

Násobení a dělení

INP 2025

FIT VUT v Brně



Násobení a násobičky

- Při násobení čísel v dvojkové soustavě můžeme násobit absolutní hodnoty čísel a pak doplnit do výsledku znaménko, anebo raději násobit přímo čísla se znaménkem.
- Vlastní násobení může být prováděno sekvenčně (postupně), anebo kombinačně (v jednom kroku). Rozlišujeme proto
 - sekvenční násobičky
 - kombinační násobičky
- Podle typu operandu rozlišujeme násobičky pracující v
 - pevné řádové čárce a
 - plovoucí řádové čárce
- Cílem implementace je získat součin co nejrychleji, za co nejnižší cenu (počet hradel, plocha čipu), popř. s co nejnižším příkonem.
- Pro pevnou řádovou čárku si ukážeme:
 - Princip násobení
 - Sekvenční násobičky
 - Kombinační násobičky
 - Princip urychlení – Boothovo překódování, Wallaceův strom
 - Násobení bez násobiček

Princip násobení (bez znaménka)

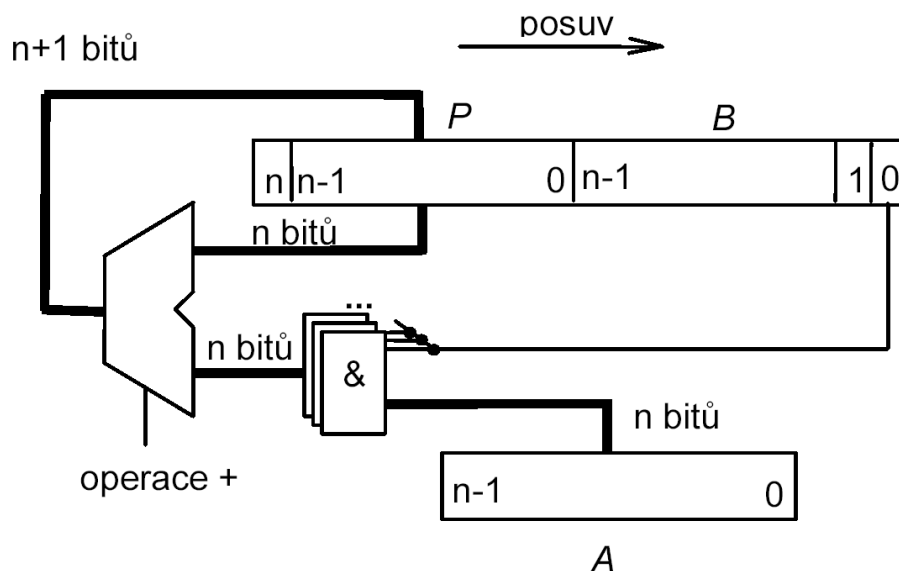
Mějme dvě čísla: N-bitový násobitel $x_{N-1}x_{N-2} \dots x_0$ a M-bitový násobenec $y_{M-1}y_{M-2} \dots y_0$.
Součin P potom bude:

$$P = \left(\sum_{j=0}^{M-1} y_j 2^j \right) \left(\sum_{i=0}^{N-1} x_i 2^i \right) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$

						y_5	y_4	y_3	y_2	y_1	y_0	Násobenec (multiplicand)
						x_5	x_4	x_3	x_2	x_1	x_0	Násobitel (multiplier)
						x_0y_5	x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0	] částečné součiny (partial products)
				x_1y_5	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0			
		x_2y_5	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0					
	x_3y_5	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0						
x_4y_5	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0							
x_5y_5	x_5y_4	x_5y_3	x_5y_2	x_5y_1	x_5y_0							
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	Součin (product)

Výsledek je na 12 bitech, obecně na $M+N$ bitech.

Sekvenční násobička



Př. $n=4$, $1111 \times 1011 = 10100101$
 ($\Rightarrow$ označuje posun vpravo)

Inicializace:

A: 1111

PB: 00000 1011

Krok 1:

PB: 00000 1011

+1111

PB: 01111 1011

$\Rightarrow$ 00111 1101

Krok 3:

PB: 01011 0110

+0000

PB: 01011 0110

$\Rightarrow$ 00101 1011

Krok 2:

PB: 00111 1101

+1111

PB: 10110 1101

$\Rightarrow$ 01011 0110

Krok 4:

PB: 00101 1011

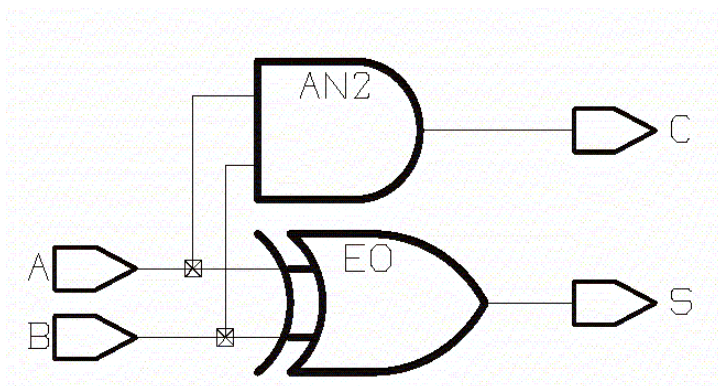
+1111

PB: 10100 1011

$\Rightarrow$ 01010 0101

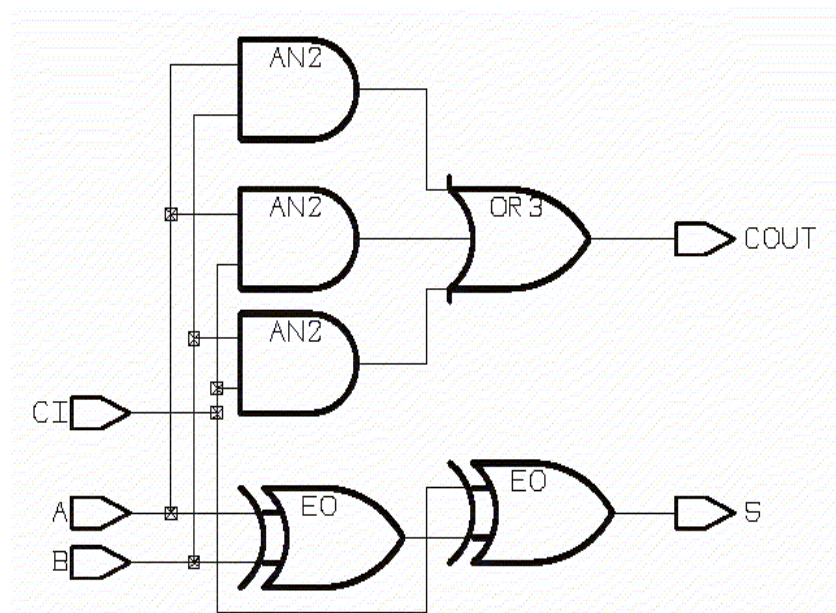
- U sekvenčního násobení čísel $A \times B$ se do dolní poloviny spojených registrů PB připraví násobitel B, do horní poloviny nuly, násobenec do registru A. Postup násobení je následující:
- (1) Nejnižším bitem registru PB se vynásobí násobenec A a přičte se k horní části registru PB, kde se udržuje průběžný součet dílčích součinů.
- (2) Obsah registrů PB se posune o jeden bit vpravo.
- Kroky 1, 2 provedeme celkem n -krát.
- Výhoda: nízký počet hradel. Nevýhoda: nízká rychlost, n taktů pro násobení

Obvodová realizace násobení je založena na sčítačkách



Poloviční sčítačka:

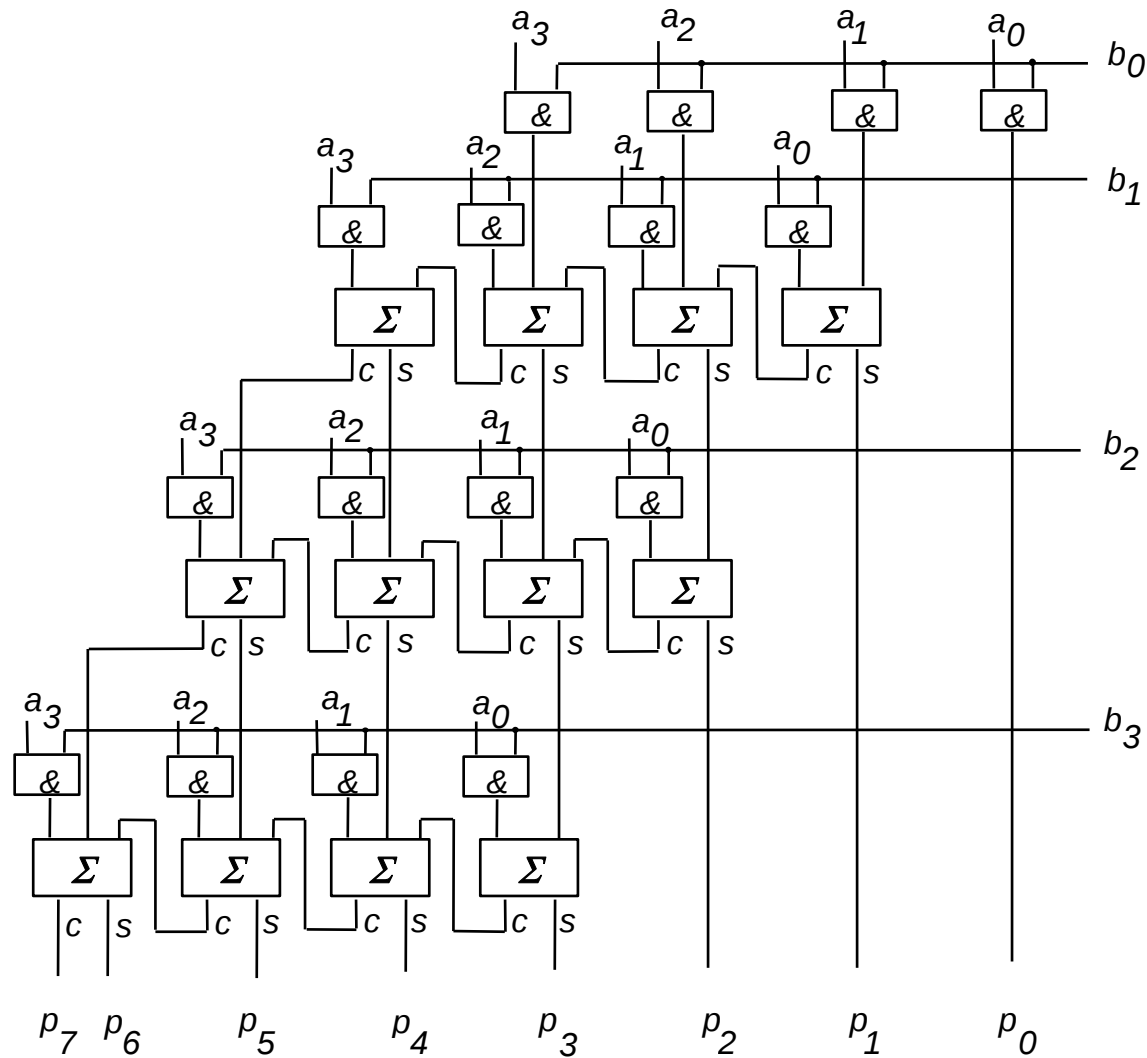
Zpoždění: 1 logický člen



Úplná sčítačka:

Zpoždění: 2 logické členy

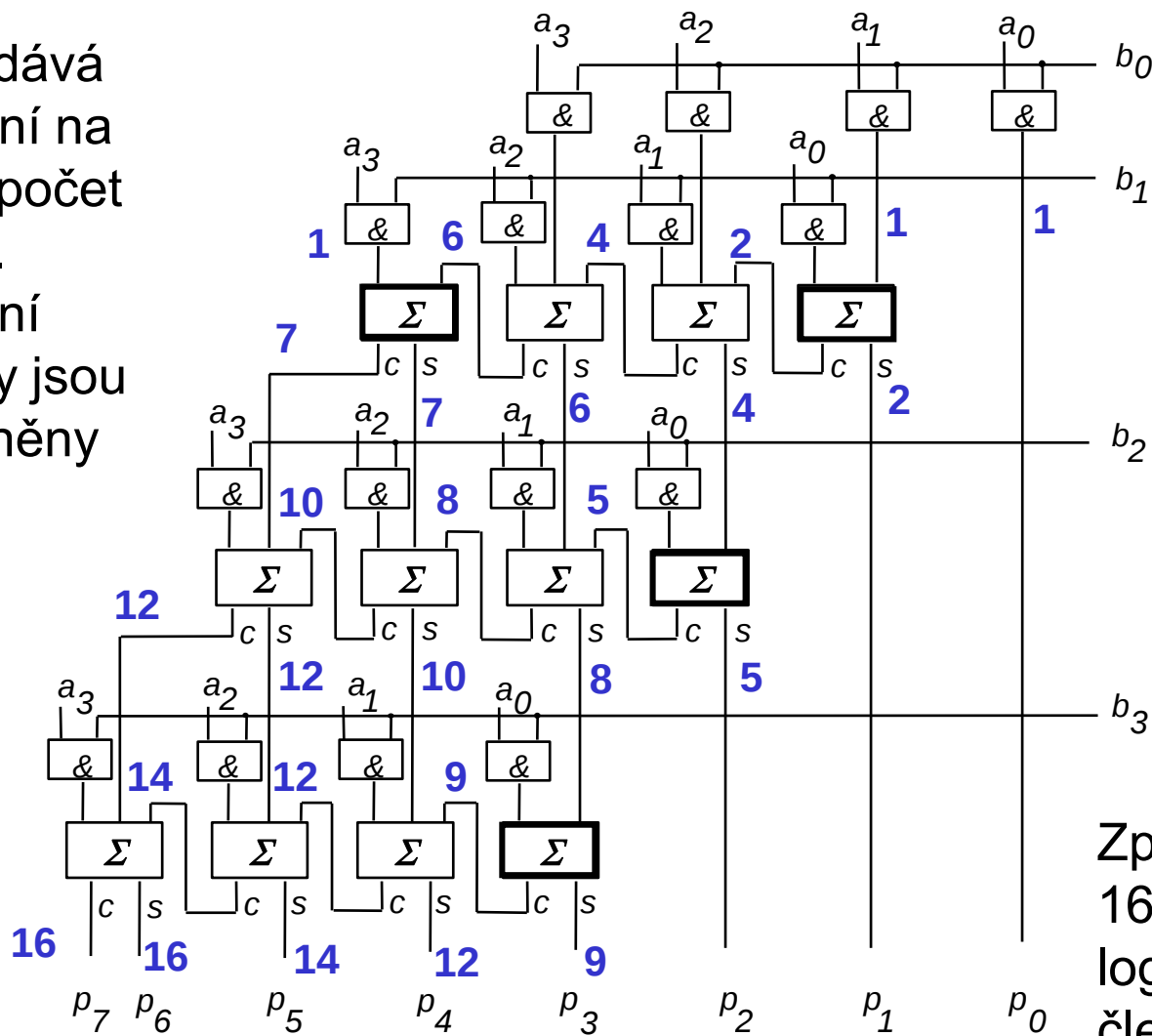
Kombinační násobička (v jednom kroku)



Kombinační násobička – odvození zpoždění

Číslo udává
zpoždění na
vodiči (počet
hradel).

Poloviční
sčítačky jsou
znázorněny
tučně.



Zpoždění:
16x zpoždění
logického
členu.

Popis ke kombinační násobičce

Jednotlivé dílčí součiny, tedy 1- a 0-násobky násobence, se tvoří řadou hradel a sčítají se na čtyřbitových sčítačkách s postupným přenosem.

Celkový **počet hradel** pro násobičku $n \times n$ bitů je $O(n^2)$, celkový **počet jednobitových sčítaček** je $(n-1) \times n$.

Zpoždění sčítačky určíme nalezením **nejdelší cesty** v kombinační síti. Je-li přenosové zpoždění hradla T a sčítačky $2T$, má **nejdelší cesta přenosové zpoždění $16T$ (nebo 8Δ , $\Delta = 2T$)**.

Toto zpoždění se s rostoucí délkou operandů dále zvětšuje. Je proto snaha nalézt uspořádání násobičky s rychlejší funkcí. Nejdříve se však zaměříme na násobení čísel se znaménkem.

Násobení čísel se znaménkem v doplňkovém kódu

$$P = (-y_{M-1}2^{M-1} + \sum_{j=0}^{M-2} y_j 2^j).(-x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i)$$

Příklad pro $M=N=6$

$$P = \sum_{i=0}^{N-2} \sum_{j=0}^{M-2} x_i y_j 2^{i+j}$$

$$x_{N-1} y_{M-1} 2^{M+N-2}$$

$$-\sum_{i=0}^{N-2} x_i y_{M-1} 2^{i+M-1}$$

+

$$-\sum_{j=0}^{M-2} x_{N-1} y_j 2^{j+N-1}$$

Příklad pro $M=N=6$

	y_5	y_4	y_3	y_2	y_1	y_0
	x_5	x_4	x_3	x_2	x_1	x_0
		x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0
	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0	
	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0	
	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0	
	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0	
x_5y_5						
1	1	$\overline{x_4y_5}$	$\overline{x_3y_5}$	$\overline{x_2y_5}$	$\overline{x_1y_5}$	$\overline{x_0y_5}$
1	1	$\overline{x_5y_4}$	$\overline{x_5y_3}$	$\overline{x_5y_2}$	$\overline{x_5y_1}$	$\overline{x_5y_0}$

Záporný číselný výraz, p

Záporný člen:
negace, přičtení 1

Výsledek je na 12 bitech, obecně na $M+N$ bitech.

Př. Násobení čísel se znaménkem - prakticky

$(-4) \times (-14)$ na 5 bitech v doplňkovém kódu. Protože je výsledek na 10 bitech, musíme důsledně **rozšiřovat znaménko** na 10 bitů – týká se záporných čísel.

$$\begin{array}{r} \begin{array}{cc} S & S \\ 1111111100 & \times 1111110010 \end{array} \\ \hline & 00000 \\ & 1111111100 \\ & 00000 \\ & 00000 \\ & 1111111100 \\ \hline & 1111111100 \\ & 1111111100 \\ & 1111111100 \\ & 1111111100 \\ & 1111111100 \\ \hline & 0000111000 \\ & S \end{array}$$

Násobení čísel se znaménkem - praxe

Co s tím? **Důsledná práce se znaménkem:**

- Výsledek násobení dvou čísel na n bitech bude na $2n$ bitech
- Tedy i vstupní operandy musí být správně zakódovány na $2n$ bitech
- Princip šíření hodnoty znaménkového bitu doleva - z toho vyplývají dva problémy:
 - **U dílčích součinů se objeví levostranné jedničky**
 - **Objeví se další dílčí součiny**
- Počet částečných součinů a šíření znaménkového bitu (1) lze redukovat pomocí Boothova algoritmu.
- Počet úrovní nutných pro sečtení částečných součinů lze redukovat pomocí Wallaceova stromu.

Princip Boothova překódování

Překódováním násobitele potenciálně velmi usnadníme násobení, protože zredukujeme počet částečných součinů a zbavíme se levostranných jedniček.

A*31	A	= A*32-A	A	násobenec
11111			10000-1	násobitel
	A		-A	
	A		A	
	A			
	A			
	A			
4 součty			1 součet	

Konvenční přístup

Boothova metoda

Boothovo překódování s radixem 2

Základem Boothova algoritmu je překódování násobitele do soustavy **relativních číslic**. Máme-li násobit číslem 31, tedy 00011111, dostaneme stejný výsledek, vynásobíme-li číslem $(32 - 1)$, což zapíšeme v soustavě relativních číslic, která používá číslice 0, +1, -1, jako:

$$\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{array}$$

Tento princip aplikujeme na dvojkové číslo opakovaně pro všechny skupiny jedniček, přičemž za skupinu jedniček považujeme i jedinou jedničku.

Příklad:

$$\begin{array}{cccccccccccccccc} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 \end{array}$$

Odtud můžeme sestavit překódovací tabulku. Kromě překódovaného bitu se řídíme ještě hodnotou **sousedního bitu vpravo**. Dostáváme tak základní Boothovo překódování, zvané též překódování s radixem 2.

Boothovo překódování s radixem 2

Překódovaná číslice	Sousední bit vpravo	Boothův kód
0	0	0
0	1	1
1	0	-1
1	1	0

Všimneme si překódování záporného násobitele, např. -6 na 5 bitech včetně znaménka. Toto číslo 11010 se překóduje na 0-11-10.

Rozšíříme-li zobrazení čísla -6 na 10 bitů, tedy na **11111**11010, pak v Boothově překódování se to projeví přidáním pěti levostranných nul, tedy dostaneme 000000-11-10.

Potom vznikají **nulové** částečné součiny, které usnadňují výsledné sčítání.

Příklad

13 x (-6) na 5 bitech vč. znaménka. Překódování násobitele ponecháme pouze na 5 bitech, protože levostranné nuly vyvolají vznik nulových a tedy bezvýznamných dílčích součinů.

13: 01101 6: 0 0 1 1 0

-13: 10011 -6: 1 1 0 1 0

Překódování násobitele 0-1 1-1 0

0 x 13 0000000000

-1 x 13 111110011

1 x 13 00001101

-1 x 13 1110011

0 x 13 000000

1110110010

S

-0001001110 tj. -78

Boothovo překódování s radixem 4

Je tedy odstraněn nepříznivý efekt záporného násobitele, zůstává však nutnost šíření znaménka u záporných dílčích součinů. Dalším požadavkem pro urychlení násobení je snížení počtu dílčích součinů. Oba tyto požadavky se řeší dalšími modifikacemi Boothova násobení.

Z jednoduchého Boothova překódování je odvozeno překódování "**2 bity najednou**", neboli *Boothovo překódování s radixem 4*.

1 1	0 1	0 1	1 0	0 1	původní číslo rozdělené do dvojic
2 1	2 1	2 1	2 1	2 1	váhy
0-1	1-1	1 0	-1 0	1-1	Boothovo překódování po jednom bitu
-1	1	2	-2	1	překódování "2 bity najednou"

Boothovo překódování – obecný počet bitů

Boothovo překódování lze definovat pro skupiny bitů libovolné velikosti. Obecný postup je takový: V každé skupině zopakujeme nejvyšší bit, a přičteme k nejnižšímu bitu nejvyšší bit ze skupiny vpravo. Výsledné číslo pak považujeme za relativní číslici v doplňkovém kódu.

Příklad pro radix 4 (tj. 2 bity najednou):

00	00	11	01	11	10	01	00	10	10
000	000	111	001	111	110	001	000	110	110
0	1	0	1	1	0	0	1	1	0

původní číslo
rozšíření znam. bitu
přičtení bitu zprava

000	001	111	010	000	110	001	001	111	110
0	+1	-1	2	0	-2	+1	1	-1	-2

doplňkový. kód rel. číslice
překódování s radixem 4

Z tohoto příkladu můžeme odvodit tabulku pro Boothovo překódování s radixem 4.

Boothovo překódování s radixem 4

Překódovaná skupina	Bit vpravo	Relativní číslice
00	0	0
00	1	+1
01	0	+1
01	1	+2
10	0	-2
10	1	-1
11	0	-1
11	1	0

Příklad – Boothovo překódování s radixem 4

13 x (-6):

13 01101 6: 00110

-13 10011 -6: 11010

potřebujeme sudý počet bitů,
rozšíříme znaménko na 111010

Boothovo překódování 0-11-10 po jednom bitu

0-1 -2 po dvou bitech

S

-2x13 1111100110 tj. -13 posunutá vlevo o 1 bit

-1x13 11110011 s krokem 2 bity

0x13 000000

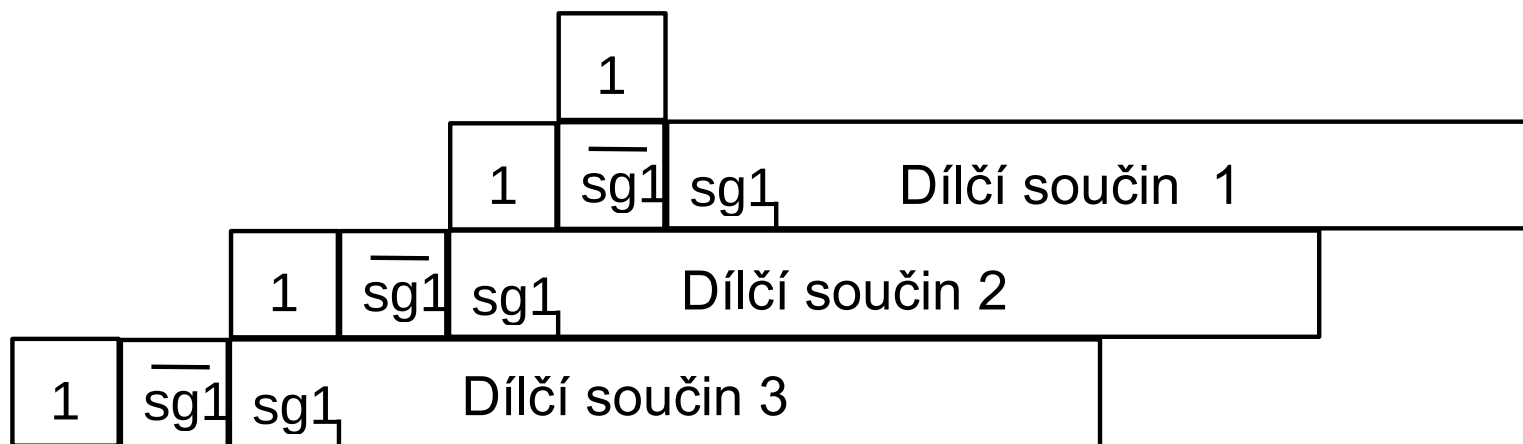
1110110010 **-78**

S

Komentář k příkladu $13 \times (-6)$

V prvním dílčím součinu se nám posunulo znaménko o jeden bit doleva. Tomu musíme přizpůsobit všechny ostatní dílčí součiny, tedy zapisovat je na 6 bitech. Znaménko výsledku očekáváme ovšem v 10. bitu.

Šíření znaménka vlevo odstraňuje [metoda vroubení](#). Místo šíření znaménka se ke každému dílčímu součinu přepíše zleva negace znaménkového bitu a jednička, a nad znaménkový bit prvního dílčího součinu se napíše též jednička.

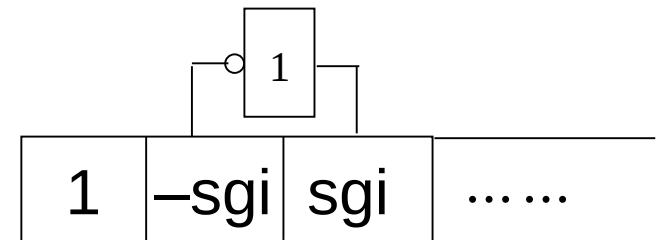


Př. 13 x (-6) s vroubením znaménka, 2 bity najednou

```

      001101
      0 -1 -2
-----
      1
    10100110
    10110011
    11000000
-----
    111110110010
      S
  
```

Obvodová realizace vroubení znaménka:



Boothovo překódování s radixem 8 (po 3 bitech)

Tabulka odvozena stejným způsobem jako pro radix 4.

Používá 9 relativních číslíc.

Překódovat	Rel. číslice
000 0	0
000 1	+1
001 0	+1
001 1	+2
010 0	+2
010 1	+3
011 0	+3
011 1	+4
100 0	-4
100 1	-3
101 0	-3
101 1	-2
110 0	-2
110 1	-1
111 0	-1
111 1	0

Jak urychlit kombinační násobičku?

Urychlit sečtení částečných součinů zavedením „uchování přenosů“

Příklad: Sečtěte $6 + 7 + 12 + 8$

$$\begin{array}{r} 110 \quad (6) \\ 111 \quad (7) \\ \hline 1101 \quad (13) \\ + 1100 \quad (12) \\ \hline 11001 \quad (25) \\ + 01000 \quad (8) \\ \hline 100001 \quad (33) \end{array}$$

Konvenční sčítání:

Sečtou se první dva operandy, potom se k průběžnému výsledku přičítají další operandy. V rámci sčítání dvou operandů se použije sčítačka s postupným přenosem složená z úplných sčítaček.

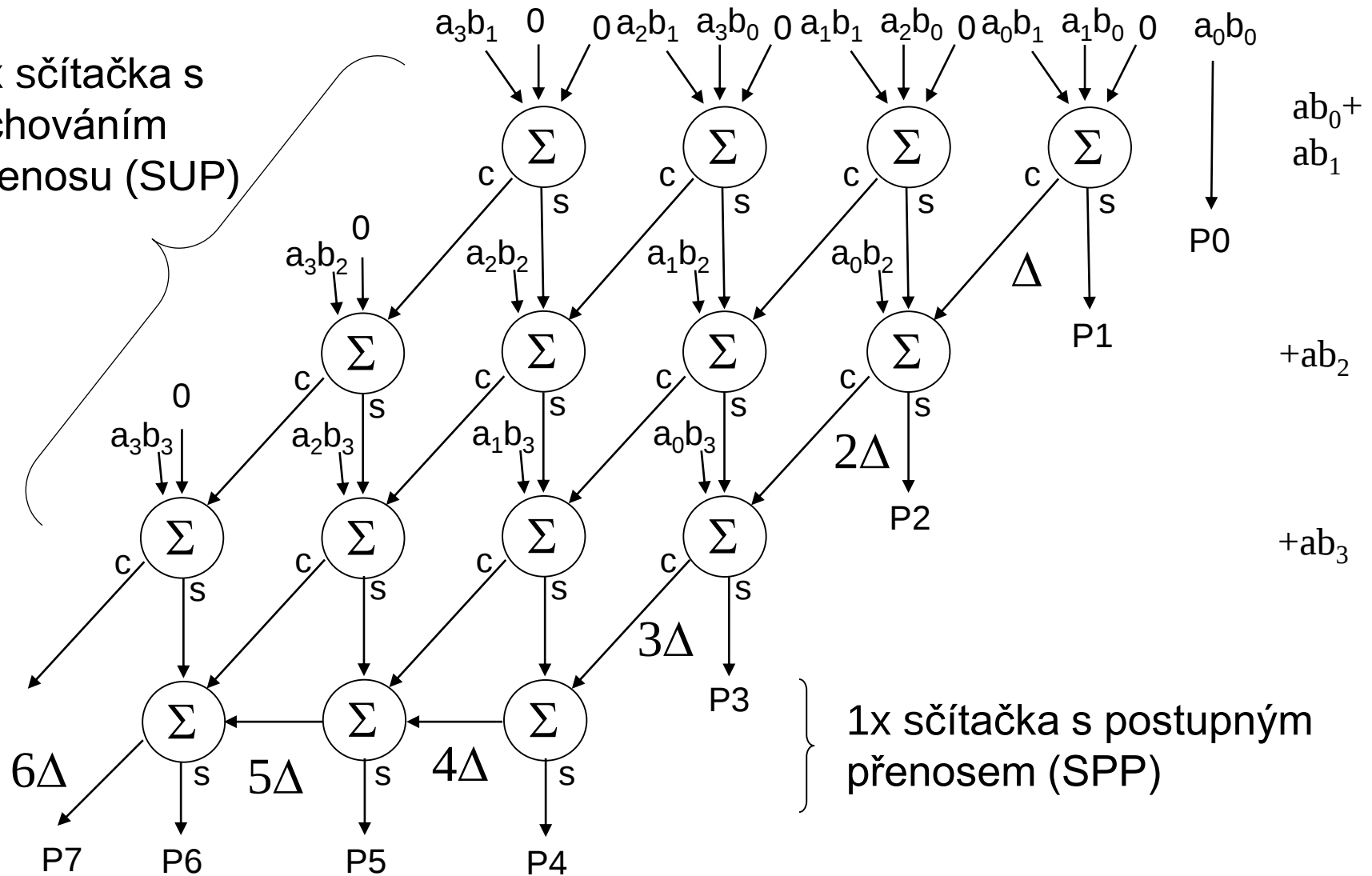
$$\begin{array}{r} 110 \quad (6) \\ 111 \quad (7) \\ \hline 001 \quad \text{součet bez přenosů (S1)} \\ 110 \quad \text{uchování přenosu (C1)} \\ + 1100 \quad (12) \\ \hline 0001 \quad \text{S1+C1+12 bez přenosů (S2)} \\ 1100 \quad \text{uchování přenosu (C2)} \\ + 01000 \quad (8) \\ \hline 100001 \quad \text{součet S2+C2+8 s přenosem} \end{array}$$

Sčítání s uchováním přenosu:

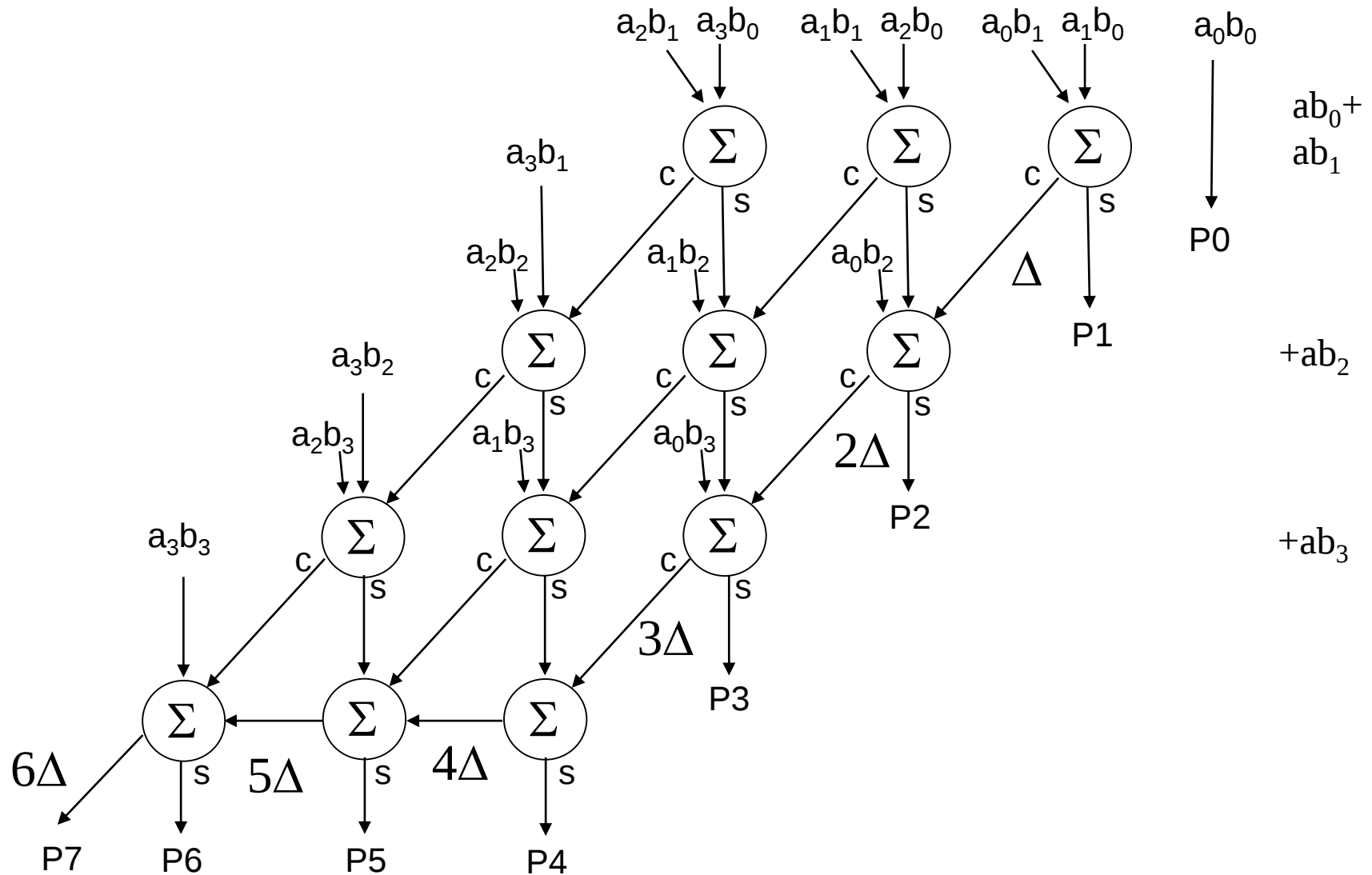
Sčítá se bez uvážení přenosů (tj. rychle). Avšak přenosy se uchovávají v registru a přičítají (pomocí úplných sčítaček) k průběžnému součtu a dalšímu operandu v následujícím kroku (opět bez přenosu). Přičtení posledního operandu se provede pomocí sčítačky s postupným přenosem.

4b kombinační násobička s uchováním přenosu

3x sčítačka s uchováním přenosu (SUP)



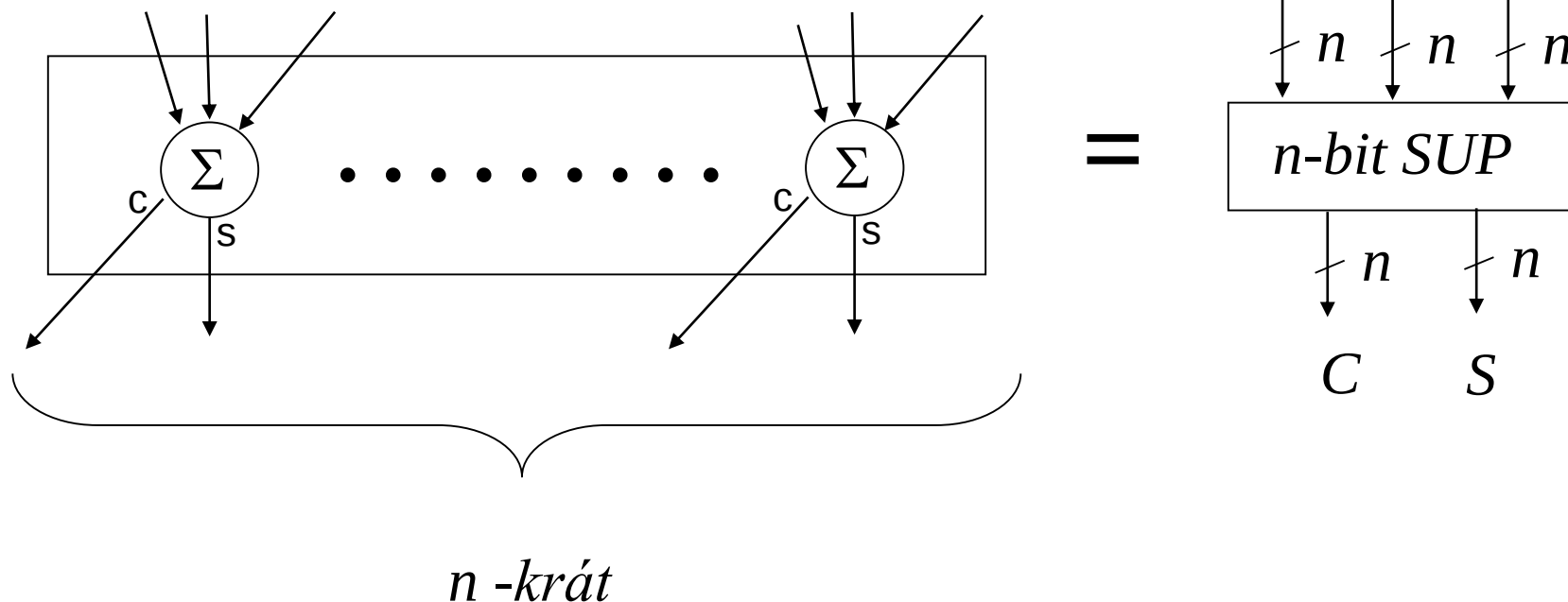
Optimalizovaná 4b kombinační násobička s uchováním přenosu



Komentář k násobičce s uchováním přenosů

Ze tří řad sčítacího stromu jsme odstranili sčítačky s postupným přenosem. Sčítačky v jedné řadě se označují jako **sčítačka s uchováním přenosu** (SUP, angl. CSA - Carry-Save Adder). V každém stupni se sčítačka v nejvyšším řádu redukovala na hradlo. Museli jsme však nakonec jednu řadu sčítaček s postupným přenosem (SPP) přidat. Nicméně toto uspořádání činnost násobičky obecně zrychluje.

Základem n -bitové násobičky s uchováním přenosu je n -bitová SUP:



Zobecněné využití sčítačky

Sčítačka s uchováním přenosu dovoluje chápat sčítání poněkud odlišným způsobem. Víme, že jednobitová sčítačka má tři vstupy a dva výstupy. Vzhledem k tomu, že vstupy pro přenos všech jednobitových sčítaček v prvním stupni jsou nevyužité, může se na ně přivést třetí dílčí součin.

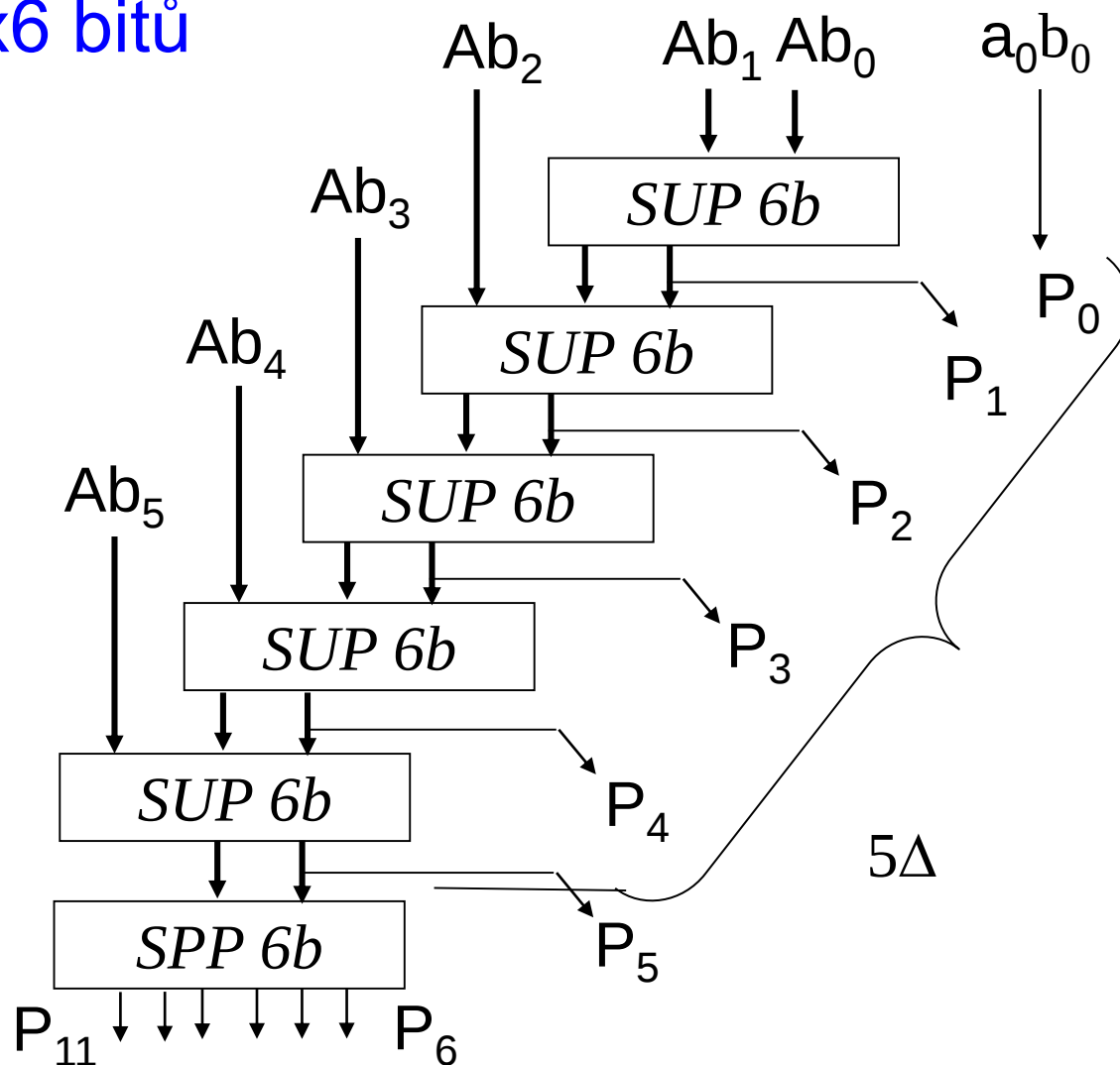
Na vstupech má tedy sčítačka tři binární vektory délky n , ovšem vzájemně posunuté do správné polohy. To se projeví tak, že v každém stupni je plný počet sčítaček (n). Součet tří dílčích součinů je dán výstupním vektorem součtu a výstupním vektorem přenosů. Poloha vektoru součtu je vzhledem ke konečnému výsledku správná, vektor přenosů se posouvá o jeden bit do vyššího řádu.

Takováto zobecněná **optimalizovaná sčítačka** je známá též pod označením **pseudosčítačka**, protože nedává ještě definitivní součet vstupních vektorů.

Blokové schéma násobičky s uchováním přenosu 6x6 bitů

$$A = a_5a_4a_3a_2a_1a_0$$

$$B = b_5b_4b_3b_2b_1b_0$$

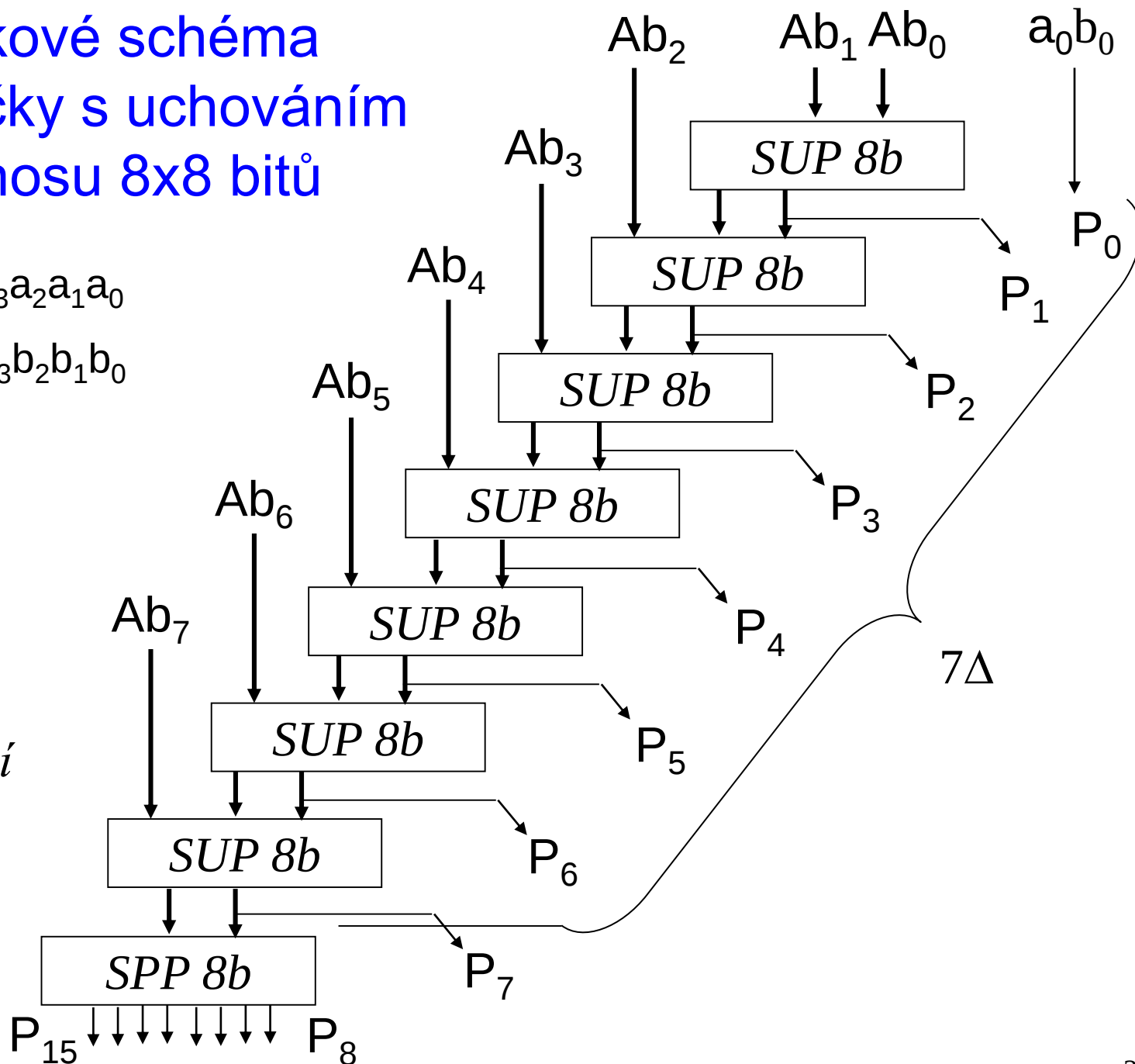


Blokové schéma násobičky s uchováním přenosu 8x8 bitů

$$A = a_7a_6a_5a_4a_3a_2a_1a_0$$

$$B = b_7b_6b_5b_4b_3b_2b_1b_0$$

Nebylo by možné zredukovat počet sčítání a tím i zpoždění?

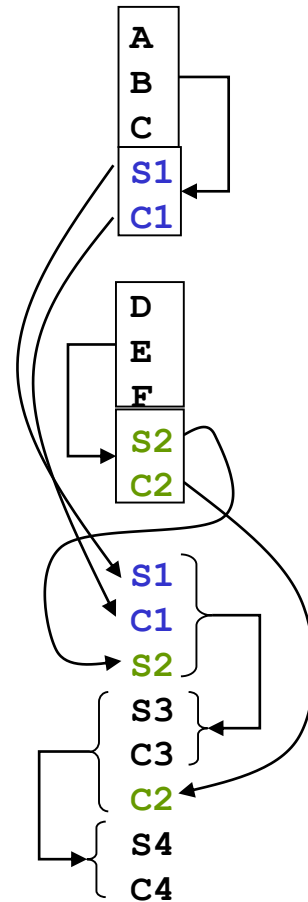


Zrychlené sčítání částečných součinů (A, B, C, D, E, F)
při násobení s uchováním přenosu $M \times Q$ ($6b \times 6b$)

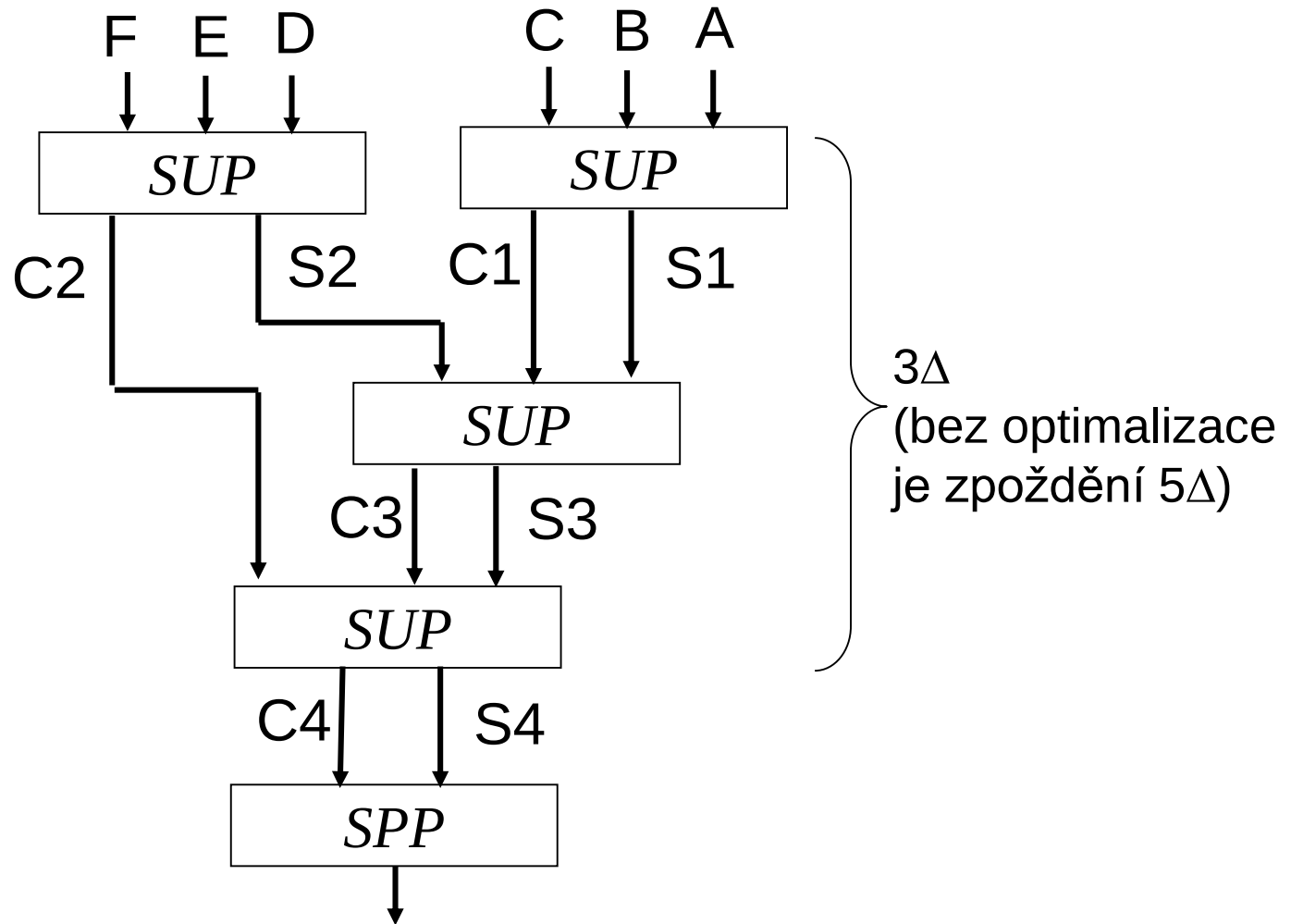
Nejdříve jsou sečteny
částečné součiny A, B a C,
výsledek je v S1 a C1. Potom
jsou sečteny částečné součiny
D, E a F, výsledek je v S2 a
C2. Následuje součet S1, C1 a
S2, výsledek je v S3 a C3.
Nakonec jsou sečteny S3, C3
a C2, výsledek je v S4 a C4.
Sčítačkou s postupným
přenosem jsou nakonec
sečteny S4 a C4.

Kromě posledního sčítání jsou všechna ostatní sčítání s uchováním přenosu.

$$\begin{array}{r}
 \begin{array}{cccccccc}
 & & & 1 & 0 & 1 & 1 & 0 & 1 & M \\
 x & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{1} & Q
 \end{array} \\
 \\
 \begin{array}{cccccccc}
 & & & 1 & 0 & 1 & 1 & 0 & 1 \\
 & & 1 & 0 & 1 & 1 & 0 & 1 \\
 & 1 & 0 & 1 & 1 & 0 & 1 \\
 \hline
 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0
 \end{array} \\
 \\
 \begin{array}{cccccccc}
 & & & 1 & 0 & 1 & 1 & 0 & 1 \\
 & 1 & 0 & 1 & 1 & 0 & 1 \\
 1 & 0 & 1 & 1 & 0 & 1 \\
 \hline
 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0
 \end{array} \\
 \\
 \begin{array}{cccccccc}
 & & & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
 & & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
 \hline
 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
 \hline
 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
 +0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 \hline
 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
 \end{array}
 \end{array}$$

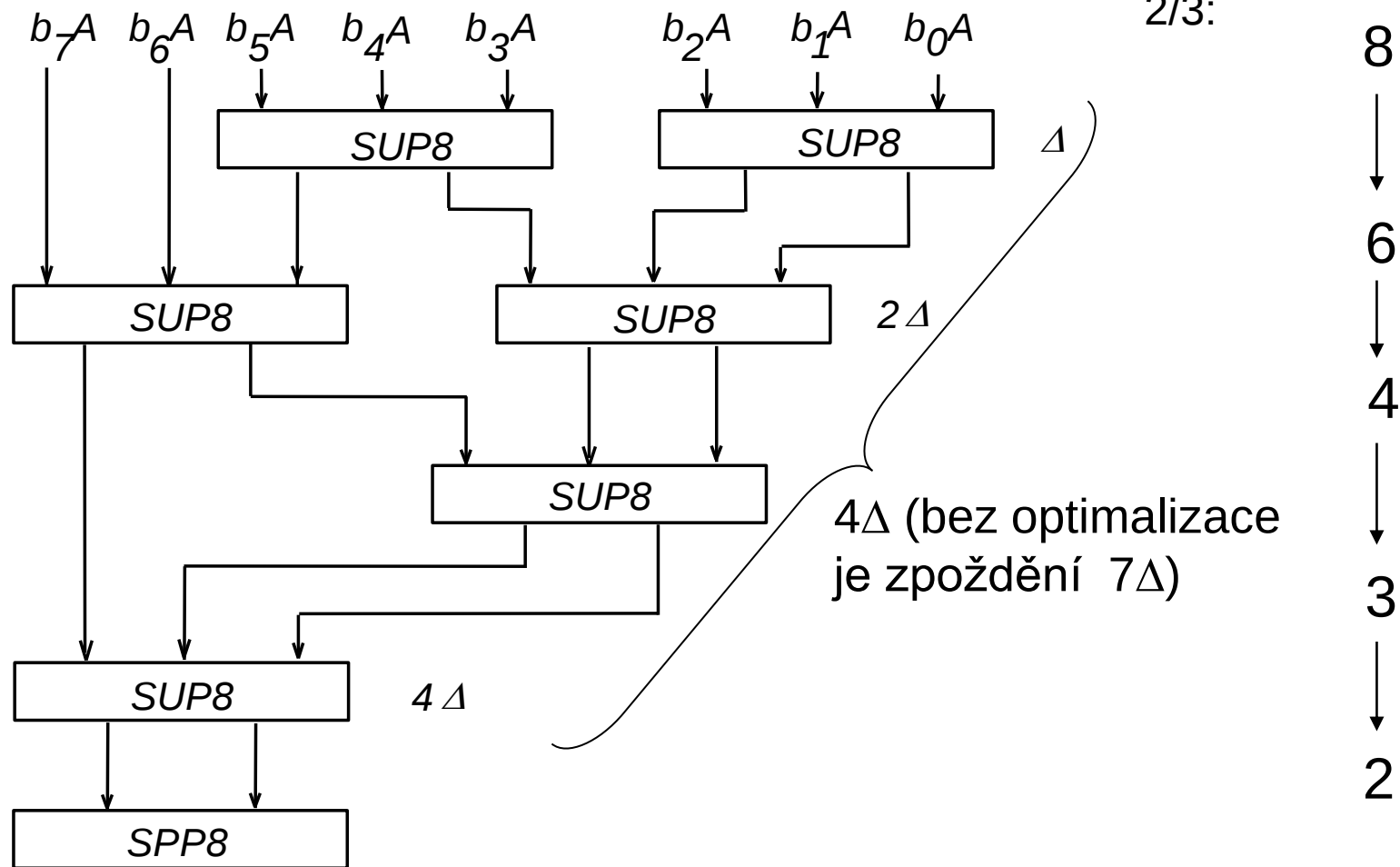


Zrychlené sčítání částečných součinů při násobení s uchováním přenosu 6x6 bitů – Wallaceův strom

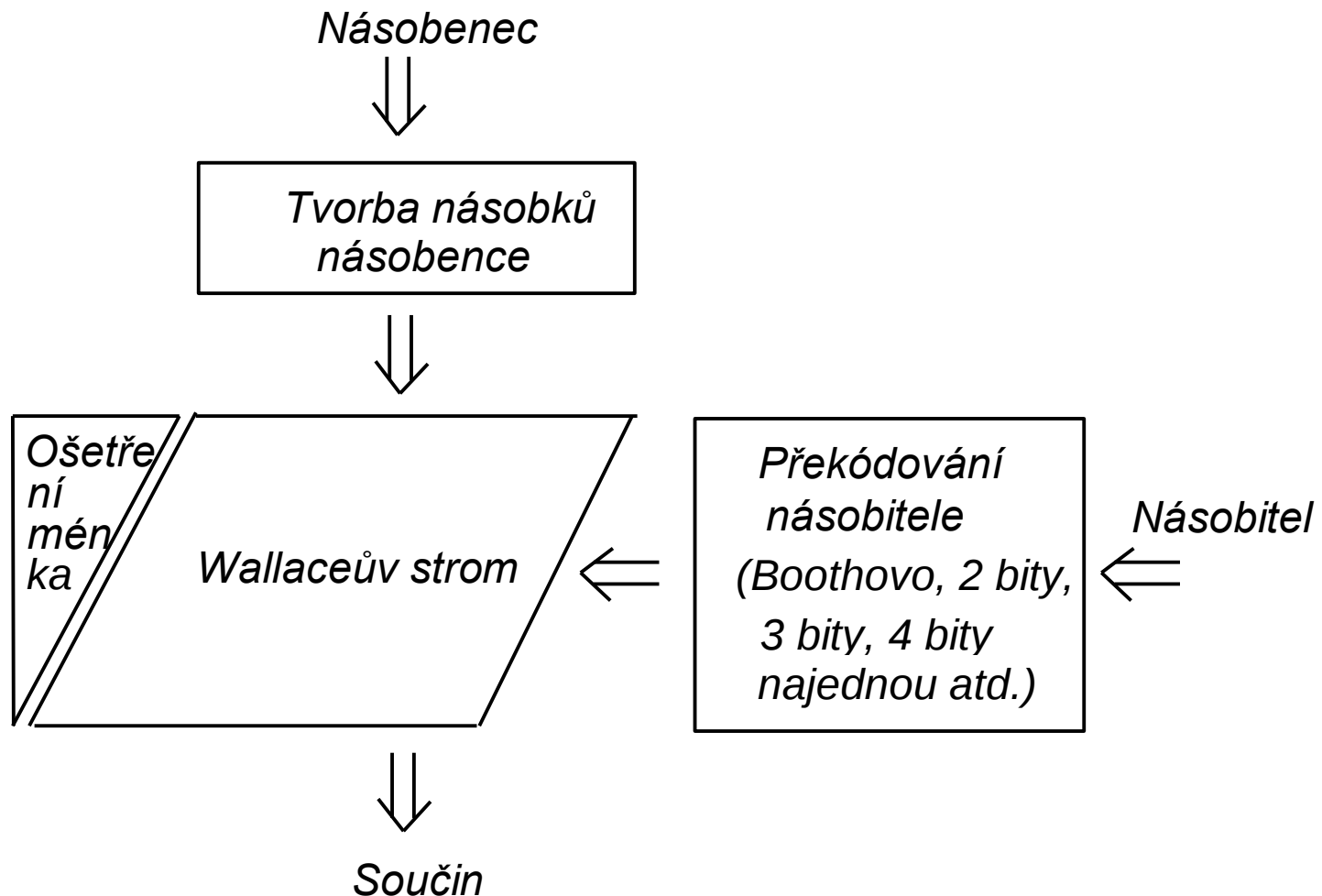


Wallaceův strom pro násobičku 8x8 bitů

V každém stupni (SUP) je redukován počet operandů na 2/3:

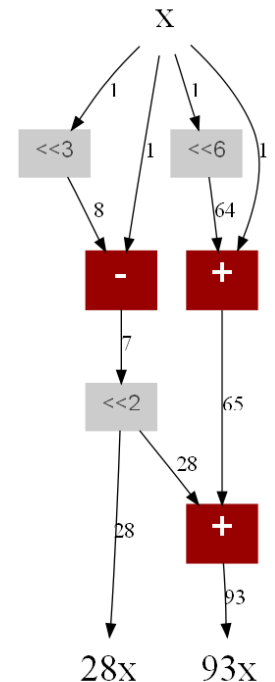
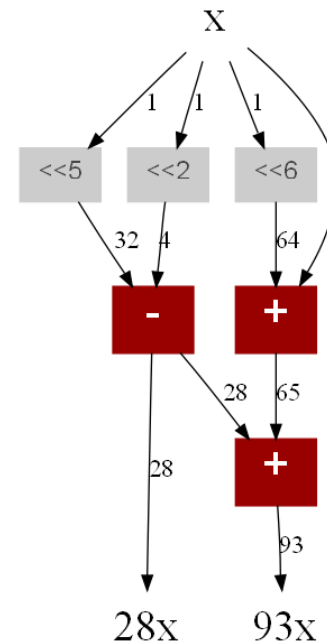
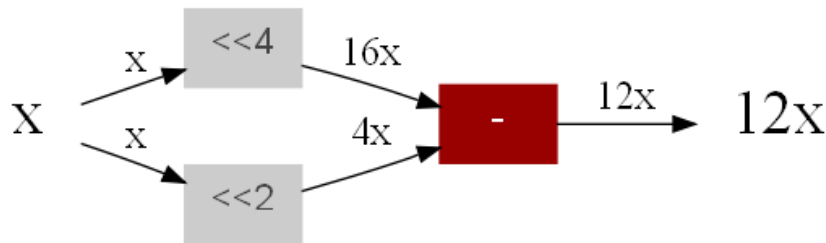


Shrnutí: Kombinační násobička čísel se znaménkem



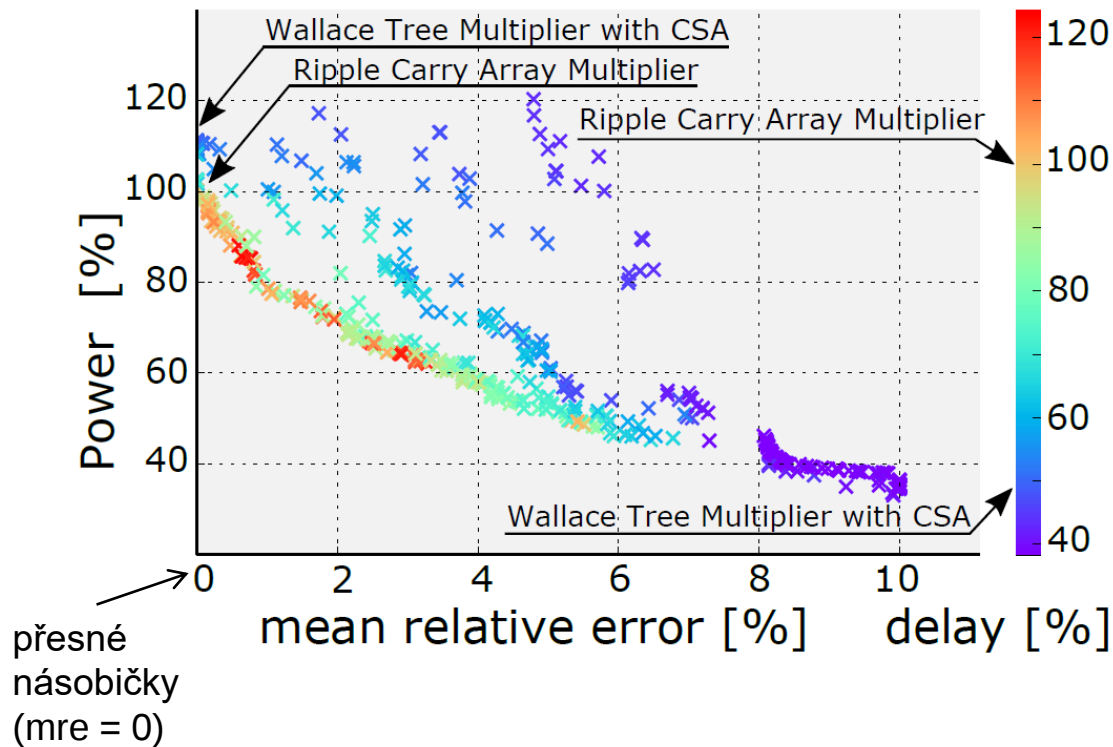
Násobení konstantou

- Násobení $C \times x$, kde C je konstanta, se často implementuje bez násobičky, pouze pomocí sčítání, odčítání a posuvů, což je úspornější. Posuv zde má nulovou cenu!
- Obdobně lze také implementovat násobení $x \times y$, pokud víme, že y nabývá několika málo předem známých hodnot.
- Použití: zpracování signálů a obrazů, neuronové sítě ...
- Příklady:
 - $12 \times x$
 - $x \times y$, kde $y \in \{28, 93\}$



Aproximace aritmetických operací

- Pokud menší chyby ve výpočtu nemají významný dopad na kvalitu výstupu dané aplikace (např. zpracování obrazu, rozpoznávání pomocí neuronových sítí apod.), je užitečné aproximovat aritmetické operace a tím buď ušetřit energii nutnou pro výpočet nebo urychlit výpočet.
- Př. 8-bit násobičky bez znaménka (45 nm, 1V)



Dělení a děličky

Dělení a děličky

- **Sekvenční dělička** pro čísla **bez znaménka**
 - Je založena na postupném **odčítání** dělitele od průběžného zbytku (n kroků pro n-bitový dělitel)
- **Sekvenční dělička** pro čísla **se znaménkem**
 - Algoritmus SRT (Sweeney, Robertson, Tocher, 1958)
 - Je založen na postupném odčítání/přičítání dělitele od/k průběžnému zbytku. Používá soustavu relativních číslic pro podíl a radix vyšší než 2.
 - Výběr operace (přičtení nebo odečtení dělitele) a číslice výsledku jsou určeny v každém kroku podle několika bitů průběžného zbytku a dělitele (předpočítáno v tabulce).
 - Například v Pentiu se určují číslice podílu podle 7 bitů průběžného zbytku a podle 5 bitů hodnoty dělitele. Používá se radix 4.
- **Kombinační dělička**
 - založena na úplné odčítačce
 - obvodová struktura je podobná kombinační násobičce
- **Iterační dělička** a dělení v **pohyblivé řádové čárce**
 - další přednáška

Dělení čísel s pevnou řádovou čárkou

Budeme se zabývat dělením čísel s pevnou řádovou čárkou **bez znaménka**. Pro jednotlivé činitele operace dělení zavedeme symboly

D	dělenec
d	dělitel
Q	podíl
q_i	i-tý bit podílu
R_i	i-tý (průběžný) zbytek

Máme vypočítat **Q**, **R** tak, aby byla splněna rovnice

$$D = Q \cdot d + R, \quad 0 \leq |R| < d.$$

Pro d, Q, R máme k dispozici n bitů, pro D vyhradíme 2n bitů.

Nejdříve si ukážeme dělení čísel bez znamének, resp. dělení jejich absolutních hodnot.

Pozor, d je nutné posunout o **n bitů doleva**.

Příklad 2293 : 51 (dekadicky vs binárně, n = 6) *Posunutý dělitel*

$$\begin{array}{r} \underline{22}93 : 51 = 0 \\ -00 \end{array}$$

$$\begin{array}{r} \underline{100011}110101 : 110011000000 = 0 \quad (i = 0) \\ -000000 \end{array}$$

$$\begin{array}{r} \underline{229}3 : 51 = 04 \\ -204 \end{array}$$

$$\begin{array}{r} \underline{1000111}10101 : 11001100000 = 01 \\ -110011 \end{array}$$

$$\begin{array}{r} 0\underline{253} : 51 = 044 \\ -204 \end{array}$$

$$\begin{array}{r} \underline{0101001}0101 : 1100110000 = 010 \\ -000000 \end{array}$$

49 (zbytek)

$$\begin{array}{r} \underline{01010010}101 : 110011000 = 0101 \\ -110011 \end{array}$$

$$\begin{array}{r} \underline{111111}01 : 11001100 = 01011 \\ -110011 \end{array}$$

$$\begin{array}{r} \underline{0011000}1 : 1100110 = 010110 \\ -000000 \end{array}$$

$$\begin{array}{r} 00\underline{110001} : 110011 = 0101100 \quad (44) \end{array}$$

110001 (49, zbytek)

V kroku i se
pokoušíme odečíst
od průběžného
zbytku R_i posunutý
dělitel $2^{n-i} d$

Postup dělení a základní HW realizace

Při rozhodování o hodnotě bitu podílu q_{n-i} jsme postupovali podle vztahů:

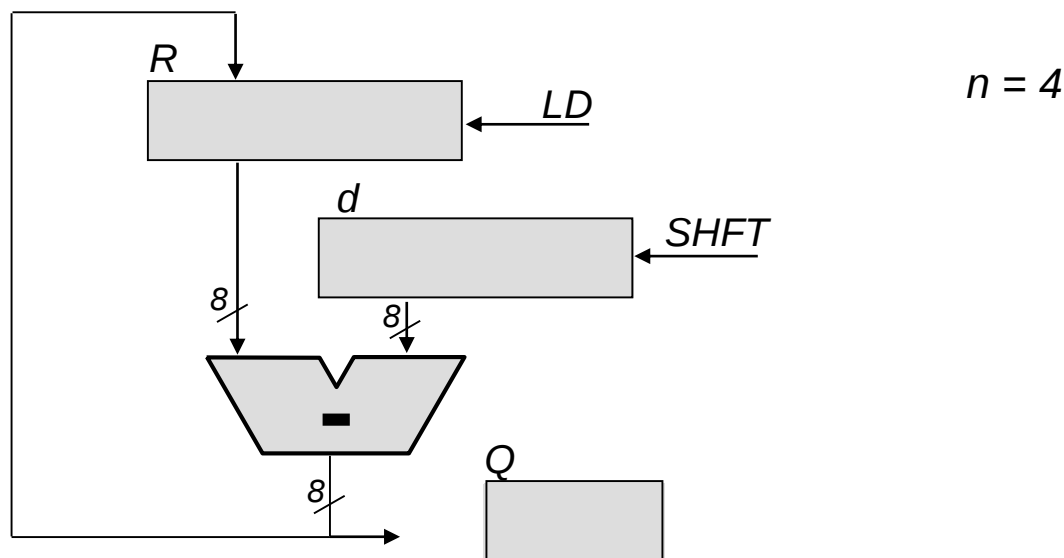
je-li $2^{n-i} d$ **menší než nebo rovno** R_i , pak $q_{n-i} = 1$,

je-li $2^{n-i} d$ **větší** než R_i , pak $q_{n-i} = 0$.

$$Q = q_n \dots q_0$$

Nový zbytek se vypočte:

$$R_{i+1} := R_i - q_{n-i} 2^{n-i} d$$

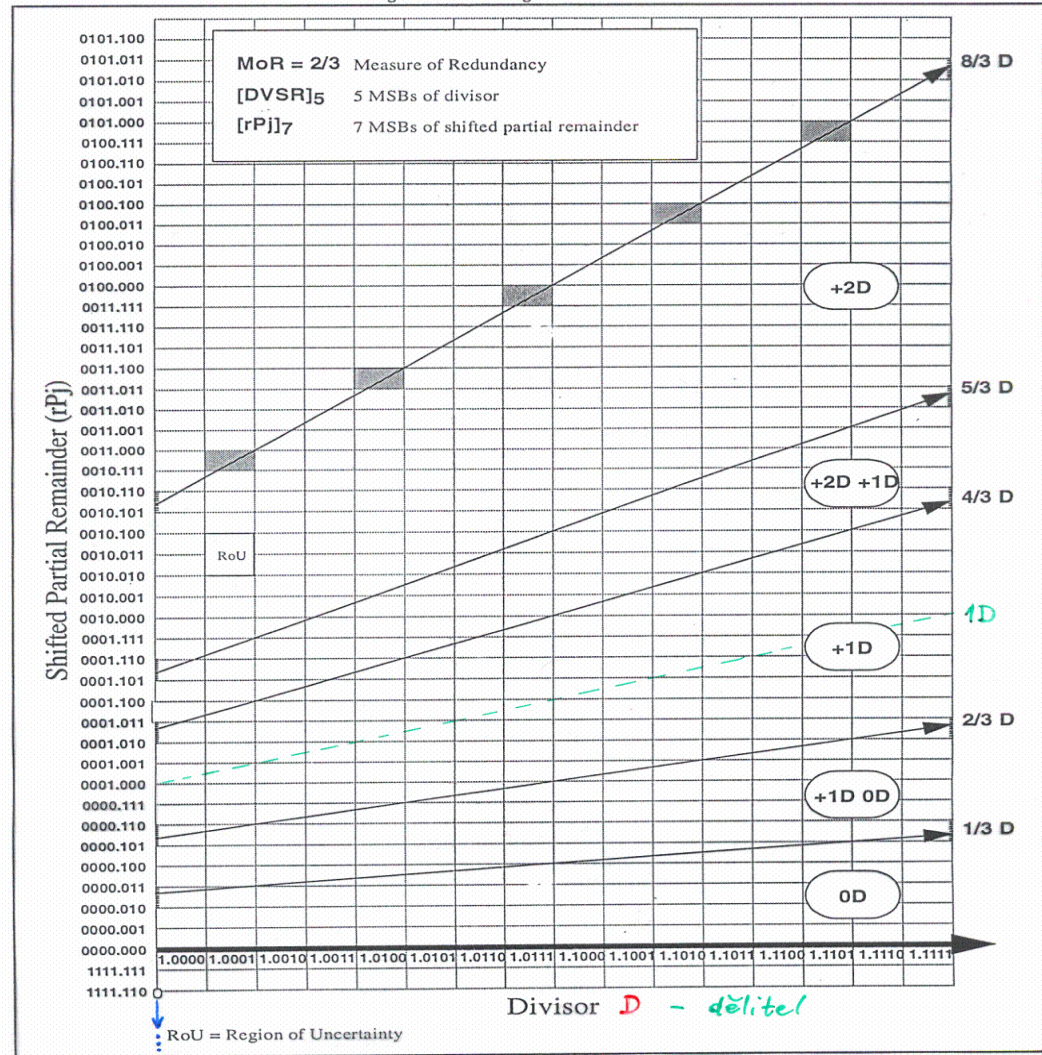


Vylepšení HW realizace: (1) d v pevné poloze, posunuje se R ; (2) přičítání d , pokud je R záporný; (3) algoritmus SRT řízený tabulkou

index má 7+5=12 bitů
(7-bitový odhad podle přibližného zbytku + ...)

Chyba dělení u
Pentia
(listopad 1994,
ztráta \$475M)

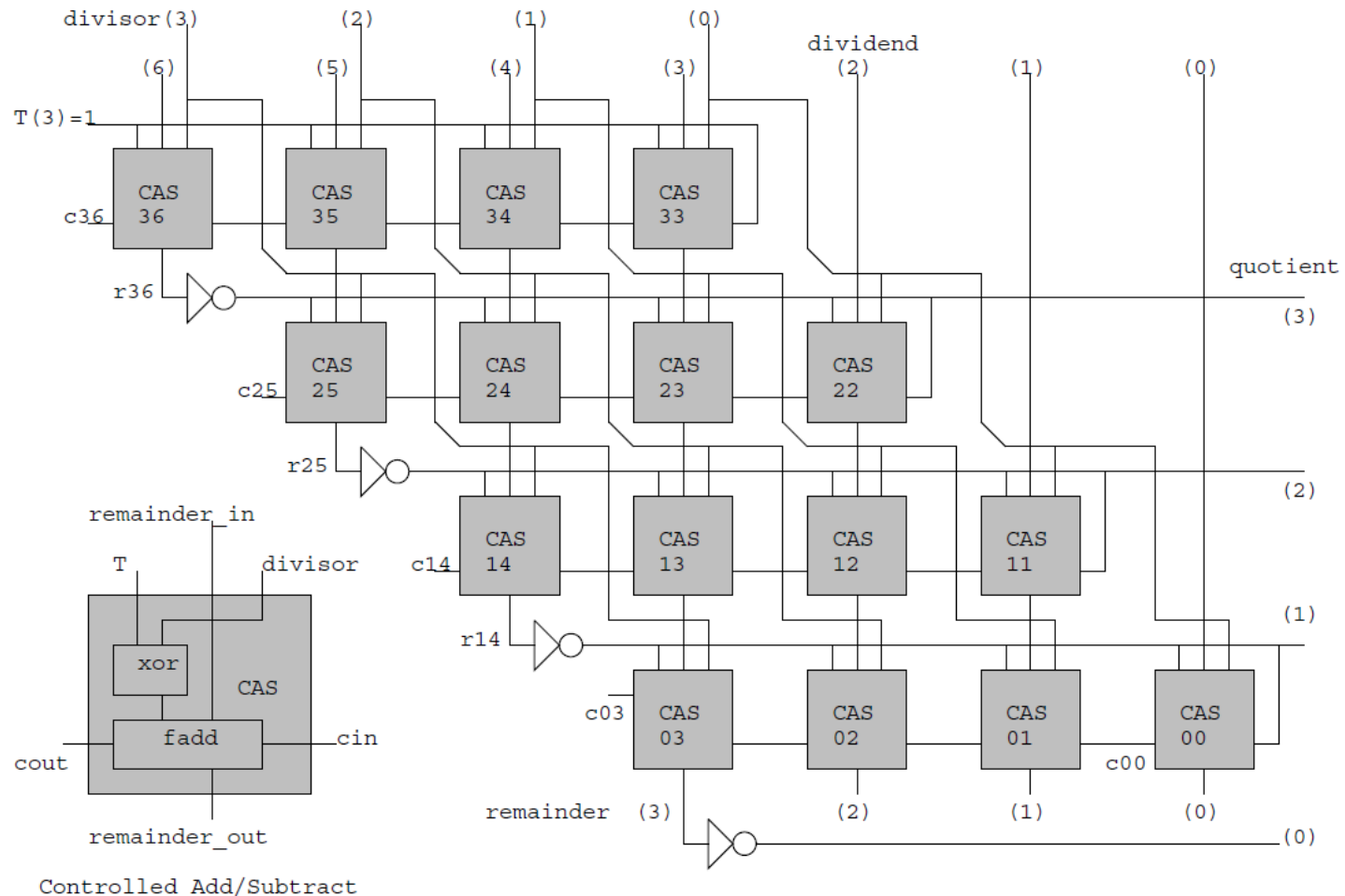
Figure 4-3 Missing Terms in P-D Plot



! D ~ d !

Princip realizace kombinační 4b děličky

(bez návratu k nezápornému zbytku)



<http://www.cs.umbc.edu/portal/help/VHDL>

- Základní stavební prvek: konfigurovatelná sčítačka/odčítačka
- T je signál řízení: sčítání (T=0, cin = 0), odčítání (T=1, cin = 1)

Použitá literatura

- Drábek, V.: Výstavba počítačů. Skriptum VUT, 1995
- Weste, N. H. E., Harris, D.: CMOS VLSI Design, 3. vydání, Addison Wesley, 2005
- Hamacher, C., Vranesic, Z., Zaky, S.: Computer Organization. McGraw-Hill, 2001