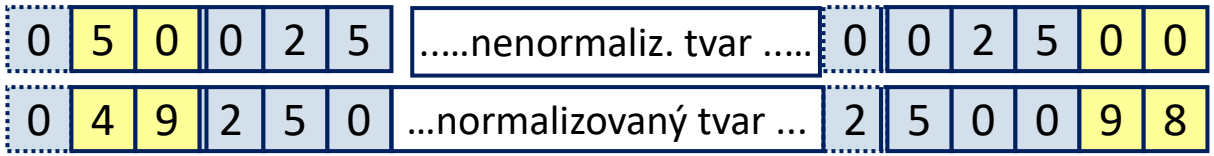


Př. $(0{,}02 \cdot 10^{-5})_{10}$

...pohyblivá řádová čárka

- **Normalizovaný tvar**
  - je **tvar čísla, kdy už nelze mantisu posunout více doleva**
  - zjednodušuje aritmetické operace
- Normalizovaný tvar operandů nezaručí normalizovaný tvar výsledku

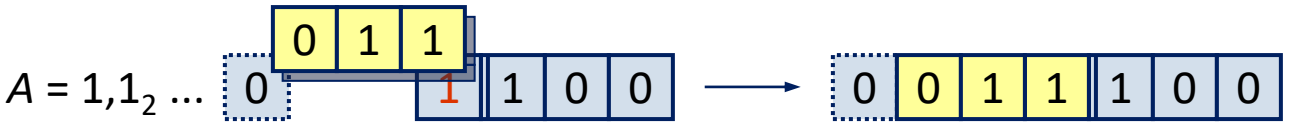
  ⇒ **normalizace**
  - tj. **úprava výsledku na normalizovaný tvar**
  - nutno provádět po každé operaci
- Př.



$A = 0{,}025_{10}$

...pohyblivá řádová čárka

- **Skrytá jednička**
  - předp. $z = 2$, normaliz. tvar, M přímý kód, $M \neq 0$, $\mathcal{A}(e) \neq 0$, aditivní konstanta 11 (3_{10})
    $\Rightarrow$ **v nejvyšším řádu mantisy bude vždy 1**
    $\Rightarrow$ tuto 1 můžeme „skrýt“ (tj. vynechat ze zápisu čísla v ř.m.)



  - V případě $\mathcal{A}(e) = 0$ se skrytá jednička nepoužívá!!!